Savitribai Phule Pune University

S. Y. B. B. A. (C. A.) Semester-III

(CBCS 2019 Pattern)

Data Structure, Angular JS, PHP, Big Data and Block Chain

CA-306: Lab Book

Student Name: _____

College Name: _____

Roll No.: _____ Division: _____ Seat No: _____

Academic Year: _____

CERTIFICATE

This is to certify that Mr./Ms._____

Seat Number _____ of S.Y.B.B.A. (C.A) Sem-III has successfully completed Laboratory course (Data structure, Angular JS/PHP, Big Data/ Block Chain) in the year _____ .

He/She has scored _____mark out of 10 (For Lab Book).

Subject Teacher **H.O.D./Coordinator**

Internal Examiner **External Examiner**

Introduction

1. About the work book:

This workbook is intended to be used by S.Y.B.B.A. (C.A.) Semester-III students for Data structure, Angular JS, PHP, Big data, Block chain Practical assignments. This workbook is designed by considering all the practical topics mentioned in syllabus.

2. The objectives of this workbook are:

- Defining the scope of the course.
- To bring the uniformity in the practical conduction and implementation in all colleges affiliated to SPPU.
- To have continuous assessment of the course and students.
- Providing ready reference for the students during practical implementation.
- Provide more options to students so that they can have good practice before facing the examination.
- Catering to the demand of slow and fast learners and accordingly providing the practice assignments to them.

3. How to use this workbook:

The workbook is divided into five sections. Section-I is related to Data Structure assignments, Section-II is related to Angular JS assignments, Section-III is related to PHP assignments, Section-IV is related to Big Data assignments and Section-V is related to Block Chain assignments.
Section-I: Data Structure is divided into nine assignments.
Section-II: Angular JS is divided into assignments.
Section-III: PHP is divided into assignments.
Section-IV: Big Data divided into assignments
Section-V: Block Chain divided into assignments.
Students have to perform practical assignment of selected elective subject both from Section-II or Section-III and from Section IV or Section V.
Each assignment of all sections has three SETs-A, B and C. It is mandatory for students to complete SET A and SET B in lab. It also includes practice programs which are expected to be solved by students as home assignments and to be evaluated by subject teachers.

4. Instructions to the students

Please read the following instructions carefully and follow them during practical.

- Students are expected to carry this workbook every time they come to the lab for computer practical.
- Students should prepare for the assignment by reading the relevant material which is

mentioned in ready reference and the concepts taught in class.

- Instructor will specify which problems to solve in the lab during the allotted slot and student should complete them and get verified by the instructor. However, student should spend additional hours in Lab and at home to cover all workbook assignments if needed.
- Students will be assessed for each exercise on a scale from 0 to 5.

Notdone	0
Incomplete	1
Late Complete	2
Needs improvement	3
Complete	4
WellDone	5

5. Instruction to the Instructors

Make sure that students should follow above instructions.

- Explain the assignment and related concepts in around ten minutes using whiteboard if required or by demonstrating the software.
- Evaluate each assignment carried out by a student on a scale of 5 as specified above by ticking appropriate box.
- The value should also be entered on assignment completion page of the respective Lab course.

6. Instructions to the Lab administrator

You have to ensure appropriate hardware and software is made available to each student.

The operating system and software requirements on server side and also client side are as given below:

- Operating System - Windows
- Turbo C
- Java script
- WampServer
- RStudio

Assignment Completion Sheet

Section-I: Data Structure			
Sr. No.	**Assignment Name**	**Marks (out of 5)**	**Teacher's Sign**
1	Array		
2	Sorting Techniques (Non Recursive)		
3	Sorting Techniques (Recursive)		
4	Searching Techniques		
5	Linked List		
6	Stack		
7	Queue		
8	Trees		
9	Graph		
Total (Out of 45)			
Total (Out of 4)			

Instructor Signature:

Section-II: Angular JS			
Sr. No.	**Assignment Name**	**Marks (out of 5)**	**Teacher's Sign**
1	Introduction to Angular JS, AngularJS Directives , Expressions, Events		
2	AngularJS Modules, Controller, View and Scope		
3	Filter, Forms Validation		
4	AngularJS Services		
Total (Out of 20)			
Total (Out of 3)			

'OR'

Section-III: PHP			
Sr. No.	**Assignment Name**	**Marks (out of 5)**	**Teacher's Sign**
1	Basics in PHP		
2	Control Structures and Loops		
3	Arrays and Strings		
4	Functions, Class, and Object		
5	Working With Form and form element		
6	Session and Cookies		
7	Database		
Total (Out of 35)			
Total (Out of 3)			

Instructor Signature:

Section-IV: Big Data			
Sr. No.	**Assignment Name**	**Marks (out of 5)**	**Teacher's Sign**
1	Basic R Programming		
2	Decision making and loop control structures		
3	Sting and Function in R Programming		
4	Vector and List in R Programming		
5	Array and Matrices in R Programming		
6	Factor and Data Frame in R Programming		
7	Data Analysis		
8	Data Visualization		
Total (Out of 40)			
Total (Out of 3)			

'OR'

Section-V: Block Chain			
Sr. No.	**Assignment Name**	**Marks (out of 5)**	**Teacher's Sign**
1	Working with Blockchain		
2	Implementation of Smart Contracts and Hyperledger		
Total (Out of 10)			
Total (Out of 3)			

Instructor Signature:

Section-I

Data Structure

Assignment No 1: Array (One Dimensional Array)

ARRAY

- An array is a finite ordered collection of homogeneous data elements which provide random access to the elements.

  Finite: - There are specific no. of elements in the array.

  Ordered: - The elements are arranged one by one i.e. first then second and so on.

  Homogeneous: - All the elements are of same type.

WHAT IS POYNOMIAL

- A polynomial p(x) is the expression in variable x which is in the form ($ax^n + bx^{n-1} + \ldots. + jx + k$), where a, b, c …., k fall in the category of real numbers and 'n' is non negative integer, which is called the degree of polynomial.
- **An essential characteristic of the polynomial is that each term in the polynomial expression consists of two parts:**
  - one is the coefficient
  - other is the exponent

EXAMPLE:

- $10x^2 + 26x$, here 10 and 26 are coefficients and 2, 1 is its exponential value.

Practice Program:

1) Write a menu driven C program to perform the following operation on an integer array:
   a) Display the sum of elements at even subscript position of array
   b) Display the sum of elements at odd subscript position of array
2) Write a C Program to find the largest pair sum in an unsorted array.(hint: find 2 maximum elements from array and then find the sum of both numbers.)
3) Write a C Program to calculate Median of two sorted arrays of different sizes.

SET A:

1) Write a C Program to Count number of occurrences (or frequency) in a given sorted array

   Input: arr[] = {1, 1, 2, 2, 2, 2, 3,}, x = 2
   Output: 4 // x (or 2) occurs 4 times in arr[]
2) Write a C program to accept n elements, store those elements in array and store the square of these numbers in another array and display both the array.
3) Write a C program to Copy one array into another array.

SET B:

1) Write a C program accept the polynomial and display it in format e.g. 6x^4+2x^2+5x^1+3
2) Write a 'C' program to accept n elements store those elements in array and find and replace a given number.
3) Write a 'C' program to accept two polynomials and find the addition of accepted polynomials.

SET C:

1) Write a ‘C’ program to accept two polynomials and find the Multiplication of accepted polynomials.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No 2: Sorting Techniques (Non Recursive)

SORTING

- Sorting means arranging a set of data in some given order or ordering a list of items.

  'Or'

  Sorting is a process of ordering a list of elements in either ascending or descending order.
- List is a collection of record each contains one or more fields. The field which contains unique value for each record is called key field.
- Definition:-

  Sorting is the operation of arranging the records of a table according to the key value of each record

  e.g. consider a telephone directory which consists of 4 field phone number, name, address, pin code .

  So a large data is maintained in the form of records. If we want to search a phone no and name it should be alphabetically sorted then we can search easily. It would be very difficult if records were unsorted.
- The sorting algorithm are divided into two categories

1) Internal sorting-

   Sorting is done on data which is sorted in main memory.
2) External sorting –

   Sorting is done on data which is stored on auxiliary storage device.

   e.g. hard disk, floppy, tape etc.

- **BUBBLE SORT**
  - This is one of the simplest and most popular sorting methods. The basic idea is to pass through the file sequentially several times.
  - In each pass we compare successive pairs of elements(x[i] with x[i+1]) and interchange the two if they are not in the required order.
  - One element is placed in its correct position in each pass.
  - In first pass, the largest element will sink to the bottom, second largest in the second pass and so on. Thus a total of n-1 passes are required to sort n keys
  - **Time Complexity**: Base Case: O(n), Worst Case: O(n2), Average Case: O(n2)

Algorithm for Bubble sort:

Step1: Start
Step2: Accept 'n' numbers in array 'A'
Step3: set i=0
Step4: set j=0
Step5: if j<n-i-1 then go to next stepelse go to step 8
Step 6: if i A[j+1] then interchange A[j] and A[j+1]
Step7: j=j+1 and goto step 5
Step8: i=i+1 and goto step 4
Step9:Stop

- **INSERTION SORT**
  - Insertion sort inserts each item into its proper place in the final list
  - In this the first iteration starts with comparison of 1^{st} element with 0^{th}
  - In second iteration 2^{nd} element is compared with the 0^{th} and 1^{st} element and so on.
  - In every iteration an element is compared with all elements
  - The basic idea of this method is to place an unsorted element into its correct position in a growing sorted list of data. We select one element from the unsorted data at a time and insert it into its correct position in the sorted set.
  - E.g. in order to arrange playing cards we pick one card at a time and insert this card hold in the hand.
  - **Time Complexity**: Base Case: O(n) Worst Case: O(n2) Average Case: O(n2)

Algorithm for Insertion Sort:

Step1: Start
Step2: Accept 'n' numbers and store all in array 'A'
Step3: set i=1
Step4: if i<=n-1 then goto next step else goto step 10
Step5: set Temp=A[i] and j=i-1
Step6: if Temp < A[j] && j>=0 then goto next step else goto step 9
Step7: set A[j+1]=A[j]
Step8: set j=j-1
Step9: set A[j+1]=Temp
Step10: Stop

- **SELECTION SORT**
  - It is also called pushdown sort.
  - In this the largest or smallest element is selected by placing it repeatedly till it reaches its proper position.
  - The 0^{th} element is compared with all other elements, if the 0^{th} is found to be greater than the compared element then they are interchanged . In this way after first iteration the smallest element is placed at 0^{th} position. The Procedure is repeated for 1^{st} element and so on.
  - It is Simple to implement.
  - The main advantage is that data movement is very less
  - It is not stable so. It is an in-place sort
  - **Time Complexity**: Base Case: O(n2) Worst Case: O(n2) Average Case: O(n2)

Algorithm for Selection Sort

Step1: Start
Step2: Accept 'n' numbers and store all in array 'A'
Step3: set i=0
Step4: if i <n-1 then goto next step else goto step 11
Step5: set min=i and j=i+1

Step6: if j< n then goto next step else goto step 9
Step7: if A[j]< A[min] then min=j
Step8: set j=j+1 and goto step 7
Step9: if(min not equal to i) then interchange A[i] and A[min]
Step10: i=i+1 and goto step 4
Step11: Stop

Practice Programs:

1) Write a C program to create a integer array with elements {56,23,11,67,12,89,2} and sort the given array using bubble sort.
2) Write a C program to sort a random array of n integers (value of n accepted from user) by using Bubble Sort / Insertion Sort algorithm in ascending order.
3) Write a C program to create a string array with 5 elements which contains word starting with vowel and sort them using Selection sort.

SET A:

1) Write a C program to accept and sort n elements in ascending order by using bubble sort.
2) Write a C program to accept and sort n elements in ascending order by using insertion sort.
3) Write a 'C' program to accept and sort n elements in ascending order using Selection sort method.

SET B:

1) Write a C program to create a string array with day of week and sort them using Insertion sort.
2) Write a 'C' program to accept names from the user and sort in alphabetical order using bubble sort.
3) Write a C program to accept and sort n elements in ascending order by using bubble sort and also count the number of swaps. Display the sorted list and total no of swap count.

SET C:

1) Write a C program to read the data from the file "employee.txt" which contains empno and empname and sort the data on names alphabetically (use strcmp) using Bubble Sort.
2) Write a C program to read the data from the file "person.txt" which contains personno and personage and sort the data on age in ascending order using insertion Sort / Selection Sort.
3) Modify the bubble sort, insertion sort and selection sort program of Set A to sort the integers in descending order?

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No 3: Sorting Techniques (Recursive)

QUICK SORT

- It is also called "partition Exchange sort. The strategy used here is "divide and conquer" i.e we successively partition the list in smaller lists and apply the same procedure to the sub-list. The procedure is as follows:-

Procedure –

- We will consider one element at a time (pivot) and place it in its correct position.
- The pivot is placed in a position such that all elements to the left of the pivot are less than the pivot and all elements to the right are greater.
- The array is partitioned into two parts:- left partition and right partition.
- The same method is applied for each of the partition.
- The process continues till no more partition can be made. We shall be considering the first element of the partition as the pivot element.

Algorithm:

Step 1: start.
Step 2: A is an array of n element.
Step 3: lb=0 lb = lower bound
ub = n-1 ub = upper bound.
Step 4: if(lb<ub)
i.e. if the array can be partitioned
j=partition(A,lb,ub) // j is the pivot position

quicksort(A,lb,j-1);
quicksort(A,j+1,ub);

- Now, we must write the function to partition the array. There are many methods to do the partitioning depending upon which element is chosen as the pivot.
- We will be selecting the first element as the pivot element and do the partitioning accordingly.
- We shall choose the first element of the sub- array as the pivot and find its correct position in the sub- array.
- We will be using two variables down and up for moving down and up array.

Algorithm for partitioning

Step 1: down=lb+1
Step 2: up=ub
Step 3: pivot=A[lb]
Step 4: perform step 5 to 7 as long as down<up else go to step 8.
Step 5: while (A[down]<=pivot && down<ub) down++;
Step 6: while (A[up]>pivot) up--;
Step 7: if (down<up) interchange A[down] and A[up]
Step 8: interchange A[up] and pivot, j=up, i.e. pivot position=up

Step 9: return up
Step 10: stop.

- In this algorithm we want to find the position of pivot i.e. A[lb].
- We use two pointers up and down initialized to the first and last elements respectively.
- We repeatedly increase down as long as the element is < pivot.
- We repeatedly decrease up as long as the element is > pivot.
- If up and down cross each other i.e. up<=down, the correct position of the pivot is up and A[up and pivot are interchanged.
- If up and down do not cross A[up] and A[down] are interchanged and process is repeated till they do not cross or coincide.
- Efficiency of quick sort.
  Best case = average case = O(nlogn)
  Worst case = $O(n^2)$

MERGE SORT.

- Merging is the process of combining two or more sorted data lists into a third list such that it is also sorted.
- Merge sort follows Divide and Conquer strategy.
  - Divide :- divide an n element sequence into n/2 subsequence.
  - Conquer :- sort the two sequences recursively.
  - Combine :- merge the two sorted sequence into a single sequence.
- In this two list are compared and the smallest element is stored in the third array.

Algorithm:-

Step 1: start
Step 2: initially the data is considered as a single array of n element .
Step 3: divide the array into n/2 sub-array each of length 2^i (I is 0 for 0^{th} iteration). i.e. array is divided into n sub-arrays each of 1 element.
Step 4: merge two consecutive pairs of sub-arrays such that the resulting sub-array is also sorted.
Step 5: The sub-array having no pairs is carried a sit is
Step 6: step 3 and 4 are repeated till there is only one sub-array remaining of size n.
Step 7: stop.

Practice Programs:

1) Write a C program to create a integer array with elements {888,111,666,444,222,999,333} and sort the given array using Merge sort.
2) Write a C program to sort a random array of n integers (value of n is accepted from user) by using quick Sort algorithm in ascending order.
3) Write a C program to sort a random array of n integers (value of n accepted from user) by using merge Sort algorithm in ascending order

SET A:

1) Write a C program to accept and sort n elements in ascending order by using merge sort.
2) Write a C program to accept and sort n elements in ascending order by using quick sort.

3) Modify the Quick sort program of SET A to sort the integers in descending order?

SET B:

1) Write a C program to create a string array with months (accept atleast 6 month) and sort them using Quick sort.
2) Write a C program to create a string array with atlaest 5 elements which contains word ending with ‘at’ and ‘an’ sound and sort them using Merge sort.
3) Modify the Merge sort program of Set B to sort the integers in descending order?

SET C:

1) Write a C program to read the data from the file “person.txt” which contains personno, name and personage and sort the data on age in ascending order using merge Sort.
2) Write a C program to read the data from the file “student.txt” which contains rollno, name and age and sort the data on age in ascending order using quick Sort.
3) Read the data from the file student.txt and sort on names in alphabetical order (use strcmp) using Merge sort / Quick sort. Write the sorted data to another file 'sortstudentname.txt'.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No 4: Searching Techniques

Searching is the process of finding a value in a list of values. The commonly used searching methods used are linear search and Binary search.

LINEAR SEARCH

- In Linear search, we search an element or value in a given array by traversing the array from the starting, till the desired element or value is found.
- **Time Complexity**: Base Case: O(1) Worst Case: O(n) Average Case: O(n)

Algorithm for Linear Search:

Step 1: Start
Step 2: Accept n numbers in an array num and a number to be searched
Step 3: set i=0 and flag=0
Step 4: if i< n the goto next step else goto
Step 5: Compare num[i] and number If equal then set flag=1 and goto step 7
Step 6: i=i+1 and goto step 4
Step 7: if (flag=1) then Print "Required number is found at location i+1" else Print "Require data not found"
Step 8: Stop

BINARY SEARCH

- Binary Search is used with sorted array or list. So a necessary condition for Binary search to work is that the list/array should be sorted. It works by repeatedly dividing in half the portion of the list that could contain the item.
- **Time Complexity**: Base Case: O(1) Worst Case: O(log n) Average Case: O(log n)

Algorithm for Binary search:

Step 1: Start
Step 2: Accept n numbers in an array num and a number to be searched
Step 3: set low=0, high=n-1 and flag=0
Step 4: if low <= high then middle=(low+high)/2 else goto step 7.
Step 5: if (num[middle]=number) position=middle, flag=1 goto step 7. else if (number< num[middle]) high=middle-1 else low=middle+1
Step 6: goto step 4
Step 7: if flag=1 Print "Required number is found at location position+1" Else Print "Required number is not found.
Step 8: Stop

Practice Programs:

1) Write a C program to linearly search an element in a given array. (Use Recursion).
2) Read the data from file 'employee.txt' containing names of n employees, their qualification and salary. Accept a name of the employee from the user and by using linear search algorithm check whether the name of employee is present in the file or not if present display salary of that employee, otherwise display "Employee not found".

3) Read the data from file ‘player.txt’ containing names of n Player, their game_played and age. Accept a name of the player from the user and by using binary search algorithm check whether the name of player is present in the file or not if present display game_played and age of that player, otherwise display “player not found”.

SET A:

1) Write a C program to accept n elements from user store it in an array. Accept a value from the user and use linear/Sequential search method to check whether the value is present in array or not. Display proper message.
2) Write a C program to accept n elements from user store it in an array. Accept a value from the user and use binary search method to check whether the value is present in array or not. Display proper message. (Students should accept sorted array and use Recursive function).
3) Write a ‘C’ program to create a random array of n integers. Accept a value of n from user and use Binary search algorithm to check whether the number is present in array or not. (Students should accept sorted array and use Non-Recursive function also use random function).

SET B:

1) Write a ‘C’ program to accept the names of cities and store them in array. Accept the city name from user and use linear search algorithm to check whether the city is present in array or not.
2) Write a C program to accept n elements from user store it in an array. Accept a value from the user and use recursive binary search method to check whether the value is present in array or not. Display proper message. (use any sorting method to sort the array)
3) Read the data from file ‘sortedcities.txt’ containing sorted names of n cities and their STD codes. Accept a name of the city from user and use linear search algorithm to check whether the name is present in the file and output the STD code, otherwise output “city not in the list”.

SET C:

1) Write a C program to read the data from file 'cities.txt' containing names of 10 cities and their STD codes. Accept a name of the city from user and use Binary search algorithm to check whether the name is present in the file and output the STD code, otherwise output “city not in the list”.
2) Write a C program to read the data from file 'student.txt' containing names of 10 students and their roll no. Accept a name of the student from user and use Binary search algorithm to check whether the name is present in the file and output the roll no, otherwise output “Student name not in the list”.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No 5: Linked List

- **Linked list:-**

A linked list is an ordered collection of items which is dynamic in nature i.e. its size varies and each item is 'linked" or connected to another item. It is a linear collection of data elements called nodes.

- **LINKED LIST IMPLEMENTATION-:**

A linked list may be implemented in two ways:

1) Static representation
2) Dynamic representation.

1) Static representation:-

An array is used to store the elements of the list. The elements may not be stored in a sequential order. The correct order can be stored in another array called "link"
The values in this array are pointers to elements in the disk array.

Data array

0	Blue
1	Red
2	
3	Violet
4	Green
5	
6	Orange

Link array

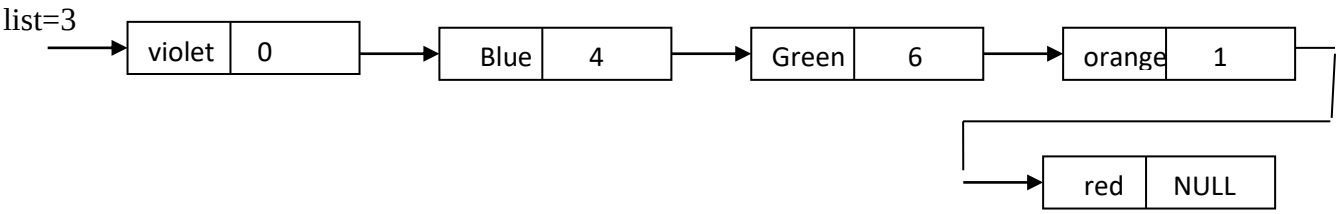
0	4
1	-1
2	
3	0
4	6
5	
6	1

Data[3] = violet
Data[0] = Blue
Data[4] = Green
Data[6] = Orange
Data[1] = Red

Link[3] = 0
Link[0] = 4
Link[4] = 6
Link[6] = 1
Link[1] = -1 list end

2) **Dynamic Representation:-**

- The static representation uses arrays which is a static data structure and has its own limitations.
- A linked list is a dynamic data structure i.e. the size should increase and when elements are deleted, its size should decrease.
- This cannot be possible using an array which uses static memory allocation i.e. memory is allocated during compile time. Hence we have to use "dynamic memory allocation" where memory can be allocated and de-allocated during run-time.
- Another way of storing a list in memory is by dynamically allocating memory for each node and linking them by means of pointers since each node will be at random memory location. We will need a pointer to store the address of the first node.



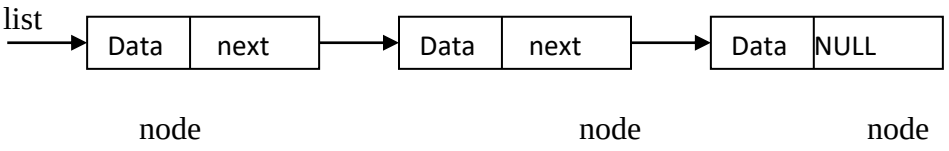
List is an external pointer which stores the address of the first node of the list.

- **TYPES OF LINKED LIST**

1) Singly Linked list
2) Circular linked list
3) Doubly linked list
4) Circular doubly linked list

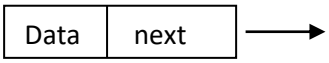
1) Linear linked list

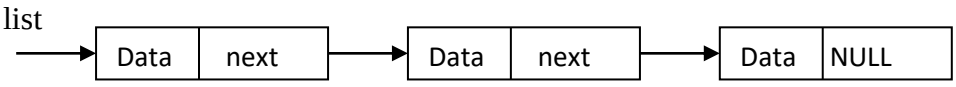
In this list the elements are organized in a linear fashion and list terminates at some point i.e. the last node contains a NULL pointer.



- **Singly linked list-**

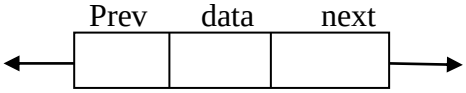
Each node in this list contains only one pointer which points to the next node.

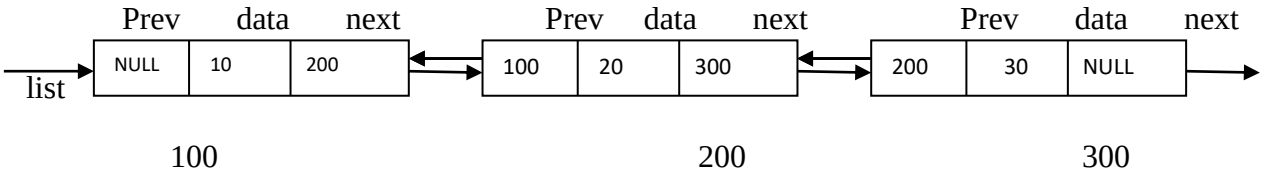




- **Doubly linked list**

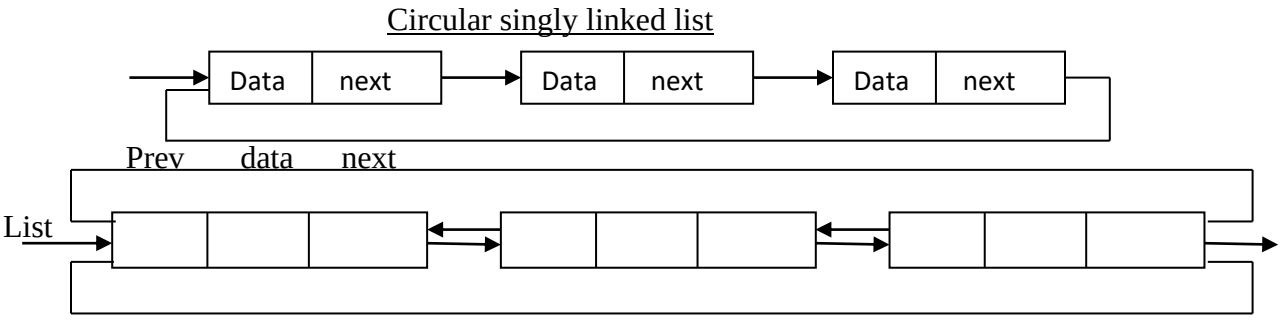
Each node in this contains two pointers, one pointing to the previous node and the other pointing to the next node. This list is used when traversing in both directions is required.





2) Circular list

In this list, the last node does not contain a NULL pointer but points back to the first node i.e. it contains the address of the first node. Each of these lists can be either a singly linked or a doubly list.



Circular doubly linked list

- **OPERATIONS ON A LIST**

The following are some of the basic list operations:-

1) Traversing a list:-
   Visiting each node of the list is called traversal
2) Insertion:-
   A node can be inserted at the beginning, end or in between two nodes of the list.
3) Deletion:-
   Deletion from a list may be done either position-wise or element-wise
4) Display:-
   Display each element of the list.
5) Searching:-
   This process searches for a specific element in the list.
6) Reversing or inversion:-
   This process reverses the order of nodes in the list
7) Concatenation:-
   This process appends the nodes of the second list at the end of the first list i.e. it joins two lists.
8) Computation of length:-
   Count the total no. of nodes in the list
9) Creating a linked list
10) Intersection, union, difference.

Practice Programs:

1) Write a C Program to find largest element of doubly linked list.
2) Write a C Program to interchange the two adjacent nodes in given circular linked list.
3) Write C Program to find length of linked list without using recursion.
4) Write C Program to print alternative nodes in linked list using recursion.

SET A:

1) Write a C program to implement a singly linked list with Create and Display operation.
2) Write a C program to implement a Circular Singly linked list with Create and Display operation.
3) Write a C program to implement a doubly linked list with Create and Display operation.
4) Write a C program to implement a Circular doubly linked list with Create and Display operation

SET B:

1) Implement the following programs by adding the functions one by one in SET A(Question1)
   i) To count total number of nodes and display the count.
   ii) To insert node at the start.
   iii) To reverse the Linked List and display both the list.
2) Write a Menu driven program in C to implement the following functions:
   i) To search the number in the list. If the number is present display the Position of node .If number not present print the message "Number not Found"
   ii) To swap mth and nth element of linked list.
   iii) To delete node from specific position of linked list.
3) Write a 'C' program to sort elements of a singly linked list in ascending order and display the sorted List.
4) Write a 'C' program to create doubly link list and display nodes having odd value.

SET C:

1) Write a C program to find intersection of two singly linked lists.
2) Write a C program to divide a singly linked list into two almost equal size lists.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No 6: Stack

A stack is an ordered collection of items into which items may be inserted and deleted from one end called the top of the stack.

The stack operates in a LIFO (last in first out) manner. i.e. the element which is put in last is the first to come out. That means it is possible to remove elements from stack in reverse order from the insertion of elements into the stack.

The real life e.g. of the stack are stack of coins, stack of dishes etc. only the topmost plate can be taken and any new plates has to be put at the top.

- **PRIMITIVE OPERATIONS ON A STACK.**

1) Create
2) Push
3) Pop
4) Isempty
5) Isfull
6) Peek

- **STACK IMPLEMENTATION:-**

The stack implementation can be done in two ways:-

1) Static implementation:-

- It can be achieved using arrays. Though it is very simple method it has few limitations.
- Once a size of an array is declared, its size cannot be modified during program execution.
- The vacant space of stack also occupies memory space.
- In both cases, if we store less argument than declared, memory is wasted and if we want to store more elements than declared array cannot be expanded. It is suitable only when we exactly know the number of elements to be stored.
- **Operations on static stack:-**

1) Declaring of a stack :-

A stack to be implemented using an array will require.

- An array of a fixed size.
- An integer called top which stored the index or position of the topmost element.

We can use a structure for the above purpose.

2) Creating a stack:-

This declaration only specifies the template. The actual stack can be declared as-

STACK s1;

3) initialize a stack:-

When a stack variable is declared the integer top has to be initialized to indicate an empty stack. Since we are using an array the first element will occupy position 0. Hence to indicate an empty stack top has to be initialized to -1

4) Checking whether stack is empty:-

An empty stack can be tested from the value contained in top. If top contains -1 it indicates an empty stack.

5) Checking whether stack is full:-

If the value of top reaches the maximum array index i.e. MAX-1 no more elements can be pushed into the stack.

6) The push operation:-

The element can be pushed into the stack only if it is not full. In such case the top has to be incremented first and the element has to be put in this position.

7) The pop operation:

An element can be removed from the stack if it is not empty. The topmost element can be removed after which top has to decremented

8) The peek operation:

It displays the topmost element of the stack without decrementing the top.

2) Dynamic implementation:-

- Pointers are used for implementation of stack. The linked list is an e.g. of this implementation.
- The limitations noticed in static implementation can be removed using dynamic implementation. The dynamic implementation is achieved using pointers.
- Using pointer implementation at runtime there is no restriction on the no. of elements. The stack may be expandable.
- The memory is efficiently utilized with pointers.
- Memory is allocated only after element is pushed to the stack
- In static representation there is a limitation on the size of the array if less elements are stored, memory will be wasted. To overcome the program the stack can be implemented using linked list.
- In the linked organization
  - The stack can grow to any size.
  - We need not have prior knowledge of the number of elements.
- When an element is popped the memory can be freed. Thus memory is not unnecessary occupied.
- Since random access to any element is not required in a stack, the linked representation is preferred over the sequential organization.

Applications of Stack

1) Inter conversion between infix, postfix and prefix expression.
2) Evaluating the postfix expression.
3) Reversing a string.
4) Reversing each word of the string and many more

There are 3 notations for specifying the operands:-

1) Infix :- if the operator symbols are placed between the operands then the expression is in the infix notation (a+b)*c
2) Postfix :- if the operator symbols are placed after its operands, then the expression is in postfix notation. ab+c*
3) Prefix :- if the operator symbols are placed before its operands, then the expression is in prefix notation. *+abc

Associativity and Precedence rule

Operator	Precedence	Associativity
{},[].()	High	L-R
^ exponent	High	R-L e.g. 3^2^2 =3^4=81
Mul/div *,/	Intermediate	L-R
+,-	Low	L-R

EVALUATE POSTFIX EXPRESSION:

Algorithme

1) Scan the string from Left to right
2) If symbol = = digit then push symbol in the stack.
3) If symbol = = operator then pop 2 elements from stack
   Put first pop element in operand2
   Put second pop element in operand1
   Evaluate the value (operand1 operator operand2) and put the evaluated answer in result
4) Push the result in the stack
5) After all the symbol are finished from symbol column pop the last element from the stack which will be the final result of the postfix expression

INFIX TO POSTFIX CONVERSION

Algorithm

1) Scan the string from left to right
2) If symbol = = opening bracket push in stack
3) If symbol = = closing bracket pop all the elements from stack till we get opening bracket, pop the opening bracket also and then put the pop elements in the postfixstring leaving opening bracket.
4) If symbol = = alphabet/ digit then put the symbol in postfixstring
5) If symbol = = operator check priority of top element in the stack.
   If priority(top element)>= priority(symbol operator) then pop top element and put it in postfixstring
   If priority(top element)< priority(symbol operator) then push the symbol in the stack
6) Repeat steps 2-5 until infix expression is scanned.
7) Print the output i.e. postfixstring

INFIX TO PREFIX CONVERSION

Algorithm

1) Scan the string from right to left
2) If symbol = = Closing bracket push in stack

3) If symbol = = opening bracket pop all the elements from stack till we get opening bracket, pop the closing bracket also and then put the pop elements in the Prefixstring leaving closing bracket.
4) If symbol = = alphabet/ digit then put the symbol in Prefixstring
5) If symbol = = operator check priority of top element in the stack.
   If priority(top element)> priority(symbol operator) then pop top element and put it in Prefixstring
   If priority(top element)<= priority(symbol operator) then push the symbol in the stack
6) Repeat steps 2-5 until infix expression is scanned.
7) Print the output i.e. print Prefixstring in reverse

STRING REVERSE AND CHECKING PALINDROME STRING:

A string of characters can be reversed by reading each character from a string starting from the first index and pushing it on a stack. Once all the characters have been read, the characters can be popped one at a time from the stack and then stored in the another string starting from the first index.

Algorithm to reverse the string:

1) Read the string character by character.
2) Push every character into the stack of characters.
3) When string becomes empty pop every character from stack and attach to the new string.

Algorithm to check palindrome of string:

1) Read the string character by character.
2) Push every character into the stack of characters.
3) When string becomes empty pop every character from stack and attach to the new string.
4) Compare original and reversed string if it matches string is palindrome else it is not palindrome

Practice Programs:

1) Let stack_ptr be a pointer to stack of integers and item be an integer variable.
   Write function like Push, Pop, Initialize, Empty, and Full for doing the following tasks. [You may declare additional variable in your functions in needed].
   a. Return the top element of stack and leave the top element unchanged. If the stack is empty, return INT_MAX.
   b. Return the third element from the top of the stack, provided that the stack contains at least three integers. If not, return INT_MAX. Leave the stack unchanged.
   c. Return the bottom element of stack (or INT_MAX if stack empty), and leave the stack unchanged.

2) Given an expression string exp, write a C program to examine whether the pairs and the orders of "{", "}", "(", ")", "[", "]" are correct in exp.

   Example:
   Input: exp = "[()]{}{[() ()]()}"
   Output: Balanced

Input: exp = "[(])"
Output: Not Balanced

3) Write a C Program to solve Tower Of Hanoi Problem (Use Recursion).
4) Write a C Program to sort a stack using temporary stack.

SET A:

1) Write a C program to implement Static implementation of stack of integers with following operation:
   -Initialize(), push(), pop(), isempty(), isfull(), display()
2) Write a C program to implement Dynamic implementation of stack of integers with following operation:
   -Initialize(), push(), pop(), isempty(), display().
3) Write a C program to reverse each word of the string by using static and dynamic implementation of stack.
   Example: Input - This is an input string
   Output – sihT si na tupni gnirts

SET B :

1) Write a 'C' program which accepts the string and check whether the string is Palindrome or not using stack. (Use Static/Dynamic implementation of Stack).
2) Write a 'C' program to read a postfix expression, evaluate it and display the result. (Use Static/Dynamic implementation of Stack).
3) Write a 'C' program to accept an infix expression, convert it into its equivalent postfix expression and display the result. (Use Static/Dynamic implementation of Stack).

SET C:

1) Write a program to check whether the contents of two stacks are identical.
2) Write a program that copies the contents of one stack into another. The order of two stacks must be identical.(Hint: Use a temporary stack to preserve the order).
3) Write a 'C' program to accept an infix expression, convert it into its equivalent prefix expression and display the result. (Use Static/Dynamic implementation of Stack).

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No 7: Queue

A queue is an ordered collection of items from which items may be deleted from (or removed from) one end called the front and into which items may be inserted at the other end called rear.

- **BASIC OPERATIONS ON QUEUE**

1) Create:

   Create a new queue. This operation creates an empty queue.

2) Add or insert:

   Add an element to the queue. A new element can be added to the queue at the rear.

3) Delete:

   Remove an element from the queue. This operation removes the elements, which is at the front of the queue. This operation can only be performed if the queue is not empty.

   The result of an illegal attempt to remove an element from an empty queue is called underflow.

4) isempty:

   Check whether a queue is empty. The operation return true if the queue isempty and false otherwise

5) isfull:

   Check whether a queue is full. The operation return true if the queue isfull and false otherwise

- **REPRESENTATION OF LINEAR QUEUES.**

There are two ways to represent a queue in memory.

1) Static (using an array)
2) Dynamic (using an linked list)

1) Static implementation of queue

   Static implementation or array representation of queue requires three entities-

   - An array to hold queue elements.
   - A variable to hold the index of the front element.
   - A variable to hold the index of the rear element.

   The implementation of a queue using sequential representation is done by using some size MAX and two integer variable front and rear. Initially front and rear is set to -1. Whenever new element is added it is added from the rear and whenever an element is to be removed from the front. The queue full condition is when rear reaches to MAX - 1. Queue empty condition is when front is equal to rear.

2) Dynamic implementation of linear queue (using an linked list)

A queue can be considered as a list in which all insertions are made at one end called the rear and all deletions from the other end from front.

A queue can be easily represented using a linked list. The front and rear will be two pointers pointing to the first and last node respectively.

Practice Programs:

1) Write a C program to Implement Static implementation of circular queue of integers which includes operation as: a) Initialize() b) insert() c) delete() d) isempty() e) isfull() f) display() g) peek()
2) Write a C program to Implement Dynamic implementation of circular queue of integers includes operation as : a)Initialize() b) insert() c)delete() d) isempty() e)display() f) peek()
3) Write a C Program to implement Deque using doubly linked list

SET A:

1) Write a C program to Implement Static implementation of Queue of integers with following operation:
   -Initialize(), insert(), delete(), isempty(), isfull(), display(), peek()

2) Write a program to reverse the elements of a queue (Use Static implementation of Queue)

SET B:

1) Write a C program to Implement Dynamic implementation of Queue of integers with following operation:
   -Initialize(), insert(), delete(), isempty(), display(), peek()

2) Write a program to reverse the elements of a queue (Use Dynamic implementation of Queue)
3) Write a C program to Implement Static implementation of circular queue of integers with following operation:
   -Initialize(), insert(), delete(), isempty(), isfull(), display(), peek()

SET C:

1) Write a c program to simulate waiting list operations of railway reservation system.
2) Implement a priority of integers using a static implementation of the queue and implementing the below two operations. Write a menu driven program
   a) Add an element with its priority into the queue.
   b) Delete an element from queue according to its priority.
3) A doubly ended queue allows additions and deletions from both the ends that is front and rear. Initially additions from the front will not be possible. To avoid this situation, the array can be treated as if it were circular. Implement a queue library (dstqueue.h) of integers using a static implementation of the circular queue and implementing the nine operations : 1)init(Q), 2) isempty(Q) 3) isFull(Q) 4)getFront(Q), 5)getRear(Q), 6)addFront(Q,x), 7)deleteFront(Q) 8) addRear(Q,x) 9)deleteRear(Q)

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No 8: Tree

Definition of tree:-

A tree is a finite set of one or more nodes such that:- there is a specially designated node called the root. The remaining nodes are partitioned into n>=0 disjoint sets T1……Tn where each of these sets is a tree. T1…..Tn are called as sub-trees of the root.

Binary Search Tree (BST) is a tree in which all the nodes follow the below-mentioned properties-

- The value of the key of the left sub-tree is less than the value of its parent (root) node's key.
- The value of the key of the right sub-tree is greater than or equal to the value of its parent (root) node's key.
- The left and right sub tree each must also be a binary search tree.

Thus, BST divides all its sub-trees into two segments; the left sub-tree and the right sub-tree and can be defined as – left_subtree (keys) < node (key) ≤ right_subtree (keys)

The operations on binary search tree are

init (T) – creates an empty Binary search tree by initializing T to NULL
insert (T, x) – inserts the value x in the proper position in the Binary search tree
search (T, x) – searches if the value x is present in the search tree
inOrder (T) – displays the node using inorder traversal of binary search tree
postOrder (T) – displays the node using postorder traversal of binary search tree
preOrder (T) – displays the node using preorder traversal of binary search tree

Practice Programs:

1) Write a C program to find all the ancestors of a given node in a binary tree.
2) Write a C program to implement binary search tree so that it handles duplicate keys properly. That is, if a key is already in the tree then the new value should replace the old rather than adding another node with the same key.
3) Write a C program to create binary search tree of integers and perform following operations using non- recursive functions
   - Preorder traversal
   - Inorder traversal
   - Postorder traversal

SET A:

1) Write C programs to implement create and display operation for binary tree.
2) Write C programs to implement create and display operation for binary search tree.
3) Write a C Program to find the product of all leaf nodes of a binary tree

SET B:

1) Write a C Program to implement the following functions on Binary Search Tree
   - To insert a new element in the tree.
   - To search an element in tree and give the proper message.
2) Write a C Program to implement the following functions on Binary Search Tree
   - To create mirror image of the tree.

- To count non-leaf nodes.

3) Write a C Program to implement the following functions on Binary Search Tree
   - To count leaf nodes.
   - To count total number of nodes.

SET C:

1) Write C programs to create and display the elements using Inorder traversal.
2) Write a C program to create binary search tree of integers and perform following operations: -
   - Preorder traversal
   - Postorder traversal

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: WellDone [ ]

Signature of Instructor

Assignment No 9: Graph

A graph consists of a set of vertices and a set of edges. The two main ways of representing graphs are adjacency matrix representation and adjacency list representation. In adjacency matrix representation of a Graph with n vertices and e edges, a two dimensional nxn array , say a , is used , with the property that a[i,j] equals 1 if there is an edge from i to j and a[i,j] equals 0 if there is no edge from i to j.
In adjacency list representation of a graph with n vertices and e edges, there are n linked lists, one list for each vertex in the graph.

The usual operations on graph are:
Indegree(i) – returns the indegree (the number of edges ending on) of the ith vertex
Outdegree(i) – returns the outdegree(the number of edges moving out) of the ith vertex
displayAdjMatrix – displays the adjacency matrix for the graph

Practice Programs:

1) Write a C Program to count the number of edges in an undirected graph.
2) Write a C Program to trace all the paths of a directed graph from the given source node to the destination node .Given the adjacency representation of a directed graph, find all the paths of the graph from source to destination.
3) Write a c program to find whether cycle in present in graph (use Directed graph)

SET A:

1) Write a C program to read a graph as adjacency matrix and display the adjacency matrix.
2) Write a C program to display total degree of each vertex.
3) Write a C program to display Indegree and outdgree degree of each vertex.

SET B:

1) Write a C program to convert adjacency matrix into adjacency list. Display the adjacency list.
2) Write a C program to traverse graph by using BFS.
3) Write a C program to traverse graph by using DFS.

SET C:

1) Implement a program to read a graph as adjacency matrix. Find the transpose of the matrix for display accepted adjacency Matrix and Adjacency Matrix and List of transpose of the matrix.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: WellDone [ ]

Signature of Instructor

Section-II

Angular JS

Assignment 1: Introduction to Angular JS, AngularJS Directives ,Expressions, Events

What is AngularJS?

- AngularJS is an Open Source efficient JavaScript framework that can create Rich Internet Applications (RIA) and provides developers an options to write client side applications using JavaScript in a clean Model View Controller (MVC) way.
- It is perfect for Single Page Applications (SPA) and the applications written in AngularJS are cross-browser compliant. AngularJS automatically handles JavaScript code suitable for each browser.
- Famous websites using AngularJS

ENVIRONMENT

- Download angular js file from : https://angularjs.org/
- Give File name: angular.min.js or any name as u wish.
- Save in your folder
- Develop angularJS first application.

First.html

```
<html>
  <head>
        <title>AngularJS First Application</title>
   </head>
   <body>     <h1>Sample Application</h1>
          <div ng-app = "">
              <p>Enter your Name: <input type = "text" ng-model = "name"></p>
              <p>Hello <span ng-bind = "name"></span>!</p>
          </div>
     <script src = "D:\satish\AngularJS\angular.min.js">
     </script>
     </body>
</html>
```

Expressions: In AngularJS, expressions are used to bind application data to HTML. AngularJS resolves the expression, and return the result exactly where the expression is written. Expressions are written inside double braces {{expression}}.They can also be written inside a directive: ng-bind="expression".

Example 1: Simple AngularJS Script to add two numbers.

```
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>
    <div ng-app>
        <p>A simple expression example: {{ 5 + 5 }}</p>
    </div>
</body>
```

```
</html>
```

Example 2: Simple AngularJS Script to initialize two variables using ng-init and calculate the result.

```
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>
        <div ng-app="" ng-init="quantity=2;cost=5">
                <p>Total in dollar: {{ quantity * cost }} </p>
          </div>
</body>
</html>
```

Example 3: AngularJS Script to concatenate to Strings.

```
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
    <div ng-app="" ng-init="person={firstName:'Satish',lastName:'Mulgi'}">
        <p>My name is {{ person.firstName +"  "+person.lastName}}</p>
    </div>
</body>
</html>
```

Example 4 : AngularJS Script to display array data using Expression.

```
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>
<div ng-app="" ng-init="points=[1,15,19,2,40]">
<p>The first result is {{ points[0] }}</p>
</div>
</body>
</html>
```

AngularJS directives are used to extend HTML. They are special attributes starting with ng- prefix. Let us discuss Some Common directives −

- ng-app − This directive starts an AngularJS Application.
- ng-init − This directive initializes application data.
- ng-model − This directive defines the model that is variable to be used in AngularJS.
- ng-repeat − This directive repeats HTML elements for each item in a collection.

ng-app directive

It defines the root element. It automatically initializes or bootstraps the application when the web page containing AngularJS Application is loaded. It is also used to load various AngularJS modules in AngularJS Application.

```
<div ng-app = "">
        ...
</div>
```

ng-init directive

The ng-init directive initializes an AngularJS Application data. It is used to assign values to the variables.

Example :

```
<div ng-app = "" ng-init = "countries = [{locale:'en-US',name:'United States'},
                                         {locale:'en-GB',name:'United Kingdom'},
                                         {locale:'en-FR',name:'France'}]">
</div>
```

ng-model directive

The ng-model directive defines the model/variable to be used in AngularJS Application. In the following example, we define a model named name.

```
<div ng-app = "">
  <p>Enter your Name: <input type = "text" ng-model = "name"></p>
</div>
```

ng-repeat directive

The ng-repeat directive repeats HTML elements for each item in a collection

Example

The following example shows the use of all the above-mentioned directives.

```
<html>
  <head>
    <title>AngularJS Directives</title>
  </head>
    <body>
    <h1>Sample Application</h1>
        <div ng-app = "" ng-init = "countries = [{locale:'en-US',name:'United States'},
                                    {locale:'en-GB',name:'United Kingdom'},
                                    {locale:'en-FR',name:'France'}]">
      <p>Enter your Name: <input type = "text" ng-model = "name"></p>
      <p>Hello <span ng-bind = "name"></span>!</p>
      <p>List of Countries with locale:</p>
    <ol>
        <li ng-repeat = "country in countries">
          {{ 'Country: ' + country.name + ', Locale: ' + country.locale }}
        </li>
      </ol>
    </div>
    <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
    </script>
   </body>
</html>
```

AngularJS Directives List

Directive	Description	Directive	Description
ng-app	It defines the root element of an application.	ng-readonly	It specifies the readonly attribute of an element.
ng-bind	It binds the content of an html element to application data.	ng-required	It specifies the required attribute of an element.
ng-blur	It specifies a behavior on blur events.	ng-selected	It specifies the selected attribute of an element.
ng-change	It specifies an expression to evaluate when content is being changed by the user.	ng-show	It shows or hides html elements.
ng-checked	It specifies if an element is checked or not.	ng-src	It specifies the src attribute for the <img> element.
ng-click	It specifies an expression to evaluate when an element is being clicked.	ng-submit	It specifies expressions to run on onsubmit events.
ng-controller	It defines the controller object for an application.	ng-switch	It specifies a condition that will be used to show/hide child elements.
ng-copy	It specifies a behavior on copy events.	ng-value	It specifies the value of an input element.
ng-cut	It specifies a behavior on cut events.	ng-disabled	It specifies if an element is disabled or not.
ng-dblclick	It specifies a behavior on double-click events.	ng-form	It specifies an html form to inherit controls from.
ng-focus	It specifies a behavior on focus events.	ng-model	It binds the value of html controls to application data.
ng-hide	It hides or shows html elements.	ng-mousedown	It specifies a behavior on mousedown events.
ng-href	It specifies a URL for the <a> element.	ng-mouseenter	It specifies a behavior on mouseenter events.
ng-if	It removes the html element if a condition is false.	ng-mouseleave	It specifies a behavior on mouseleave events.
ng-include	It includes html in an application.	ng-mousemove	It specifies a behavior on mousemove events.
ng-init	It defines initial values for an application.	ng-mouseover	It specifies a behavior on mouseover events.
ng-keydown	It specifies a behavior on keydown events.	ng-mouseup	It specifies a behavior on mouseup events.
ng-keypress	It specifies a behavior on keypress events.	ng-repeat	It defines a template for each data in a collection.
ng-keyup	It specifies a behavior on keyup events.	ng-options	It specifies <options> in a <select> list.
ng-list	It converts text into a list (array).	ng-paste	It specifies a behavior on paste events.
ng-open	It specifies the open attribute of an element.		

AngularJS has its own HTML events directives.

ng-blur, ng-change, ng-click, ng-copy , ng-cut, ng-dblclick, ng-focus, ng-keydown, ng-keypress ng-keyup, ng-mousedown, ng-mouseenter, ng-mouseleave, ng-mousemove, ng-mouseover ,ng-mouseup, ng-paste and so…on

Example 1 : ng-click Event

```
<html><head>  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"></script>
</head>
<body ng-app="myApp" >
<h1>AngularJS ng-click Demo: </h1>
  <
div ng-controller="myController">
     Enter Password: <input type="password"
ng-model="password" /> <br /><br />
<button ng-
click="DisplayMessage(password)">Show
Password</button
  </div>
  <script>
    var myApp = angular.module('myApp', []);
    myApp.controller("myController", function
($scope, $window) {
       $scope.DisplayMessage = function
(value) {
                              alert(value)
       }
    });
  </script> </body> </html>
```

Example 2 : Mouse & Button Event

```
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
<h1 ng-mousemove="count = count + 1">Mouse
Over Me!</h1>
<h2>{{ count }}</h2>

<button ng-click="myFunc()">Click
Me!</button>
<div ng-show="showMe">   <h1>Menu:</h1>
     <div>Pizza</div>   <div>Pasta</div>
</div>
<script> var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
  $scope.count = 0;
  $scope.showMe = false;
  $scope.myFunc = function() { $scope.showMe
= !$scope.showMe;   }
});
</script>
<p>Click the button to show/hide the menu.</p>
</body>
</html>
```

Practice Programs:

1) Write angular JS Program to bind product of 2 numbers.
2) Write angular JS Program to perform Arithmetic operation on two number
3) Write an angular JS Program to count movement of the cursor on the screen

SET A:

1) Write an AngularJS script to display Student name, Roll no and calculated Percentage.(use ng-init to initialize name,roll and all subject marks)
2) Write an AngularJS script to display list of games stored in an array on click of button using ng-click. And also demonstrate ng-init, ng-bing directive of Angular js.
3) Write an AngularJS script for addition of two numbers using ng-init, ng-model & ng-bind. And also Demonstrate ng-show, ng-disabled, ng-click directives on button component.

SET B :

1) Write an AngularJS script to display Product name, Quantity, Rate and Total Price.(use ng-init to initialize values)

2) Write angular JS by using ng-click Directive to display an alert message after clicking the element.
3) Using angular js display the 10 student details in Table format (using ng-repeat directive use Array to store data)

SET C:

1) Write a HTML code using Angular JS to generate the following output
   Undergraduate Courses (hint : use ng-repeat, ng-init directive)
      i. BBA(CA)
      ii. BCA(Science)
      iii. B.Sc.(Computer Science)
   Post Graduate Courses
      i. M.Sc.(Comp.Sci.)
      ii. M.Sc.(CA)
      iii. MCA
2) Write an AngularJS script to print details of bank (bank name, MICR code, IFC code, address etc.) in tabular form using ng-repeat.

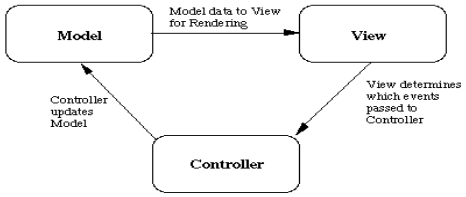
Assignment Evaluation

0: Not Done [ ]	**1: Incomplete [ ]**	**2: Late Complete [ ]**
3: Need Improvement [ ]	**4: Complete [ ]**	**5: Well Done [ ]**

Signature of Instructor

Assignment 2: AngularJS Modules, Controller, View and Scope:

MVC Architecture



- MVC – (Model View Controller) is a Software design pattern for developing web applications. It is very popular because it isolates the application logic from the user interface layer and supports separation of concerns.
- **Model**: It is responsible for managing application data. It responds to the requests from view and to the instructions from controller to update itself.
- **View**: It is responsible for displaying all data or only a portion of data to the users. It also specifies the data in a particular format triggered by the controller's decision to present the data.
- **Controller**: It is responsible to control the relation between models and views. It responds to user input and performs interactions on the data model objects. The controller receives input, validates it, and then performs business operations that modify the state of the data model.

In AngularJS controllers are used to control the flow of data of AngularJS application. A controller is defined using ng-controller directive. A controller is a JavaScript object containing attributes/properties and functions. Each controller accepts $scope as a parameter which refers to the application/module that controller is to control. The scope is the binding part between the HTML (view) and the JavaScript (controller). The scope is an object with the available properties and methods and is available for both the view and the controller.

<html> <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"> </script> <body> First Name: <input type="text" ng-model="firstName"> Last Name: <input type="text" ng-model="lastName"> Full Name: {{fullName()}} <!—Function Calling --> <script> var app = angular.module('myApp', []); app.controller('myCtrl', function($scope) { $scope.firstName = "Satish"; $scope.lastName = "Mulgi"; $scope.fullName = function() { return $scope.firstName + " " + $scope.lastName;	Here, the AngularJS application runs inside the is defined by ng-app="myApp". and the AngularJS directive is ng-controller="myCtrl" attribute. The myCtrl function is a JavaScript function. AngularJS will invoke the controller with a $scope object. The controller creates two properties (variables) in the scope (firstName and lastName). The ng-model directives bind the input fields to the controller properties (firstName and lastName). 1) View, which is the HTML. 2) Model, which is the data available for the current view. 3) Controller, which is the JavaScript function that

}; });). </script> </body> </html>	makes/changes/removes/controls the data.

Example 1: Simple AngularJS Script to Print Student Marks card

```
<html>
  <head>
      <script src =
"http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">  </script>
  </head>
  <body>
    <h2>AngularJS Sample Application</h2>
    <div ng-app = "mainApp" ng-controller =
"studentController">
      <table border = "1">
        <tr>  <td>Enter first name:</td>
          <td colspan = 2 ><input type = "text" ng-model =
"student.firstName"></td>
        </tr>
        <tr>  <td>Enter last name: </td>
            <td colspan = 2> <input type = "text" ng-
model = "student.lastName"> </td>
        </tr>
        <tr>  <td>Name: </td>  <td colspan =
2>{{student.fullName()}}</td> </tr>
        <tr>  <td rowspan = 8>Subject:</td>
              <th>Subject Name</th>.
              <th>Marks</th>
            <tr ng-repeat = "subject in student.subjects">
              <td>{{ subject.name }}</td>
              <td>{{ subject.marks }}</td>
             </tr>            </td>            </tr>
 <tr>  <td> Total </td>  <td>{{student.tot}}</td>
</table>
    </div>
```

```
  <script>
      var mainApp =
angular.module("mainApp", []);

mainApp.controller('studentController',
function($scope) {
        $scope.student = {
          firstName: "Rahul",
          lastName: "Patil",
          subjects:[
            {name:'Physics',marks:85},

{name:'Chemistry',marks:80},
            {name:'Math',marks:90},
            {name:'English',marks:80},
            {name:'Hindi',marks:70}
          ],
           tot:600,
           fullName: function() {
            var studentObject;
            studentObject =
$scope.student;
 return studentObject.firstName + " " +
studentObject.lastName;
          }
        };
      });
    </script>
</body>
</html>
```

Example : To Print Simple ETicket

```
<html>
  <head>
      <script src =
"http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
                  </script>
  </head>
  <body>
    <h2>AngularJS eTickect  Application</h2>
    <div ng-app = "mainApp" ng-controller =
"eTickectController">
      <table border = "1">
        <tr>  td>Enter name:</td>
          <td ><input type = "text" ng-model =
"name"></td>  </tr>
        <tr>  <td>Enter Address : </td><td>
            <input type = "text" ng-model = "address">
</td>  </tr>
        <tr>  <td>Enter Contact Number:</td>
          <td><input type = "text" ng-model =
"contact"></td>  </tr>
        <tr>  <td>Enter Source : </td>
          <td>  <select ng-model="source">
        <option value=" ">
        <option value="Pune">Pune
        <option value="Mumbai">Mambai
        <option value="Pimpri">Pimpri   </select> </td>
</tr>
<tr> <td>Enter Destination  : </td>
 <td> <select ng-model="destination">
            <option value=" ">
            <option value="Pune">Pune
            <option value="Mumbai">Mumbai
            <option value="Pimpri">Pimpri </select> </td>
        </tr> <tr> <td>Enter Journey Date  : </td>
         <td>  <input type = "text" ng-model = "jDate">
</td>  </tr>              <tr> <td>Enter of Pasenger Name 1   :
</td>
          <td> <input type = "text" ng-model = "p1"> </td>
</tr>
<tr>       <td>Enter of Pasenger Name 2   : </td>
 <td>     <input type = "text" ng-model = "p2">  </td>  </tr>
</table>
<table border = 2  align = center>
<tr><td colspan = 3 align =
center><font size = 5 color = red >
Indian Railway
Conter</font></td></tr>
<tr> <td>Name  :
{{name}}</td><td>Address :
{{address}}</td><td>Contact :
{{contact}}</td> </tr>
<tr>  <td>Source  :
{{source}}</td><td>Destination :
{{destination}}</td><td>Journey
Date : {{jDate}}</td> </tr> <tr>
<td> Pasengers1  :
</td><td><input type =text value
= {{p1}}></td>
<td><input type =radio name = p
>Male
<input type= radio name =
p>FeMale</td> </tr> <tr>
<td> Pasengers2 :
</td><td><input type =text value
= {{p2}} ></td>
<td><input type =radio name = p
>Male
      <input type= radio name =
p>FeMale</td> </tr> </table>
</div>
<script>
var mainApp =
angular.module("mainApp", []);
mainApp.controller('eTickectContr
oller', function($scope) {
$scope.name = "Satish"
$scope.address = "Pimpri"
$scope.contact = "9028813474"
$scope.source = "Pune"
$scope.destination = "Bidar"
$scope.jDate = "10/10/2020"
});
</script>
```

	</body> </html>

Practice Problems:

1) Using angular js create a SPA to display the student information in well formatted form(use CSS)
2) Create Simple AngularJS Script to Print Student Marks card with grade
3) Using angular js create a SPA to display the Pune Univercity information in well formatted.

SET A:

1. Using angular js create a SPA that to accept the details of Employee(5-6) having field's eno, ename, address, and salary number. Display those in table format. (use MVC.)
2. Using Angular JS Create a SPA to take the information of a customer for booking Ticket consisting of fields such as name, address, contact no., gender, Date of booking, date of journey, name of passengers etc. Display the e –Ticket.
3. Using angular js Create a SPA that show Syllabus content of all subjects of SY BBA (CA)(use ng-view)

SET B:

1) Using Angular JS Create a SPA for Bus Ticket Reservation consisting of fields : Name, Address, contact no, source station(Dropdown list), Destination station(Dropdown list), Date of booking, date of journey, name of passenger, gender of passenger etc. Display the e –Ticket.
2) Create an HTML form using Angular JS that contain the Employee Registration details and validate DOB, Joining Date, and Salary and also create a simple arithmetic calculator using radio buttons (use ng-switch, ng-switch-when)
3) Using angular js create a SPA that shows Teacher Profile who is teaching SY BBA (CA) with photo

SET C:

1) Using angular js create a SPA to display the details of product (srNo,Name, Price, quantity) available in the departmental stores.

Assignment Evaluation

0: Not Done [ ] **1: Incomplete [ ]** **2: Late Complete [ ]**
3: Need Improvement [ ] **4: Complete [ ]** **5: Well Done [ ]**

Signature of Instructor

Assignment 3: Filter, Forms Validation.

AngularJS filters are used to format data, Following is a list of filters used for transforming data.

currency: Format a number to a currency format. **date** : Format a date to a specified format. **filter** : Select a subset of items from an array. **limitTo** : Limits an array/string, into a specified number of elements/characters.	**lowercase** : Format a string to lower case. **number** : Format a number to a string. **orderBy** : Orders an array by an expression. **uppercase** : Format a string to upper case. **json**: Format an object to a JSON string.

Adding Filters to Expressions

Filters can be added to expressions by using the pipe character |, followed by a filter. case:

Example 1 : The uppercase filter format strings to upper

```
< <div ng-app="myApp" ng-
controller="personCtrl">
      <p>The name is {{ lastName |uppercase
}}</p>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('personCtrl', function($scope) {
   $scope.lastName ="Satish Mulgi";
});
</script>
```

Example 2 : The lowercase filter format strings to lower

```
<div ng-app="myApp" ng-
controller="personCtrl">
          <p>The name is {{ lastName | lowercase
}}</p>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('personCtrl', function($scope) {
   $scope.lastName ="Satish Mulgi";
});
</script>
```

Adding Filters to Directives

Filters are added to directives, like ng-repeat, by using the pipe character |, followed by a filter:

Example 3 : The orderBy filter sorts an array:

```
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/
angularjs/1.6.9/angular.min.js">
</script>
<body>
<div ng-app="myApp" ng-
controller="namesCtrl">
<p>Looping with objects:</p>
<ul>
  <li ng-repeat="x in names |
orderBy:'country'">
    {{ x.name + ', ' + x.country }}
  </li>
</ul>
</div>

<script>
angular.module('myApp',
[]).controller('namesCtrl', function($scope)
```

Example 6: Date

```
<html><script
src="https://ajax.googleapis.com/ajax/libs/angularjs
/1.4.8/angular.min.js"></script>  <body>
<div ng-app="myApp" ng-controller="datCtrl">
  <p><strong>The following date format is the by
default date format.</strong> </p>
<p>Date = {{ today | date }}</p>
  <p><strong>You can write the date in many
different formats.</strong></p>
<p>Date = {{ today | date : "dd.MM.y" }}</p>
  <p><strong>You can use predefinted formats
when displaying a date.</strong></p>
<p>Date = {{ today | date : "fullDate" }}</p>
  <p><strong>This is another format.</strong></p>
<p>Date = {{ today | date : "'today is ' MMMM d,
y" }}</p>    </div>
<script>
var app = angular.module('myApp', []);
app.controller('datCtrl', function($scope) {
   $scope.today = new Date();
});  </script>  </body>  </html>
```

```
{
   $scope.names = [
{name:'Ramesh',country:'Norway'},{name
:'Anil',country:'Sweden'},
{name:'Mangesh',country:'England'},
{name:'Suresh',country:'Norway'},
{name:'Yogesh',country:'Denmark'},
{name:'Satish',country:'Sweden'},
{name:'Bharat',country:'Denmark'},
{name:'Alli',country:'England'},{name:'Ka
i',country:'Norway'} ];
});
</script></body></html>
```

Example 4: The currency filter formats a number to a currency format

```
<div ng-app="myApp" ng-controller="costCtrl">
<h1>Price: {{ price | currency }}</h1>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('costCtrl', function($scope) {
   $scope.price = 58;
});
</script>
```

Example 5 : Filter

```
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/
angularjs/1.6.9/angular.min.js"></script>
<body ng-app="">
<div ng-init="friends = [{name:'Ram',
phone:'555-1276'},
                {name:'Kumar',
phone:'8454545334'},
                {name:'Rahul',
phone:'9120232322'},
                {name:'Kiran', phone:'555-
56434'},
                {name:'Anil', phone:'555-
833434'},
                {name:'Suresh',
phone:'555-5455'}]"></div>
<label>Search: <input ng-
model="searchText"></label>
<table id="searchTextResults">

<tr><th>Name</th><th>Phone</th></tr>
  <tr ng-repeat="friend in friends |
filter:searchText">
    <td>{{friend.name}}</td>
    <td>{{friend.phone}}</td>
  </tr>
</table>
<hr>
```

```
<<label>Any: <input ng-
model="search.$"></label> <br>
<label>Name only <input ng-
model="search.name"></label><br>
<label>Phone only <input ng-
model="search.phone"></label><br>
<label>Equality <input type="checkbox" ng-
model="strict"></label><br>
<table id="searchObjResults">
  <tr><th>Name</th><th>Phone</th></tr>
  <tr ng-repeat="friendObj in friends |
filter:search:strict">
    <td>{{friendObj.name}}</td>
    <td>{{friendObj.phone}}</td>
  </tr>
</table>
```

Form Validation

AngularJS offers client-side form validation. AngularJS monitors the state of the form and input fields (input, textarea, select), and lets you notify the user about the current state. AngularJS also holds information about whether they have been touched, or modified, or not.we can use standard HTML5 attributes to validate input, or you can make your own validation functions

Form State and Input State

AngularJS is constantly updating the state of both the form and the input fields.

Input fields have the following states:

- $untouched :The field has not been touched yet
- $touched : The field has been touched
- $pristine : The field has not been modified yet
- $dirty : The field has been modified
- $invalid : The field content is not valid
- $valid : The field content is valid

They are all properties of the input field, and are either true or false.

Forms have the following states:

- $pristine : No fields have been modified yet
- $dirty : One or more have been modified
- $invalid : The form content is not valid
- $valid : The form content is valid
- $submitted :The form is submitted

They are all properties of the form, and are either true or false.

You can use these states to show meaningful messages to the user. Example, if a field is required, and the user leaves it blank, you should give the user a warning:

Required

```
<html> <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body ng-app="">
<p>Try leaving the first input field blank:</p>
<form name="myForm">
<p>Name:
<input name="myName" ng-model="myName" required>
<span ng-show="myForm.myName.$touched && myForm.myName.$invalid">The name is
required.</span>
</p> <p>Address:
<input name="myAddress" ng-model="myAddress" required>
</p> </form>
<p>We use the ng-show directive to only show the error message if the field has been touched
AND is empty.</p></body> </html>
```

Form Validation

```
<html>  <head>  <script src = "http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>  </head
<body>  <h2>AngularJS Sample Application</h2>
  <div ng-app = "mainApp" ng-controller = "studentController">
<form name = "studentForm" novalidate>
<table border = "0">
<tr><td>Enter first name:</td>
<td><input name = "firstname" type = "text" ng-model = "firstName" required>
<span style = "color:red" ng-show = "studentForm.firstname.$dirty && studentForm.firstname.$invalid">
<span ng-show = "studentForm.firstname.$error.required">
      First Name is required.</span>
</span></td></tr>
<tr><td>Enter last name: </td>
<td><input name = "lastname"  type = "text" ng-model = "lastName" required>
<span style = "color:red" ng-show = "studentForm.lastname.$dirty && studentForm.lastname.$invalid">
<span ng-show = "studentForm.lastname.$error.required">
        Last Name is required.</span>
</span></td> </tr>
<tr> <td>Email: </td><td><input name = "email" type = "email" ng-model = "email" length = "100" required>
<span style = "color:red" ng-show = "studentForm.email.$dirty && studentForm.email.$invalid">
<span ng-show = "studentForm.email.$error.required">Email is required.</span>
<span ng-show = "studentForm.email.$error.email">Invalid email address.</span>
</span></td></tr>
<tr><td> <button ng-click = "reset()">Reset</button></td>
<td> <button ng-disabled = "studentForm.firstname.$dirty && studentForm.firstname.$invalid || studentForm.lastname.$dirty && studentForm.lastname.$invalid || studentForm.email.$dirty && studentForm.email.$invalid" ng-click="submit()">Submit</button>
</td></tr></table></form></div>
<script> var mainApp = angular.module("mainApp", []);
mainApp.controller('studentController', function($scope) {
$scope.reset = function(){
$scope.firstName = "Suresh";
$scope.lastName = "Jadhav";
$scope.email = "Sureshjadhav@dypvp.edu.in"; }
$scope.reset(); });
</script>
</body>
</html>
```

Practice Problems:

1) Create student registration form and validate all fields
2) Store 10 students information in an array and display students ingormation intable form orderBy Name (use ordeBy filter sorts an array)

SET A:

1) Using angular js display the student details who are live in pune in Table format (using ng-repeat directive, use Array to store data, use filter)
2) Write an AngularJS Script to search a product with its rate (use ng-repeat directive, use Array to store data, use filter currency)

3) Write an AngularJS script to search student name according to the character typed and display details (use array and filter).

SET B:

1) Using angular js display the Employee details order by salary in Table format (using ng-repeat directive, use Array to store data, use filter)
2) Using angular js create a SPA that to accept the details such as name, mobile number, pin-code and email address and make validation. Name should contain character only, mobile number should contain only 10 digit, Pin code should contain only 6 digit, email id should contain only one @, . Symbol
3) Using AngularJS create a SPA for Login System.

SET C:

1) Using angular js create a SPA for eLearning System.

Assignment Evaluation

0: Not Done [ ]	**1: Incomplete [ ]**	**2: Late Complete [ ]**
3: Need Improvement [ ]	**4: Complete [ ]**	**5: Well Done [ ]**

Signature of Instructor

Assignment 4: AngularJS Services

In AngularJS, a service is a function, or object, that is available for, and limited to, your AngularJS application. In AngularJS you can make your own service, or use one of the many built-in services. AngularJS has about 30 built-in services.

$location service: The $location service has methods which return information about the location of the current web page:

```
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
<p>The url of this page is:</p>
<h3>{{myUrl}}</h3>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $location) {
   $scope.myUrl = $location.absUrl();
});
</script></body></html>
```

The $timeout Service

The $timeout service is AngularJS' version of the window.setTimeout function.

```
<html><script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body><div ng-app="myApp" ng-controller="myCtrl">
<p>This header will change after two seconds:</p>
<h1>{{myHeader}}</h1>
</div>
<p>The $timeout service runs a function after a specified number of milliseconds.</p>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $timeout) {
  $scope.myHeader = "Hello World!";
  $timeout(function () {
     $scope.myHeader = "How are you today?";
  }, 2000);
});</script></body></html>
```

The $interval Service

The $interval service is AngularJS' version of the window.setInterval function.

```
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
<p>The time is:</p>
<h1>{{theTime}}</h1>
</div>
<p>The $interval service runs a function every specified millisecond.</p>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $interval) {
  $scope.theTime = new Date().toLocaleTimeString();
  $interval(function () {
      $scope.theTime = new Date().toLocaleTimeString();
  }, 1000);
});
</script></body></html>
```

Create Your Own Service

To create your own service, connect your service to the module:

```
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
<p>The hexadecimal value of 255 is:</p>
<h1>{{hex}}</h1>
</div>
<p>A custom service with a method that converts a given number into a hexadecimal number.</p>
<script>
var app = angular.module('myApp', []);
app.service('hexafy', function() {
    this.myFunc = function (x) {
        return x.toString(16);
    }
});
app.controller('myCtrl', function($scope, hexafy) {
  $scope.hex = hexafy.myFunc(257);
});
</script></body></html>
```

The AngularJS $http service makes a request to the server, and returns a response.

```
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
<p>Today's welcome message is:</p>
<h1>{{myWelcome}}</h1>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $http)
{
  $http.get("/welcome.htm").then(function (response) {
      $scope.myWelcome = response.data;
  });
});</script></body></html>
```

Practice Problems:

1) Create angular JS Application that shows location of the current web page.
2) Create angular JS Application that demonstrate $timeout Service

SET A:

1) Create angular JS Application that show the current Data and Time of the System(Use Interval Service)

2) Create a angular JS Application that greet the User (Use $timeout Service)
3) Using angular js create a SPA to carry out validation for a username entered in textbox. If the textbox is blank, alert 'Enter username'. If the number of characters is less than three, alert ' Username is too short'. If value entered is appropriate the print 'Valid username' and password should be minimum 8 characters

SET B

1) Create an HTML form using Angular JS that contain the Student Registration details and validate Student first and last name as it should not contain other than alphabets and age should be between 18 to 50 and display greeting message depending on current time using ng-show (e.g. Good Morning, Good Afternoon, etc.)(Use AJAX).
2) Using angular js create a SPA that to accept the details such as name, mobile number, pin-code and email address and make validation. Name should contain character only, mobile number should contain only 10 digit, Pin code should contain only 6 digit, email id should contain only one @, . Symbol.
3) Using angular js create a SPA that accept Voters details and check proper validation for (name, age, and nationality) as Name should be in upper case letters, Age should not be less than 18 yrs and Nationality should be Indian.

SET C:

1) Using angular js create a SPA to fetch suggestions when is user is typing in a textbox. (eg like google suggestions. Hint create array of suggestions and matching string will be displayed).

Assignment Evaluation

0: Not Done [ ]	**1: Incomplete [ ]**	**2: Late Complete [ ]**
3: Need Improvement [ ]	**4: Complete [ ]**	**5: Well Done [ ]**

Signature of Instructor

Section-III

PHP

Assignment 1:Basics in PHP

Basics of PHP

For learning PHP, we have to learn the following basic points:

PHP Delimiters:

PHP is an embedded application, for writing PHP code we have to use its delimiters

<? php => starting delimiter and

?> => ending delimiter.

Syntax:

```
<? php

?>
```

Data Types:

PHP supports the following data types:

- String
- Integer
- Float (floating-point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

Operators in PHP

Types of operators that can be used in PHP programs are

There are the following types of operators:

- Arithmetic operators:- + , - , * , / , % ,**
- Assignment operators :- =
- Comparison operators :- <, >, <=, ==, ===, !=, <>, !==
- Increment/Decrement operators :- ++,--
- Logical operators: - &&, ||, !
- String operators:- .(concatenation)
- Conditional assignment operators: - ?,:

***Note: S**tudents can design HTML form to accept input from the user as per the requiremet of the program.

Practice Programs:

1. Write a PHP script to perform arithmetic operations on two numbers (Addition, Subtraction, Multiplication Division).
2. Write a PHP script to display a maximum of two numbers using a conditional operator.
3. Write a PHP script that will perform pre and post-increment of a number. (Example ++a, a++).

SET A:

1. Write a PHP Script to display Quotient and Remainder of the division of two variables.
2. Write a PHP Script to swap the values of two variables.
3. Write a PHP Script which will convert temperatures from Celsius(C)to Fahrenheit (F). (Hint: C=5.0/9(F-32)

SET B:

1. Write a PHP Script to display the surface area and volume of a cuboid.
   (Hint: surface area=2(lb+lh+bh), volume = l*b*h)
2. Write a PHP Script to calculate the area of Circle, Square, and Rectangle.
3. Write a PHP Script to display the total and percentage of Marks of Subjects (Out of 100) Data Structure, Digital Marketing, PHP, SE, and Bigdata.

SET C:

1. Write a PHP Script to calculate the total cost of AIR Ticket Reservation and display the details for Name, Address, Contact No, Source, Destination, Date of journey, Gender of passenger, No of Persons, Price per Ticket, etc.

Assignment Evaluation

0: Not Done [ ] **1: Incomplete [ ]** **2: Late Complete [ ]**
3: Need Improvement [ ] **4: Complete [ ]** **5: Well Done [ ]**

Signature of Instructor

Assignment 2:Control Structures and Loops

Conditional Statements

Conditional statements are used to check conditions and the programmer can accordingly display the results. PHP supports the following conditional statements.

1. if Statement
2. if else Statement
3. elseif Statement
4. switch Statement

Name/Use	Syntax	Example
if Statement: It is used to check a condition. If the condition is true the corresponding body of if statement is executed.	if(Condition) { Statements; }	<?php $n=10; if ($n%2==0) { echo “Number Is Even”; } ?> **Output:**Number Is Even
if else Statement: It checks a condition and if the condition is true the corresponding body of the if statement is executed otherwise else part is executed.	if(Condition) { Statements; } else { Statements; }	<?php $n=10; if ($n%2==0) { echo “Number Is Even”;} else { echo “Number Is Odd”;} ?> **Output:**Number Is Even
elseif Statement: It checks more than one condition.	if (condition) { Statements;} elseif (condition) { Statements;} else { Statements;}	<?php $a=10; $b=20; if ($a==$b) { echo “a and b are same”;} elseif ($a<$b) { echo “a is less than b”;} else { echo “a is greater than b”;} ?> **Output:**a is less than b
switch Statement It checks the result of the expression with multiple conditions (cases). The case is	switch (expression) { case value1: Statements	<?php $num=2; switch($num) {

executed for which the match is found.	break; case value2: Statements break;..... default: Code to be executed if all cases are not matched; }	Case 1: echo “One”; break; Case 2: echo “two”; break; Case 3: echo “Three” break; Case 4: echo “Four; break; Case 5: echo “Five”; break; default: echo “Invalid Number”; ?> **Output:** Two

Loops

Loops in php are used to execute a similar group of statements. PHP supports the following 4 types of loops.

1. for Loop
2. while Loop
3. do…while Loop
4. foreach Loop

Name/Use	**Syntax**	**Example**
for Loop It executes a block of code for a specified number of times.	for (initialization; condition; increment) { code to be executed; }	<?php For($j=1; $j<=5; $j++) { echo $j; } ?> **Output:** 12345
while Loop It executes a block of code until the condition specified is true.	while(expression) { Statements; }	<?php $j=1; while($j<=5) { echo $j; $j++; } ?>

		Output: 12345
do…while Loop It executes ablock of code once and then repeats the loop as long as a special condition is true.	do { code to be executed; } while (condition);	<?php $j=1; do{ echo $j; $j++; } while($j<=5); ?> **Output:** 12345
foreach Loop It is used to traverse the array and the block of code is executed for each element of the array.	foreach (array as value) { code to be executed; }	<?php $array = array(1, 2, 3, 4, 5); foreach($array as $value) { echo "Value is $value"; } ?> **Output:** Value is 1Value is 2Value is 3Value is 4Value is 5

***Note: S**tudents can design HTML form to accept input from the user as per the requirement of the program.

Practice Programs:

1. Write a PHP Script to display a maximum of two numbers.
2. Write a PHP Script to check whether a number is positive or negative.
3. Write a PHP Script to display a Multiplication table of a number

SET A:

1. Write a PHP Script to check whether a year is a leap or not.
2. Write a PHP Script which will perform the Addition, Subtraction, Multiplication, and Division of two numbers as per the choice. (Use Switch Case)
3. Write a PHP Script to display the grade of the student according to percentage. Use the following conditions:
   Percentage <40 => Grade=”Fail”
   Percentage >= 40 and Percentage <=50 => Grade= “Pass Class”
   Percentage >=50 and Percentage <=60 => Grade= “Higher Second Class”
   Percentage >60 and Percentage <=70 => Grade= “First Class”
   Percentage >70 => Grade= “First Class with Distinction”

SET B:

1. Write a PHP Script to display prime numbers between 1 to 50.

2. Write a PHP Script to display a perfect numbers between 1 to100.
3. Write a PHP Script to display the reverse of a number. E.g. 607 =>706
4. Write a PHP Script to display Armstrong numbers between 1 to 500.

SET C:

1. Write a PHP script to display a number in words (Use Switch case)
   e.g. 345–three four five
2. Write a PHP script to change the background color of the browser using a switch statement according to the day of the week.
3. Write a PHP script to count the total number of even and odd numbers between 1 to 1000.

Assignment Evaluation

0: Not Done [ ] **1: Incomplete [ ]** **2: Late Complete [ ]**
3: Need Improvement [ ] **4: Complete [ ]** **5: Well Done [ ]**

Signature of Instructor

Assignment 3: Arrays and Strings

Strings in PHP

A string is a sequence of characters. There are two types of strings.

1. **Single-Quoted String:**In this type, characters are enclosed with a single quotation mark(‘’);

Examples:

‘Hello World’
‘Amar’
‘Pune’

The limitation of a single-quoted string is that variables are not interpolated.

Example:

```
<?php
$name=‘Amar’;
$str='Hello $name';
echo $str;
?>
```

Output:

Hello $name

2. **Double-Quoted String:** In this type,characters are enclosed with double quotation marks (“ ”).

PHP interpreter interprets variables and special characters inside double-quotes.

Example:

```
<?php
    $name=‘Amar’;
    $str=“Hello $name”;
    echo $str;
?>
```

Output:

Hello Amar

It expands the many PHP escape sequences. The escape sequences recognized by PHP in double-quoted strings are as follows:

Escape Sequence	Meaning
\n	New Line
\r	carriage return
\t	horizontal tab
\v	vertical tab
\e	escape
\f	form feed
\\	Backslash
\$	dollar sign
\"	double-quote

String Functions

PHP provides approximately one hundred functions for string manipulations. Some of the functions that can be performed on strings are:

- Compare two strings
- Find a String In AnotherString
- Find Out How Many Instances of A String Occur In AnotherString
- Return Part of aString
- Replace Part of aString
- Trim Whitespace From The Ends of aString
- Make An Entire String Lowercase or uppercase

Name	Use	Example
strlen()	It is used get string length.	<?php $input = 'Sunayana'; echo strlen($input); ?> **Output:**8
trim()	It used to remove the whitespaces and other characters.	<?php $input = " Programming in PHP \n"; echo trim($input); ?> **Output:** Programming in PHP
ltrim()	It used to strip whitespace or other characters from the beginning of a string.	<?php $input = " Programming in PHP \n"; echo trim($input); ?> **Output:** Programming in PHP \n
rtrim()	It is used to remove the	<?php

	white spaces from end of the string.	$input = " Programming in PHP \n"; echo trim($input); ?> **Output:** Programming in PHP
strtolower()	It converts the whole string into lower case.	<?php echo strtolower("DYPATIL ACS"); ?> **Output:** dypatil acs
strtoupper()	It converts the whole string into upper case.	<?php echo strtoupper("d y patil acs "); ?> **Output:** D Y PATIL ACS
ucfirst()	It used to convert the first character of a string to upper case.	<?php echo ucfirst("dypatil"); ?> **Output:**Dypatil
ucwords()	It used to convert the first character of a string to upper case in each string	<?php echo ucwords("d y patil pimpri"); ?> **Output:** D Y Patil Pimpri
strcmp()	It is used to compare two strings. If two string are equal it returns 0 otherwise 1.	<?php echo "The result is "; echostrcmp("Hello world!","Hello world!"); ?> Output: The result is 0
substr()	Returns a part of a string	<?php echo substr("D Y Patil",2); ?> **Output:** Y Patil
substr_replace()	It used to replace the part of string with another string	<?php echosubstr_replace("HelloWorld","Good Morning",0); ?> **Output:** Good Morning
substr_compare()	It used to compare two string format with a specific start position	<?php echo substr_compare("Hello","world",0)." "; echo substr_compare("abcde","de",1,3)." "; ?> **Output:** -1 -1
substr_count()	It used to count the number of sub strings	<?php echo substr_count("HelloWorld","World"); ?> **Output:** 1
strrev()	It is used to reverse a	<?php

	string.	echo strrev("sairamkrishna"); ?> **Output:**anhsirkmarias
str_pad()	It pads a string to a new length.	<?php //Pad to the right side of the string, to a new length of 20 characters: $str = "Hello World"; echo str_pad($str,15,"="); ?> **Output:** Hello World====
explode()	It is used to split a string by string	<?php //Decomposing string //Break a string into an array: $str = "Hello world. It's a beautiful day."; $arr=explode(" ",$str); print_r ($arr); echo" "; $str = "one|two|three|four"; $arr=explode('\|',$str); print_r ($arr); ?> **Output:** Array ([0] => Hello [1] => world. [2] =>It's [3] => a [4] => beautiful [5] => day.) Array ([0] => one [1] => two [2] => three [3] => four)
implode()	It creates a string from an array of smaller string.	<?php $arr = array('Hello','World!','Beautiful','Day!'); echo implode(" ",$arr); echo " "; $arr = array('Hello','World!','Beautiful','Day!'); echo implode(",",$arr); ?> **Output:** HelloWorld! Beautiful Day! Hello,World!,Beautiful,Day!
strpos()	It is used to find the position of first occurrence of a string inside another string.	<?php echostrpos("I love php, I love php too!","php")." "; ?> **Output:** 7

strstr()	It is used to find the first occurrence of a string and returns from that small string onwards.	<?php echostrstr("Hello world!","world")." "; echostrstr("w3resource.com","."); ?> **Output:** world! .com

Arrays in PHP

An array is a collection of different data elements. Multiple elements can be stored using an array under a single name.

Declaration of an Array

An array can be defined/declared by using **array ()** function.

Syntax:

```
$a=array (10, 20, 30, 40);
$colors = array("Red", "Blue, "Yellow");
```

There are following types of an array:

1. Indexed array.
2. Associative array.
3. Multidimensional array.

An array is organized as an ordered collection of (key,value) pairs. In PHP there are three types of arrays:

a) Indexed array: It is an array with a numeric index starting with 0. There are two ways to create an Indexed array:

Example:

```
$num=array (10, 20);
        OR
$num[0]=10;
$num[1]=20;
```

b) Associative array: Associative arrays are arrays that use named keys that you assign to them. There are two ways to create an associative array:

Example:

```
$age = array("Sagar"=>"35", "Abhijeet"=>"37", "Ishwar"=>"43");
                    OR
```

```
$age['Sagar'] = "35";
$age['Abhijeet'] = "37";
$age['Ishwar'] = "43";
```

c)Multidimensional array: A multidimensional array is an array containing one or more arrays. In this type of array, multiple arrays can be defined in a single array.

Example:

```
$cars = array(
  array(“Swift”,20,30),
  array(“Dezire,40,50),
  array(“Mercedez”,6,7),
  array("Scoda",12,15)
  );
```

Array Functions

Name	Use	Example
array_chunk()	It is used to split an array into chunks of a given size	<?php $a=array("10","20","30","40"); print_r(array_chunk($a,2); ?> **Output:** Array([0] => Array([0] =>10 [1] =>20) [1] => Array ([0] =>30 [1] =>40))
array_combine ()	It is used to combine two arrays into one, values of the first array are the keys and values of the second array are the values in the combined array.	<? php $X=array("a","b","c"); $Y=array("100","200","300"); $Z=array_combine($X,$Y); print_r($Z); ?> **Output :**Array([a]=>100, [b]=>200, [c]=>300)
array_diff ():	It is used to compare the values of two arrays and return the difference	<?php echo "<font size=14>"; $a=array(1,2,3,4,5); $b=array(4,2,6); $c=array_diff($a,$b); print_r($c); ?> **Output:** Array ([0] => 1 [2] => 3 [4] => 5)
array_intersect()	It returns the common elements of two arrays.	<?php $a=array(1,2,3,4); $b=array(4,5,6,2); $c=array_intersect($a,$b);

		print_r($c); ?> **Output:** Array([1]=>2, [2]=>4);
array_flip()	Exchanges all keys with their associated values in an array.	<?php $a = array("a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5); print_r(array_flip($a)); ?> **Output:** Array ([1] => a [2] => b [3] => c [4] => d [5] => e)
array_splice()	It removes and replaces specified elements of an array	<?php $a=array(10,20,30,40,50,60); $b=array_splice($a,2,3); print_r($a); print_r($b); ?> **Output:**Array ([0] => 10 [1] => 20 [2] => 60) Array ([0] => 30 [1] => 40 [2] => 50)
array_slice()	It returns selected parts of an array. It returns the sequence of elements from the array array as specified by the offset and length para meters.	<?php $a=array(1,2,3,4,5,6); $b=array_slice($a,2,3); print_r($a); echo " "; print_r($b); ?> **Output:** Array ([0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5 [5] => 6) Array ([0] => 3 [1] => 4 [2] => 5)
array_reverse()	Returns an array in the reverse order.	<?php $a=array(1,2,3); $d=array_reverse($a); print_r($d) ?> **Output:** Array ([0] => 3 [1] => 2 [2] => 1)
array_key_exists( )	This function is used to check if an element exists in thearray	<?php $a=array("a"=>"ABC","p"=>"PQR","x"=>"XYZ"); if(array_key_exists("p",$a)) echo "Key Exists"; else echo "Key Does not Exists"; ?> **Output:** Key Exists
array_push()	This function add the	<?php

	new element at the end of an array.	$a = array("a"=>"banana","b"=>"apple","c"=>"orange"); print_r(array_push($a, "Straberry")); print_r($input); ?> **Output:** 4 Array ([a] => banana [b] => apple [c] => orange [0] =>Straberry)
array_pop()	This functions remove last element of an array.	<?php $a=array("a"=>"banana","b"=>"apple","c"=>"orange"); print_r(array_pop($a)); ?> **Output:** orange
array_shift()	It removes the first element from an array, and returns the value of the removed element	<?php $a = array("a"=>"banana","b"=>"apple","c"=>"Mango"); print_r(array_shift($a)); ?> **Output:** banana
array_unshift()	It adds one or more elements to the beginning of an array	<?php $a = array("orange", "banana"); array_unshift($a, "apple"); print_r($a); ?> **Output:** Array ([0] => apple [1] => orange [2] => banana)
array_sum()	It returns the addition of array elements.	<?php $a=array(1,2,3); $sum=array_sum($a); echo "Sum=$sum"; ?> **Output:** Sum=6
array_product()	It returns the product of array elements.	<?php $a = array(5,6); print_r(array_product($a)); ?> **Output:**30
array_unique()	It removes duplicate values from an array	<?php $a = array("a" => "green", "red", "b" => "green", "blue", "red"); $result = array_unique($a); print_r($result); ?>

		Output: Array ([a] => green [0] => red [1] => blue)
extract()	It creates local variables from an array.	**?**php $a = "Original"; $my_array = array("a" => "Cat","b" => "Dog", "c" => "Horse"); extract($my_array); echo "\$a = $a; \$b = $b; \$c = $c"; ?> **Output:** $a = Cat; $b = Dog; $c = Horse
compact()	Create array containing variables and their values	<?php $city = "Pune"; $state = "Mumbai"; $result = compact("city", "state"); print_r($result); ?> **Output:**Array ([city] => Pune [state] => Mumbai)
in_array()	Checks if a specified value exists in an array	<?php $a=array(10,20,30,40,50,60); if(in_array(40,$a)) echo "Element Available in array"; else echo "Element not available in array"; ?> **Output:** Element Available in array
count()	It gives number of elements in an array.	<?php $a=array(10,20,30,40,50,60); echocount($a); ?> **Output:** 6

Array Sorting Functions

Name	Use	Syntax
sort()	It sorts array in ascending order.	<?php $a=array("mh","ap","LM","za"); print_r($a); echo " "; sort($a); echo " After Sorting "; print_r($a); ?>

		Output: Array ([0] =>mh [1] =>ap [2] => LM [3] =>za) After Sorting Array ([0] => LM [1] =>ap [2] =>mh [3] =>za)
rsort()	It sorts array in descending order.	<?php $a=array("mh","ap","LM","za"); print_r($a); echo " "; rsort($a); echo " After Sorting "; print_r($a); ?> **Output:** Array ([0] =>mh [1] =>ap [2] => LM [3] =>za) After Sorting Array ([0] =>za [1] =>mh [2] =>ap [3] => LM)
asort()	It sorts associative array in ascending order as per the values	<?php $b= array("X"=>"XYZ","A"=>"ABC","L"=>"LMN"); print_r($b); asort($b); echo " After Sorting "; print_r($b); ?> **Output:** Array ([X] => XYZ [A] => ABC [L] => LMN) After Sorting Array ([A] => ABC [L] => LMN [X] => XYZ)
arsort()	It sorts associative array in descending order as per the values.	<?php $b= array("X"=>"XYZ","A"=>"ABC","L"=>"LMN"); print_r($b); arsort($b); echo " After Sorting "; print_r($b); ?> **Output:** Array ([X] => XYZ [A] => ABC [L] => LMN) After Sorting Array ([A] => ABC [L] => LMN [X] => XYZ)
ksort()	It sorts associative array in ascending order as per the keys.	<?php $b= array("X"=>"XYZ","A"=>"ABC","L"=>"LMN"); print_r($b); ksort($b);

		echo " After Sorting "; print_r($b); ?> **Output:** Array ([X] => XYZ [A] => ABC [L] => LMN) After Sorting Array ([A] => ABC [L] => LMN [X] => XYZ)
krsort()	It sorts associative array in descending order as per the keys.	<?php $b= array("X"=>"XYZ","A"=>"ABC","L"=>"LMN"); print_r($b); krsort($b); echo " After Sorting "; print_r($b); ?> **Output:** Array ([X] => XYZ [A] => ABC [L] => LMN) After Sorting Array ([X] => XYZ [L] => LMN [A] => ABC)

***Note: S**tudents can design HTML form to accept input from the user as per the requirement of the program.

Practice Programs:

1. Write a PHP Script to define an array. Find the element from the array that matches the given value using the appropriate search function.
2. Write a PHP script to count the total number of vowels (a,e, i,o,u) from the string. Show the occurrences of each vowel from the string.
3. Write PHP program to perform the following operations on Indexed Array:
   a) Check the array element is positive or negative
   b) Calculate the average of array elements
   c) Calculate the sum of array elements

SET A:

1. Write PHP program to perform the following operations on Indexed Array:
   a) Union of two arrays
   b) Traverse the array elements in random order
2. Write a PHP program to perform the following operations on an associative array:
   a) Display the elements of an array along with the keys.
   b) Display the size of an array
   c) Delete an element from an array from the given index.
   d) Reverse the order of each element's key-value pair

e) Traverse the elements in an array in random order.

3. Write a PHP Script for the following:
   a) Declare and Display a multidimensional Array.
   b) Search and display a specific element from a Multidimensional array.

SET B:

1. Write a PHP script to perform the following operations on string :
   i) Compare string 2 with string3.
   ii) Convert all the strings to Upper case
   iii) Convert all the strings to Lowercase
2. Write a PHP script to perform the following operations on string :
   i) Convert each word of a string to Lowercase and Uppercase.
   ii) Find the first and last occurrence of string2 in string1.
3. Write a menu-driven program in PHP to perform the following operations on associative arrays:
   i) Sort the array by values (changing the keys) in ascending, descending order.
   ii) Also, sort the array by values without changing the keys.
   iii) Find the intersection of two arrays.
   iv) Find the union of two arrays.

SET C:

1. Write a PHP script to perform the following operations on string :
   i) Replace the string2 by string3 in string1.
   ii) Reverse and display the string.

Assignment Evaluation

0: Not Done [ ] **1: Incomplete [ ]** **2: Late Complete [ ]**
3: Need Improvement [ ] **4: Complete [ ]** **5: Well Done [ ]**

Signature of Instructor

Assignment 4:Functions, Class, and Object

Functions

A function is a block of code that performs a specific task. It can be called from anywhere from the program. It takes zero or any number of parameters and does some processing and returns a value.

PHP Built-in Functions

Name	Use	Example
echo	This construct is used to display many values at once on the screen.	echo “Hello”; echo (“Hello”);
print()	It prints data to the screen	print (“Hello”);
print_r()	It prints the contents of arrays and objects.	<?php $array = array(1, 2, 3); print_r($array); ?> **Output:** Array ([0] => 1 [1] => 2 [2] => 3)
var_dump()	It returns the value and data type of a given variable.	<?php $a=50; echo var_dump($a); ?> **Output:**Int(50);
isset()	It returns true value, if a given parameter is initialized with a value otherwise it returns false value.	<?php $n = 0; if (isset($n)) { echo "Variable 'a' is set.”; } **Output:**Variable 'a' is set.
unset()	It unsets a variable.	<?php $a = 10; echo "The value of variable 'a' before unset: " . $a . " "; unset($a); echo "The value of variable 'a' after unset: " . $a; ?> **Output:** The value of variable 'a' before unset:10

		The value of variable 'a' after unset:
define()	It is used to define constant.	<?php define (“PI”,3.14); echo PI ?> **Output**:3.14.
date()	The date() function formats a local date and time, and returns the formatted date string. Syntax: date(format,timestamp) List of characters commonly used for date: d - Represents the day of the month m - Represents a month Y - Represents a year l - Represents the day of the week	<?php echo "Today is " . date("Y/m/d") . " "; echo "Today is " . date("Y.m.d") . " "; echo "Today is " . date("l"); ?> **Output:** Today is 2021/01/19 Today is 2021.01.19 Today is Tuesday

Defining a user-defined function

While creating a user-defined function its name should be preceded by with keyword **function** and the function code should be put inside { and } braces.

Syntax:

```
functionfunction_name([parameters])
{
Statements;
}
```

Example:

```
<?php
        /* Defining a PHP Function */
    functionHelloWorld()
    {
    echo "HelloWorld Good Morning!!";
    }
        /* Calling a PHP Function */
    HelloWorld();
?>
```

Output: HelloWorld Good Morning!!

Passing Parameters to Functions

1. Call By Value

When a PHP function is called by value then actual values of variables are not modified if it is modified into the function.

Example:

```
<?php
        functionaddFun($num1, $num2)
        {
                $sum = $num1 + $num2;
        echo "Sum of the two numbers is : $sum";
        }
        addFun(15, 20);
 ?>
```

2. Call By Reference

When a PHP function is called by reference then the actual values of the parameters are modified by the function.

Example:

```
<?php
        functionaddFun(&$num1, &$num2)
        {
                $sum = $num1 + $num2;
        echo "Sum of the two numbers is : $sum";
        }
        addFun(15, 20);
 ?>
```

Default Parameter

If we do not pass any value to the function then the function uses a default value calleddefault parameter.

Example:

```
<?php
        functionsetHeight($maxheight=100)
        {
        echo "The height is : $maxheight<br>";
        }
        setHeight(350);
        setHeight(); // will use the default value of 100
?>
```

Output:

```
The height is: 350
The height is:100
```

Classes and Objects:

PHP supports to the object oriented programming concepts.

Class:

It is user defined data type. It is a collection of data members and functionsas a single unit.

A class can be defined as:

Example:

```
<?php
   class Car
   {
        /* Member variables */
   var $price;
        /* Member functions */
   functionsetPrice($par)
   {
         $this->price = $par;
   }
   functiongetPrice()
   {
   echo $this->price ."<br/>";
   }
  }
?>
```

Object:

Any real or runtime entity is called an object. Objects are also known as instance.

Creating Objects in PHP

After defining a class, an object of that class can be created. It can be done by using a new operator as follow:

Example:

```
Object_name=new Class_Name;
$Car1=new Car;
```

For accessing data member and member functions of a class, an object is used.

Example:

```
$Car1->setPrice(5);
```

Practice Programs:

1. Write a PHP script to calculate the area and volume of a cylinder using a function.
2. Write a PHP Script to display the sum and average of array elements(Using predefined functions)
3. Write a PHP script to calculate the factorial of a number using a function.

SET A:

1. Write a PHP script to calculate x^y using a function.
2. Write a PHP script to define a function EvenOdd, which will display even and odd numbers between 1 to 50.
3. Write a PHP script to define a function Maximum, which will accept 3 numbers as parameters and returns a maximum of 3 numbers.
4. Write a PHP script to swap two numbers using a function (Use Call by value and Call by reference)

SET B:

1. Write a PHP Script to create a class Fruit that contains data members as Name, Color and Price. Write a member function to accept and display details of Fruit.
2. Write a PHP Script to create a class Student that contains data members as Roll_Number, Stud_Name, and Percentage. Write member functions to accept Student information.
3. Write a PHP Script to create a class Book (Book_id, Book_name, Publication, Author, Book_price). Write a member function to accept and display Book details.

SET C:

1. Write a PHP script to define a function "DisplayDay", which will display the day of the current date.
2. Write a PHP script to perform arithmetic operations on two numbers. Write a PHP function to display the result. (Use the concept of function and default parameters)

Assignment Evaluation

0: Not Done [ ] **1: Incomplete [ ]** **2: Late Complete [ ]**
3: Need Improvement [ ] **4: Complete [ ]** **5: Well Done [ ]**

Signature of Instructor

Assignment 5: Working with forms

Processing a Form's Data:

HTML forms are used to send the user information to the server and returns the result to the browser. For example, if you want to get the details of visitors to your website, and send them good thoughts, you can collect the user information employing form processing. Then, the information can be validated either at the client-side or on the server-side. The final result is sent to the client through the respective web browser. To create a HTML form, the following **form** tag should be used.

- Action
- Method

Form processing contains a set of controls through which the client and server can communicate and share information. The controls used in forms are:

- Text
- Textarea
- Dropdown
- Radio
- Checkbox
- Buttons

PHP methods used in form processing are:

- **$_GET[]:** It is used the retrieve the information from the form control through the parameters sent in the URL. It takes the attribute given in the URL as the parameter.
- **$_POST[]:** It is used the retrieve the information from the form control through the HTTP POST method. It takes the name attribute of the corresponding form control as the parameter.

Example Using POST Method

stud.html

```
<html>
<body>
<form method=POST action="stud.php">
        RollNo :          <input type=text name=rno><br>
        Student Name : <input type=text name=sname><br>
        Percentage :     <input type=text name=per><br>
                        <input type=submit value=submit name=submit>
</form>
</body>
</html>
```

stud.php

```
<?php
```

```
        echo $_POST[‘rno’];
        echo $_POST[‘sname’];
        echo $_POST[‘per’];
?>
```

Example Using GET Method

stud.html

```
<html>
<body>
<form method=GET action=”stud.php”>
        RollNo :          <input type=text name=rno><br>
        Student Name : <input type=text name=sname><br>
        Percentage :     <input type=text name=per><br>
                         <input type=submit value=submit>
</form>
</body>
</html>
```

stud.php

```
<?php
        echo $_GET[‘rno’];
        echo $_GET[‘sname’];
        echo $_GET[‘per’];
?>
```

PHP function used for form processing:

- **isset():** This function is used to determine whether the variable or a form control is having a value or not.

Example using isset()

stud.html

```
<html>
<body>
<form method=POST action=”stud.php”>
        RollNo :          <input type=text name=rno><br>
        Student Name : <input type=text name=sname><br>
        Percentage :     <input type=text name=per><br>
                         <input type=submit value=submit>
</form>
</body>
</html>
```

stud.php

```
<?php

if (isset($_GET[‘submit’]))
{
        if((!isset($_GET[‘rno’])) ||(!isset($_GET[‘sname’]))|| (!isset($_GET[‘per’])))
        {
                Echo “Please fill all the required fields”;
        }
}
else
{
        echo $_GET[‘rno’];
        echo $_GET[‘sname’];
        echo $_GET[‘per’];
}
?>
```

Self ProcessingPage :

Selfprocessing page means one PHP can be used to both generate a form and process it. PHP_SELF variable is used for self processing page.

PHP_SELF variable returns the name and path of the currently executing script. This variable can used in action attribute of the form.

Example:

```
<form method="Get" action="<?php $_SERVER['PHP_SELF'];?>">
```

Example Self processing page

stud.php

```
<html>
<body>
<form method=GET action="<?php $_SERVER['PHP_SELF'];?>">
        RollNo :          <input type=text name=rno><br>
        Student Name : <input type=text name=sname><br>
        Percentage :     <input type=text name=per><br>
                        <input type=submit value=submit>
</form>

<?php

if (isset($_GET[‘submit’]))
{
        if((!isset($_GET[‘rno’])) ||(!isset($_GET[‘sname’]))|| (!isset($_GET[‘per’])))
        {
                Echo “Please fill all the required fields”;
        }
}
else
```

```
{
        echo $_GET[‘rno’];
        echo $_GET[‘sname’];
        echo $_GET[‘per’];
}
?>
</body>
</html>
```

Sticky Forms:

Form remembers the values that are entered in the input fields. For example Google search box. In the sticky form, the results of a query are accompanied by a search form whose default values are those of the previous query.

To create the sticky form, we have to follow 2 steps:

- Step 1: Taking the data sent by the form by using the “GET” or “POST” method: $data= $_GET[“data”];
- Step 2: Settings that data as a value for text fields and selected or checked for other form elements.

Example Sticky Forms

stud.php

```
<html>
<body>
<form method=GET action="<?php $_SERVER['PHP_SELF'];?>">
        Your Name : <input type=text name=sname value=”<?phpecho$_POST[‘sname’] ?>”>
<br>
                <input type=submit value=submit>
</form>

<?php
        if(isset($_GET[‘sname’]))
        {
                echo $_GET[‘sname’];
        }
?>
</body>
</html>
```

Dealing with Checkbox

```
<html>
<body>
<form method=GET action="<?php echo $_SERVER['PHP_SELF']; ?>">

Select ur Choice<br>
```

```
<input type=checkbox name=ch[] value="1" <?php if($_GET['ch']=="1") echo 'checked="checked"'; ?>>
Reading

<input type=checkbox name=ch[] value="2" <?php if($_GET['ch']=="2") echo 'checked="checked"'; ?>>
Dancing<br>

<input type=Submit name="S" value=Click>
</form>

<?php
	if ((isset($_GET['S'])))
	{
		$ch=$_GET['ch'];
		if($ch=="1")
			echo "Reading";
		else
			echo "Dancing";
	}
?>

</body>
</html>
```

Dealing with Radio button

```
<html>
<body>
<form method=GET action="<?php echo $_SERVER['PHP_SELF']; ?>">
Select ur Choice<br>
<input type=radio name="r" value="1" <?php if($_GET['r']=="1") echo 'checked="checked"'; ?>> Add

<input type=radio name="r" value="2" <?php if($_GET['r']=="2") echo 'checked="checked"'; ?>> Sub<br>

<input type=Submit name="S" value=Click>
</form>

<?php
	if ((isset($_GET['S'])))
	{
		$ch=$_GET['r'];
		switch($ch)
		{
		case 1:
```

```
                        $a=$t1+$t2;
                        echo "Addition=$a";
                        break;
                case 2:
                        $a=$t1-$t2;
                        echo "Sub=$a";
                        break;
                }
        }
?>

</body>
</html>
```

Retrieving values from List:

```
<html>
<body>
<form method=GET action="<?php echo $_SERVER['PHP_SELF']; ?>">
Select ur Choice<br>
<select name=m>
        <option value="R" <?php if($_GET['m']=="R") echo 'selected="selected"'; ?>>Reading
        <option value="D" <?php if($_GET['m']=="D") echo 'selected="selected"'; ?>>Dancing
</select>
<input type=submit name=S value=Click>
</form>

<?php
        if ((isset($_GET['S'])))
        {
                $ch=$_GET['m'];
                echo $ch;

        }
?>

</body>
</html>
```

Validating and Restricting data:

Different strategies for validating form data are,

- Fields should not be empty
- Check the length of the data entered by the user.
- Check the type of data entered by the user.
- Check specific conditions for form fields

Functions for Validating and Restricting data are,

- **Empty(varName):** it is used to check whether a variable is empty or not.

- **isset():** This function is used to determine whether the variable or a form control is having a value or not.
- **filter_var()** function filters a variable with the specified filter.
  Syntax:

  filter_var(var, filtername, options)

  Parameters: This function accepts three parameters and is described below:
  1. **var** : It is the required field. It denotes the variable to filter.
  2. **filtername** : It is used to specify the ID or name of the filter to use. Default is FILTER_DEFAULT, which results in no filtering. It is an optional field.
  3. **options** : It is used to specify one or more flags/options to use. Check each filter for possible options and flags. It is also an optional field.

  Return Value: It returns the filtered data on success, or FALSE on failure.
  Filternames are,
  - FILTER_VALIDATE_INT: to check if the variable is an integer or not
  - FILTER_VALIDATE_IP: to check if the variable is a valid IP address or not.
  - FILTER_VALIDATE_EMAIL: to check if the variable is a valid email address or not.
  - FILTER_VALIDATE_URL: to check if the variable is a valid URL or not.

Example using empty()

```
<html>
<body>
<form method=GET action="<?php echo $_SERVER['PHP_SELF']; ?>">
Search<input type=text name="t1" value="<?php if(isset($_GET['t1'])) echo $_GET['t1']; ?>">
        <input type=Submit name="s" value=Click>
</form>

<?php
        if ((isset($_GET['t1'])))
        {
                $t1=$_GET['t1'];
                if ((!empty($t1)))
                {
                        echo $t1;
                }
                else
                {
                        echo "enter the value in textbox";
                }
        }
?>
</body>
</html>
```

Example using filter_var :

FILTER_VALIDATE_INT

```
<?php
$n = 200;
if (filter_var($n, FILTER_VALIDATE_INT) === 0 ||
   !filter_var($int, FILTER_VALIDATE_INT) === false)
{
   echo("Integer is valid");
}
else
{
   echo("Integer is not valid");
}
?>
```

FILTER_VALIDATE_IP:

```
<?php

$ip = "129.0.0.1";
  if (!filter_var($ip, FILTER_VALIDATE_IP) === false)
{
   echo("$ip is a valid IP address");
}
else
{
   echo("$ip is not a valid IP address");
}
 ?>
```

FILTER_VALIDATE_EMAIL:

```
<?php
$email = "abc@gmail.com";
if (!filter_var($email, FILTER_VALIDATE_EMAIL) === false)
{
   echo("$email is a valid email address");
}
else
{
   echo("$email is not a valid email address");
}
?>
```

FILTER_VALIDATE_URL:

```
<?php
  $url= "https://www.google.com";
if(!filter_var($url, FILTER_VALIDATE_URL) === false)
```

```
{
   echo("$url is a valid URL");
}
else
{
   echo("$url is not a valid URL");
}
?>
```

Practice Programs:

1. To design an application that works as a simple calculator using PHP. (use isset()).
2. Write a PHP script to check PAN number entered by the customer is valid or not and display an appropriate message.
3. Write a PHP script to check mobile number entered by the user is valid or not and display an appropriate message.

SET A:

1. Write a PHP script to accept font name, background color, and welcome message on 1st page. Display the welcome message with the given font and background color on the next page.
2. Write a PHP program to accept name, address, pincode, gender information. If any field is blank display error messages “all fields are required”.
3. Write a PHP script to accept employee details (name, address) and earning details (basic, DA, HRA). Display employee details and earning details in the proper format.

SET B:

1. Write a PHP script to accept customer name and the list of product and quantity on the first page. On the next page display the name of the customer, name of the products, rate of the product, quantity, and total price in table format.
2. Write HTML code to design multiple choice question paper for PHP subject. Display question wise marks and total marks received by the student in table format.
3. Write a PHP script to accept student name and list of programming languages (using drop down box) and display it on the next page in the proper format.
4. Write a PHP script to accept user name, email address and age. If data entered by the user is valid then display it on the next page otherwise display the appropriate message(use filter_var()).

SET C:

1. A web application that takes name and age from an HTML page. If the age is less than 18, it should send a page with "Hello <name>, you are not authorized to visit the site" message, where <name> should be replaced with the entered name. Otherwise, it should send a "Welcome <name> to this site" message.

Assignment Evaluation

0: Not Done [ ]	**1: Incomplete [ ]**	**2: Late Complete [ ]**
3: Need Improvement [ ]	**4: Complete [ ]**	**5: Well Done [ ]**

Signature of Instructor

Assignment 6: Session and Cookies

Cookies:

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

A cookie is created with the setcookie() function.

setcookie(name[, value, expire, path, domain, secure, httponly]);

where,

- **name:** A unique name for a particular cookie. You can have multiple cookies with different
- names and attributes.
- **value :** It is used to set the value of the cookie
- **expire** : The expiration date for this cookie. If no expiration date is specified, the browser
- saves the cookie in memory and not on disk. When the browser exits, the cookie disappears.
- The expiration date is specified as the number of seconds.
- **path :**It is used to specify the path on the server for which the cookie will be available.
- **domain** :It is used to specify the domain for which the cookie is available.
- **secure** : It is used to indicate that the cookie should be sent only if a secure HTTPS connection exists.

```
<?php
        //Creating a cookie
        $cookie_name = "user";
        $cookie_value = "abc";
        setcookie($cookie_name, $cookie_value, time() + (1* 24 * 60 * 60));

        //Checking a Cookie is set or not
        if(!isset($_COOKIE[$cookie_name]))
        {
        echo "Cookie named '" . $cookie_name . "' is not set!";
        }
        else
        {
        echo "Cookie '" . $cookie_name . "' is set!<br>";
        }

        //Accessing Cookie value
        echo "Value is: " . $_COOKIE[$cookie_name];

        // set the expiration date to one hour ago
        setcookie("user", "", time() - 3600);
        echo "Cookie 'user' is deleted.";

        //cookie expire after 1 day
        setcookie($cookie_name, $cookie_value, time() + (1* 24 * 60 * 60));
?>
```

Session:

A session is a way to store information (in variables) to be used across multiple pages. The information is not stored on the user's computer.By default, session variables last until the user closes the browser.Session variables hold information about one single user and are available to all pages in one application

- The first step is to start up a session. After a session is started, session variables can be created to store information. The PHP **session_start()** function is used to begin a new session. It also creates a new session ID for the user.
- **The second step is to set Session variables using** PHP global variable: **$_SESSION.**

```
<?php
        // Start the session
        session_start();

        // Set session variables
        $_SESSION["favcolor"] = "green";
        $_SESSION["favanimal"] = "cat";
        echo "Session variables are set.<br>";

        // access session data
        echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
        echo "Favorite animal is " . $_SESSION["favanimal"] . "<br>";
        print_r($_SESSION);

        // to change a session variable, just overwrite it
        $_SESSION["favcolor"] = "yellow";
        print_r($_SESSION);

        // remove all session variables
        session_unset();
        if(($_SESSION["favcolor"]!=0) && ($_SESSION["favanimal"]!=0))
                print_r($_SESSION);
        else
                echo "Session variables are unset.<br>";

        // destroy the session
        session_destroy();
        print_r($_SESSION);
        echo "Session variables are destroyed.<br>";

?>
```

Practice Programs:

1. A web application that lists all cookies stored in the browser on clicking "list cookies" button,add cookies if necessary.

2. Write a PHP program to store the current date-time in a COOKIE and display the 'Last visited on'date-time on the web page upon reopening of the same page.
3. Write a script to keep track of a number of times the web page has been accessed using the session.

SET A:

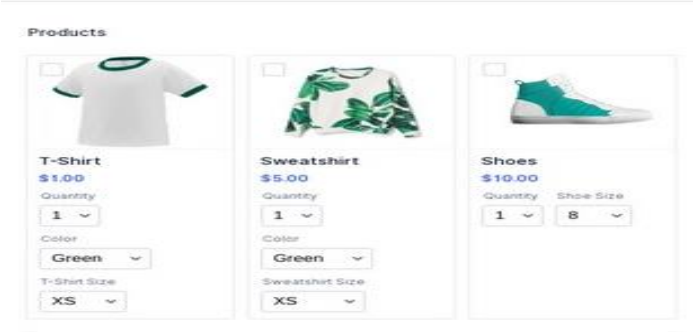
1. Write PHP program to store student information like Seat number, name, and class. On the second page, accept marks of the subject PHP, DS, CPP, and RDBMS. Display Result in table format on the third page (use cookies).
2. Write a PHP script to accept username and password. If in the first three chances, username and password entered is correct, then display the welcome message on the second form, otherwise display an error message.
3. Write a PHP script to accept font style, font size, font color, background color using a cookie. Display selected values on the next second page and actual implementation on the third web page.

SET B:

1. Create an online flight registration form. On the first page accept name, address, birthdate, and mobile number. On the second page accept flight details (flight name, source, destination, departure date-time and charges). If the user doesn't enter information within a specified time limit, expire his session and give a warning otherwise display details using sessions on the third page**.**
2. Create a form to accept patient details like name, address birthdate, and mobile number. Once thePatient information is accepted, and then accepts health details like medicare number, health fund and critical information. Display patient details and health details on the next form.
3. Write a PHP script to create an inventory management system. On the first page accept the highest sold product details like product name, total quantity and total sold. On the second page accept the latest sales details like product name, date and total sale. Display highest sold product details in one table and latest sales details in another table on the third page.

SET C:

1. Write a PHP script to create an online shopping form. On the first page accept customer name, email address, shipping address, mode of payment. Design the Second page as given below. And the third page should display a bill, which consists of customer details and purchase details in the proper format.



Assignment Evaluation

0: Not Done [ ] **1: Incomplete [ ]** **2: Late Complete [ ]**
3: Need Improvement [ ] **4: Complete [ ]** **5: Well Done [ ]**

Signature of Instructor

Assignment 7: Database

Database :

It is a collection of inter-related data that helps in efficient retrieval, insertion, and deletion of data from the database and organizes the data in the form of tables, views, schemas, reports, etc. The basic functions used in PHP for database connection are,

Function	Description	Example
mysql_connect(server, user, pwd)	It opens a database connection and returns the connection on success, or FALSE and an error on failure.	$con=mysql_connect("localhost","root","");
mysql_select_db(db name)	It is used to change the default database for the connection.	mysql_select_db("sybba");
mysql_query(query);	It executes a query on a MySQL database.This function returns the query handle for SELECT queries, TRUE/FALSE for other queries, or FALSE on failure.	$sql=”Select * from stud” $r=mysql_query($sql);
mysql_fetch_array(result,resulttype);	function fetches a result row as an associative array, a numeric array, or both. result=Required. resulttype=Optional. Specifies what type of array that should be produced. Values are, MYSQL_ASSOC, MYSQL_NUM, MYSQL_BOTH	mysql_fetch_array($r, MYSQL_ASSOC);
mysql_close(connection);	The mysql_close() function closes MySQL connection.This function returns TRUE on success, or FALSE on failure.	mysql_close($con);

1. The basic steps to create a MySQL database using PHP are:
   - Establish a connection to the MySQL server from your PHP script.
   - If the connection is successful, write a SQL query to create a database and store it in a string variable.
   - Execute the query.
   - Close the connection

```
<?php
        $con=mysql_connect("localhost","root","");
```

```
        if(!$con)
        {
        die("unable to connect");
    }
    $sql="create database sybba";
$r=mysql_query($sql);
if(! $r)
    {
        die("could not create database");
    }
echo "Database created successfully";
mysql_close($con);
?>
```

2. The basic steps to create a MySQL table using PHP are:
    - Establish a connection to the MySQL server from your PHP script.
    - If the connection is successful, then select the database.
    - Write a SQL query to create a table and store it in a string variable.
    - Execute the query.
    - Close the connection

```
<?php
        $con=mysql_connect("localhost","root","");
        if(!$con)
        {
        die("unable to connect");
    }
mysql_select_db("sybba");

    $sql="create table stud(rnoint, snamevarchar(20), per int)";
$r=mysql_query($sql);
if(! $r)
    {
        die("could not create table");
    }
    echo "Table created successfully";
mysql_close($con);
```

3. The basic steps to manipulate MySQL table using PHP are:
    - Establish a connection to the MySQL server from your PHP script.
    - If the connection is successful, then select the database.
    - Write aninsert/update/delete query to manipulate and store it in a string variable
    - Execute the query.

- Close the connection

```
<?php
        $con=mysql_connect("localhost","root","");
        if(!$con)
        {
        die("unable to connect");
      }
mysql_select_db("sybba");

      $sql="insert into stud values(1,'Neeta',84)";
$r=mysql_query($sql);
if(! $r)
      {
          die("not inserted");
      }
      echo "record added successfully";
mysql_close($con);
?>
```

4. The basic steps to fetchdata from MySQL table using PHP are:
    - Establish a connection to the MySQL server from your PHP script.
    - If the connection is successful, then select the database.
    - Write a select query and store it in a string variable
    - Execute the query
    - Display data using a while loop.
    - Close the connection

```
<?php
        $con=mysql_connect("localhost","root","");
        if(!$con)
        {
        die("unable to connect");
        }
        mysql_select_db("sybba");
        $result=mysql_query("select * from stud");
while($col=mysql_fetch_array($result,MYSQL_NUM))
        {
                echo "Rollno=".$col[0]."<br>";
                echo "Name=".$col[1]."<br>";
                echo "Per =".$col[2]."<br>";
        }
        mysql_close($con);
?>
```

Practice Programs:

1. Consider the following entities and their relationships
   Company (c_no, c_name, c_city, c_share_value)
   Person (p_no, p_name, p_city, p_ph_no)
   Relationship between Company and Person is many-to-many with descriptive attribute no_of_shares.
   Using the above database, write a PHP script to display person wise share detailsin tabular format.

2. Consider the following entities and their relationships
   Customer (c_no, c_name, c_city, c_ph_no)
   Ticket (t_no, booking_date, fare, traveling_date)

The relationship between Customer and Ticket is one-to-many. Create a RDB in 3 NF for the above.
Using the above database, write a PHP script to accept date and display,
1) The total fare collected from customers on a given date.
2) Ticket details booked by the customer.

SET A:

1. Write a PHP script to create an employee table using attributes employee number, employee name, address joining date and salary. If a table is created then display the appropriate message otherwise end the PHP script.
2. Write a PHP script to accept account details (account number, account type and balance). Store these details in the account table and display an appropriate message.
3. Write a PHP script to accept product number from the user. Update the price of the product and display an appropriate message.

SETB:

1. Consider the following entities and their relationships.
   Employee (eno, ename, sal)
   Project (pno, pname,duration)
   Employee and Project are related with a many-many relationship. Create a RDB in 3 NF for the above.
   Using the above database write a PHP script to accept the project name. Display the name of the employees and the duration of the project.
2. Consider the following entities and their relationships.
   Train(t_no, t_name)
   Passenger (p_no, p_name, addr, age)
   The relationship between Train and Passenger is many-to-many with descriptive attribute date,seat_no and amt. Create a RDB in 3 NF for the above.
   Using the above database write a PHP script to accept a date. Display train details having maximum passenger for a given date.
3. Consider the following entities and their relationships.
   Crop (c_no, c_name, c_season, pesticides)

Farmer (f_no, f_name, f_location)

The relationship between Crop and Farmer is many-to-many with descriptive attribute year.

Create a RDB in 3 NF for the above.

Using the above database write a PHP script to accept crop name and year value. Display total number of farmers harvesting given crop in a given year.

SETC:

1. Consider the following entities and their relationships.
   Client (c_no, c_name, c_addr, birth_date)
   Policy_info (p_no, p_name, maturity_amt, prem_amt, policy_term)
   The relationship between Client and Policy_info is many-to-many with descriptive attribute date_of_purchase.Create a RDB in 3NF for the above.
   Using the above database write a PHP script to display policy details of a given client for a given year in the following format.

Client Name : **Year:**

Policy Name	Maturity Amount	Premium Amount	Policy Term	Date of Purchase

Assignment Evaluation

0: Not Done [ ] **1: Incomplete [ ]** **2: Late Complete [ ]**
3: Need Improvement [ ] **4: Complete [ ]** **5: Well Done [ ]**

Signature of Instructor

Section-IV

Big Data

Assignment 1: Basic R Programming

Introduction:

R is a programming language and software environment for statistical analysis, graphics representation and reporting. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team.

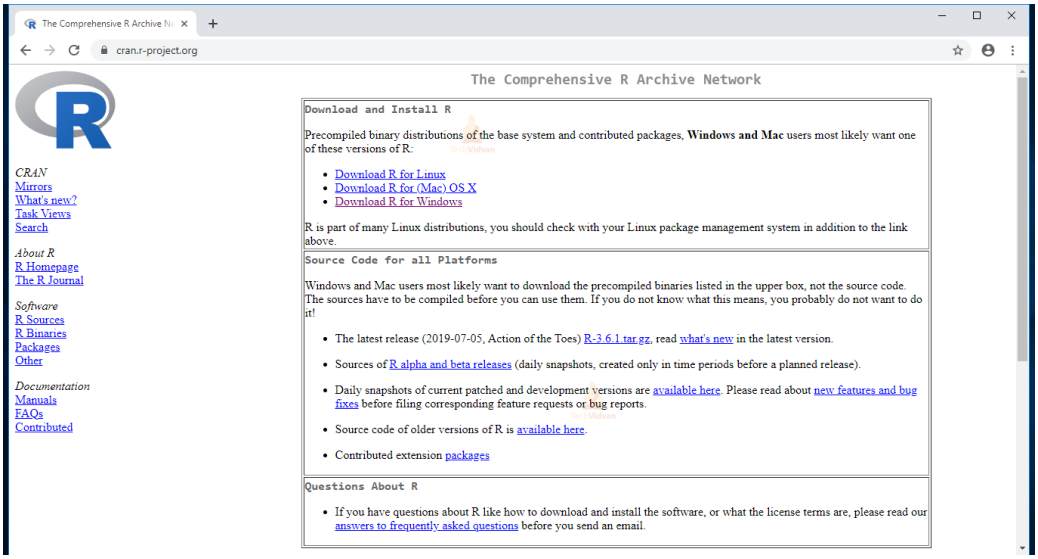
The core of R is an interpreted computer language which allows branching and looping as well as modular programming using functions. R allows integration with the procedures written in the C, C++, .Net, Python or FORTRAN languages for efficiency.

Installing R and RStudio:

To install R and RStudio on windows, go through the following steps:

Install R on windows

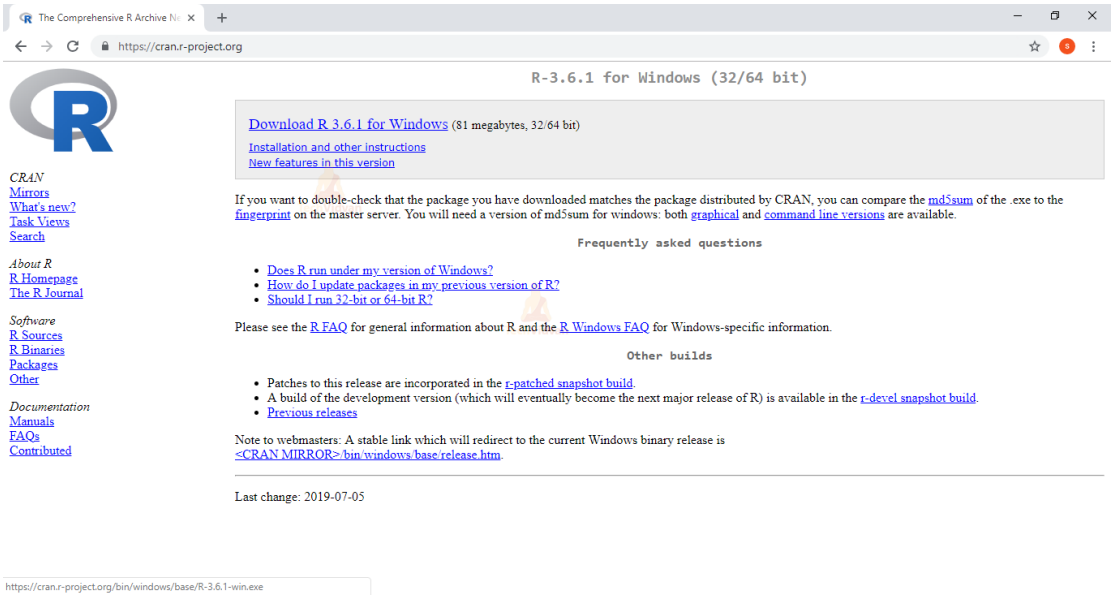
Step – 1: Go to CRAN R project website.



Step – 2: Click on the **Download R for Windows** link.

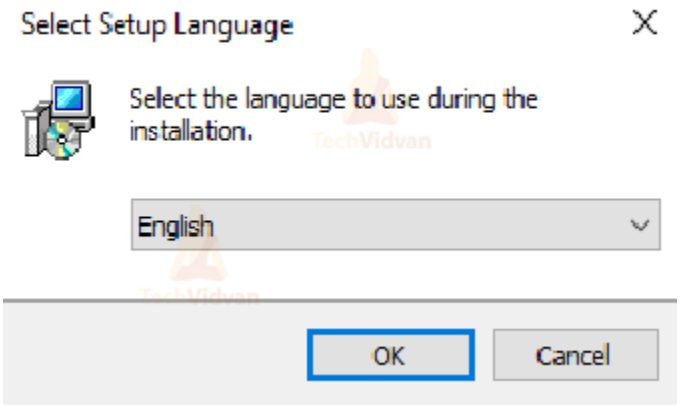
Step – 3: Click on the **base** subdirectory link or **install R for the first time** link.

Step – 4: Click **Download R X.X.X for Windows** (X.X.X stand for the latest version of R. eg: 3.6.1) and save the executable .exe file.

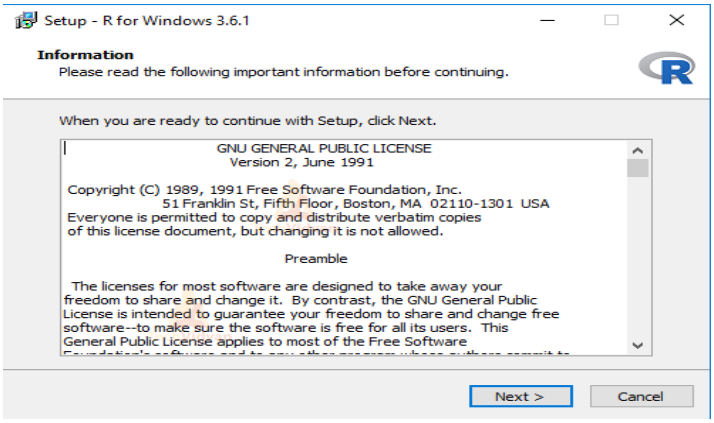


Step – 5: Run the **.exe** file and follow the installation instructions.

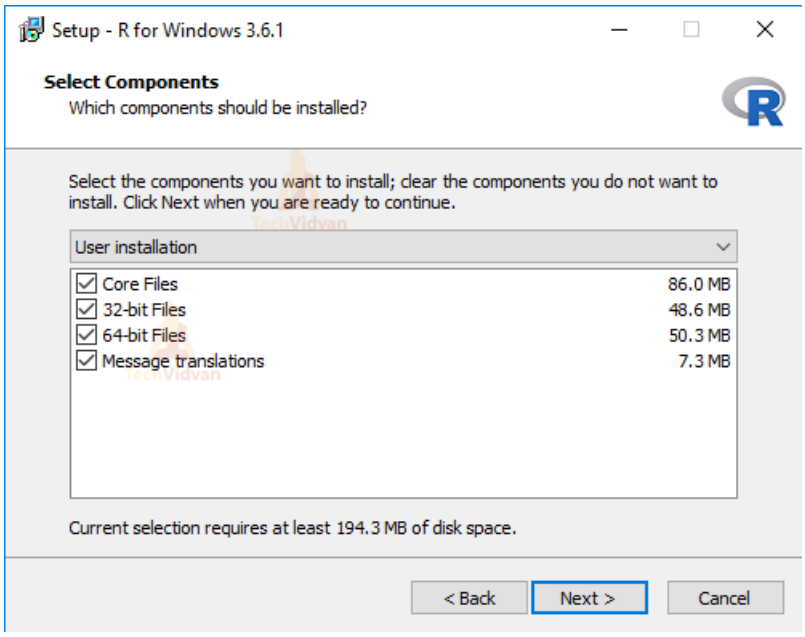
5.a. Select the desired language and then click **Next**.



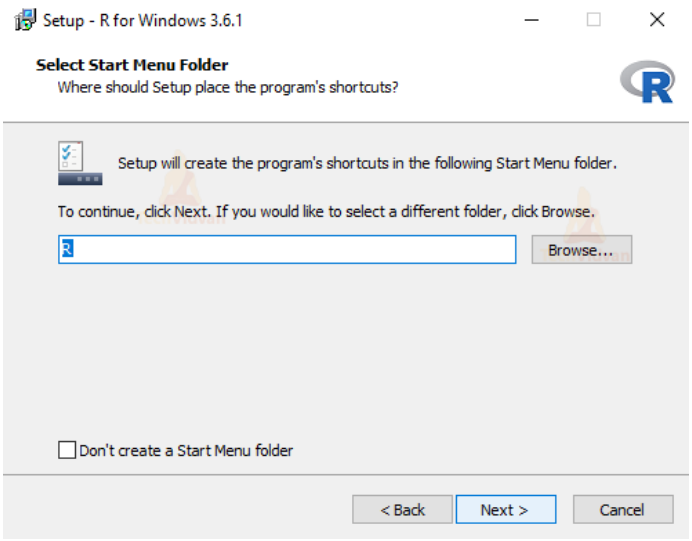
5.b. Read the license agreement and click **Next**.



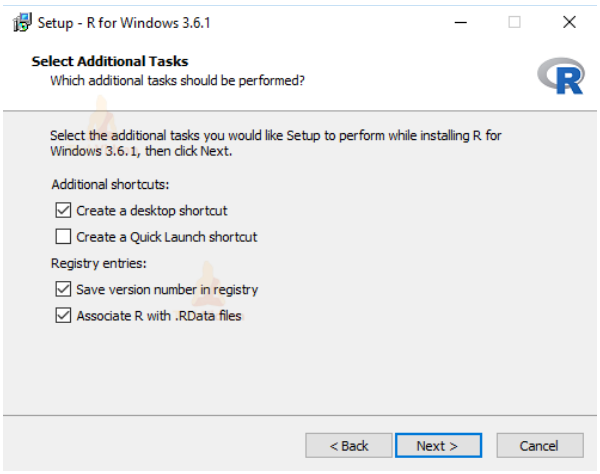
5.c. Select the components you wish to install (it is recommended to install all the components). Click **Next**.



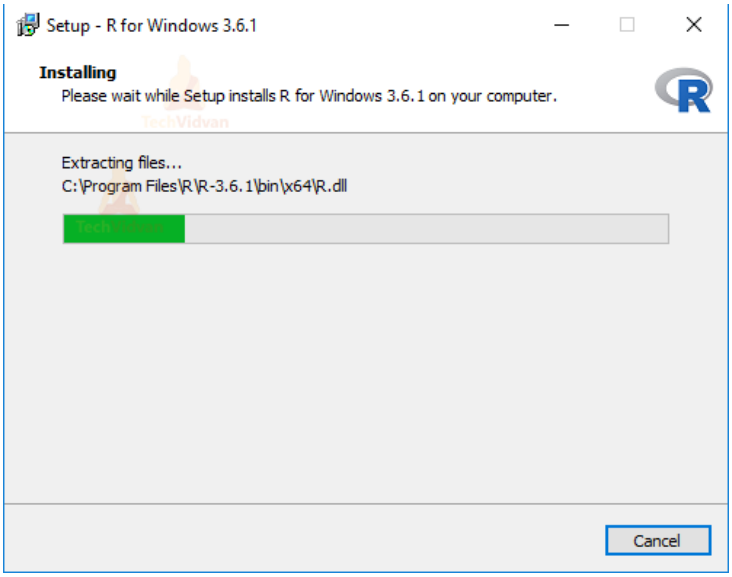
5.d. Enter/browse the folder/path you wish to install R into and then confirm by clicking **Next**.



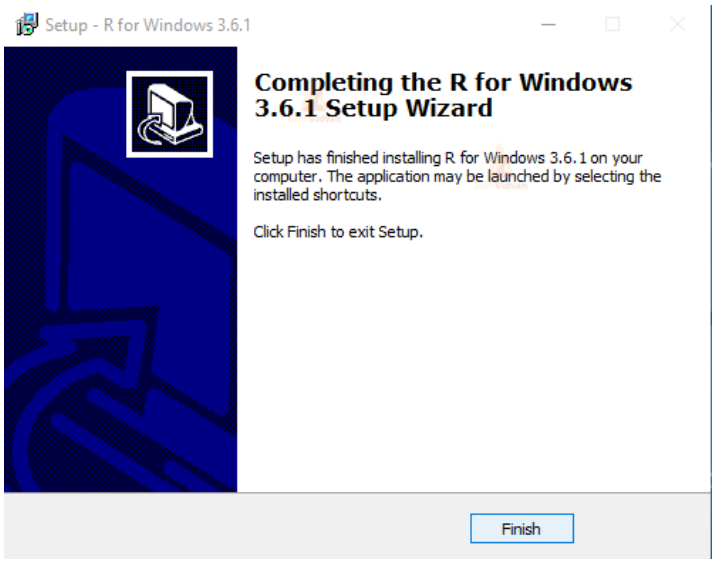
5.e. Select additional tasks like creating desktop shortcuts etc. then click **Next**.



5.f. Wait for the installation process to complete.



5.g. Click on **Finish** to complete the installation.

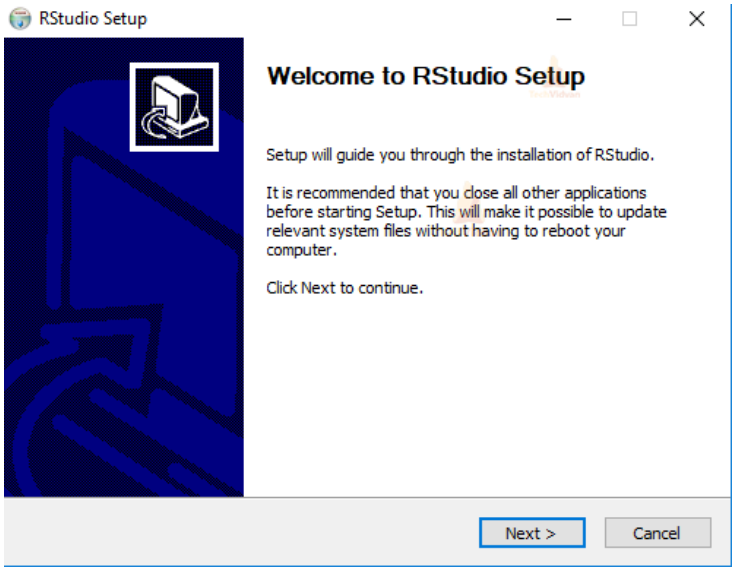


Install RStudio on Windows:

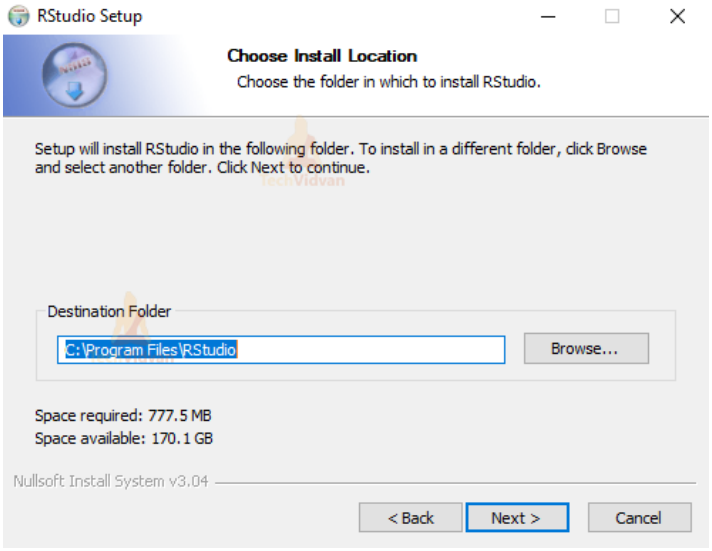
Step – 1: With R-base installed, let's move on to installing RStudio. To begin, go to download RStudio and click on the download button for **RStudio desktop**.

Step – 2: Click on the link for the windows version of RStudio and save the .exe file.
Step – 3: Run the .exe and follow the installation instructions.

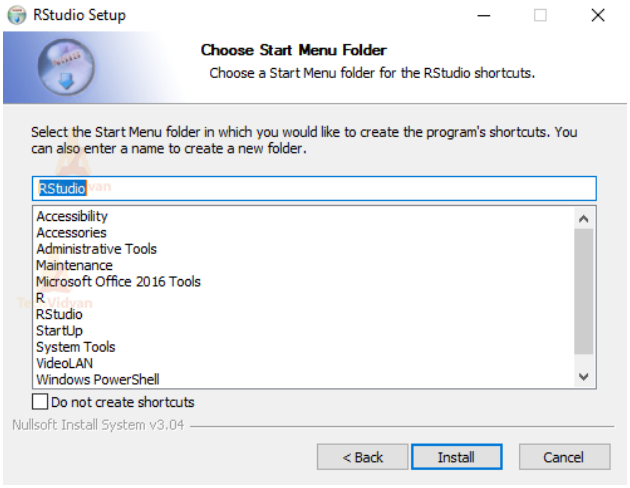
3.a. Click **Next** on the welcome window.



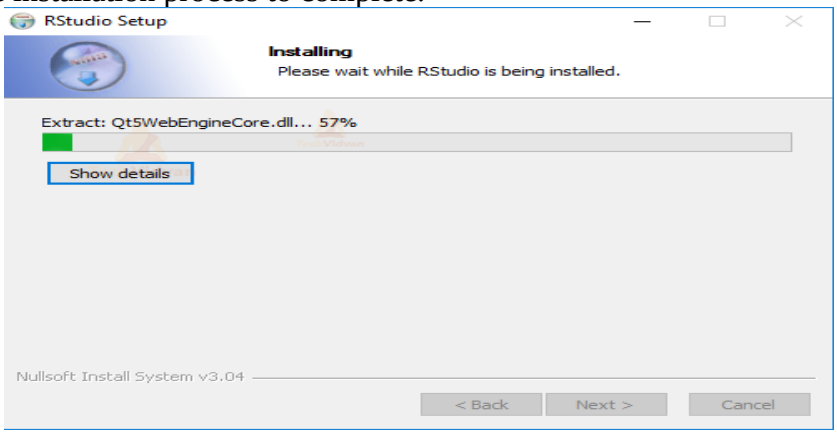
3.b. Enter/browse the path to the installation folder and click **Next** to proceed.



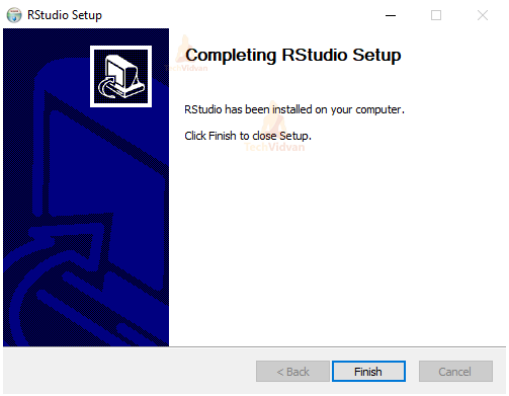
3.c. Select the folder for the start menu shortcut or click on do not create shortcuts and then click **Next**.



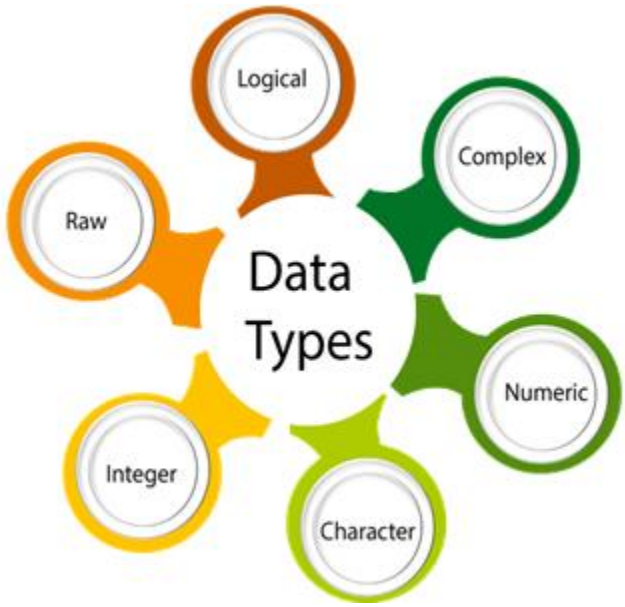
3.d. Wait for the installation process to complete.



3.e. Click **Finish** to end the installation.



Data Types:



1. **Logical:**

It is a special data type for data with only two possible values which can be construed as true/false.

Example:

True, False

2. **Numeric:**

Decimal value is called numeric in R, and it is the default computational data types.

Example:

12,32,112,5432

3. **Integer:**

Here, L tells R to store the value as an integer

Example:

3L, 66L, 2346L

4. **Complex:**

A complex value in R is defined as the pure imaginary value i.

Example:

Z=1+2i, t=7+3i

5. **Character**

In R programming, a character is used to represent string values. We convert objects into character values with the help ofas.character() function.

Example:

'a', '"good"', "TRUE", '35.4'

Variables in R:

Variables are used to store the information to be manipulated and referenced in the R program. The R variable can store an atomic vector, a group of atomic vectors, or a combination of many R objects.

A valid variable name consists of letters, numbers and the dot or underline characters. The variable name starts with a letter or the dot not followed by a number.

Example:

Var.1 = c(0,1,2,3)

Operators in R programming:

Operators		**Description**
Arithmetic Operators()	+	Adds two vectors
	-	Subtracts second vector from the first
	*	Multiplies both vectors
	/	Divide the first vector with the second
	%%	Give the remainder of the first vector with the second
	%/%	The result of division of first vector with second (quotient)
	^	The first vector raised to the exponent of second

		vector
Relational Operators	>	Checks if each element of the first vector is greater than the corresponding element of the second vector
	<	Checks if each element of the first vector is less than the corresponding element of the second vector.
	==	Checks if each element of the first vector is equal to the corresponding element of the second vector.
	<=	Checks if each element of the first vector is less than or equal to the corresponding element of the second vector.
	>=	Checks if each element of the first vector is greater than or equal to the corresponding element of the second vector.
	!=	Checks if each element of the first vector is unequal to the corresponding element of the second vector.
Logical Operators	&&	Called Logical AND operator. Takes first element of both the vectors and gives the TRUE only if both are TRUE.
	||	Called Logical OR operator. Takes first element of both the vectors and gives the TRUE if one of them is TRUE.
Assignment Operators	<− or = or <<−	Called Left Assignment
	-> or ->>	Called Right Assignment
Miscellaneous Operators	:	Colon operator. It creates the series of numbers in sequence for a vector.
	%in%	This operator is used to identify if an element belongs to a vector.
	%*%	This operator is used to multiply a matrix with its transpose.

Practice Programs:

1. Write a R program to take input from the user (name and age) and display the values. Also print the version of R installation.
2. Write a R program to get the details of the objects in memory.

SET A:

1. Write a R program to accept dimensions of a cylinder and print the surface area and volume.
2. Write a R program to accept temperatures in Fahrenheit (F) and print it in Celsius(C) and Kelvin (K).
3. Write a R program to accept two numbers and print arithmetic and harmonic mean of the two number.
4. Accept three dimensions length (l), breadth(b) and height(h) of a cuboid and print surface area and volume

SET B:

1. Accept the x and y coordinates of two points and computes the distance between the two points.
2. A cashier has currency notes of denomination 1, 5 and 10. Accept the amount to be withdrawn from the user and print the total number of currency notes of each denomination the cashier will have to give.

SET C:

1. Write a R program to create a sequence of numbers from 20 to 50 and find the mean of numbers from 20 to 60 and sum of numbers from 51 to 91.

Assignment Evaluation

0: Not Done [ ]	**1: Incomplete [ ]**	**2: Late Complete [ ]**
3: Need Improvement [ ]	**4: Complete [ ]**	**5: Well Done [ ]**

Signature of Instructor

Assignment 2: Decision making and loop control structures

if Statement:

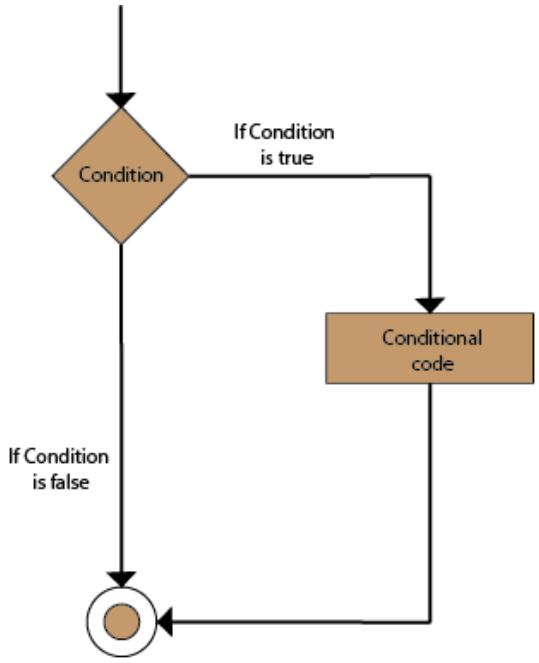
The if statement consists of the Boolean expressions followed by one or more statements. The if statement is the simplest decision-making statement which helps us to take a decision on the basis of the condition.

The if statement is a conditional programming statement which performs the function and displays the information if it is proved true.

The syntax of if statement in R is as follows:

```
if(boolean_expression) {
    // If the boolean expression is true, then statement(s) will be executed.
}
```

Flow Chart:



Example:

```
x <- 5
if(x > 0){
print("Positive number")
```

}

Output
[1] "Positive number"

If-else statement:

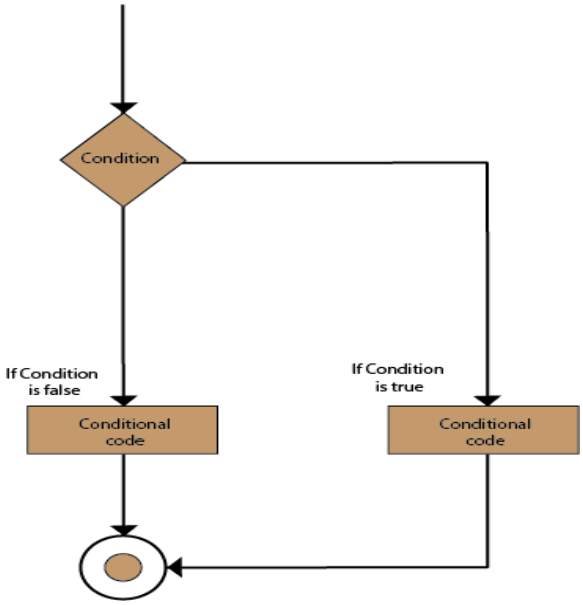
In the if statement, the inner code is executed when the condition is true. The code which is outside the if block will be executed when the if condition is false.

There is another type of decision-making statement known as the if-else statement. An if-else statement is the if statement followed by an else statement. An if-else statement, else statement will be executed when the boolean expression will false. In simple words, If a Boolean expression will have true value, then the if block gets executed otherwise, the else block will get executed.

The basic syntax of If-else statement is as follows:

```
if(boolean_expression) {
        // statement(s) will be executed if the boolean expression is true.
} else {
        // statement(s) will be executed if the boolean expression is false.
}
```

Flow Chart:



Example:

```
x <- -5
if(x > 0){
print("Non-negative number")
} else {
print("Negative number")
}
```

Output:

```
[1] "Negative number"
```

Switch Statement:

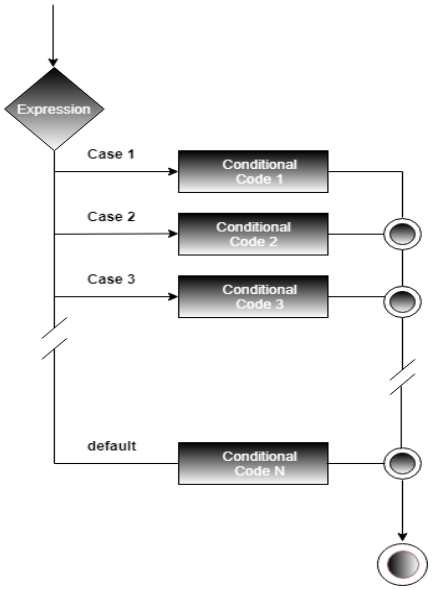
A switch statement is a selection control mechanism that allows the value of an expression to change the control flow of program execution via map and search.

The switch statement is used in place of long if statements which compare a variable with several integral values. It is a multi-way branch statement which provides an easy way to dispatch execution for different parts of code. This code is based on the value of the expression.

The basic syntax of If-else statement is as follows:

switch(expression, case1, case2, case3....)

Flow Chart:



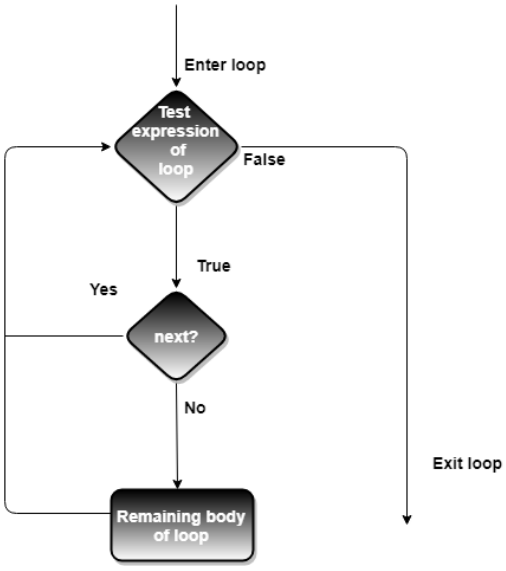
next Statement:

The next statement is used to skip any remaining statements in the loop and continue executing. In simple words, a next statement is a statement which skips the current iteration of a loop without terminating it. When the next statement is encountered, the R parser skips further evaluation and starts the next iteration of the loop.

Syntax

next

Flowchart



Example:

```
x <- 1:5
for (val in x) {
if (val == 3){
        next
}
print(val)
}
```

Output:

```
[1] 1
[1] 2
[1] 4
[1] 5
```

Break Statement:

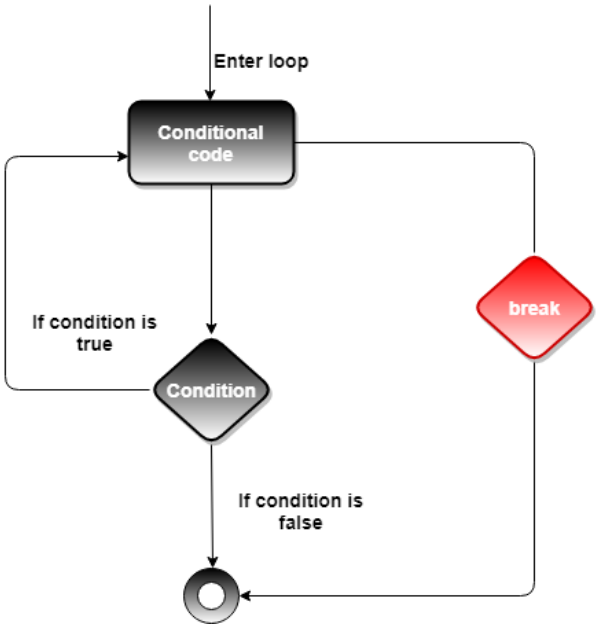
The break statement is used to break the execution and for an immediate exit from the loop. In nested loops, break exits from the innermost loop only and control transfer to the outer loop.

It is useful to manage and control the program execution flow. We can use it to various loops like: for, repeat, etc.

Syntax:

Break

Flowchart:



Example:

```
x <- 1:5
for (val in x) {
if (val == 3){
        break
}
print(val)
}
```

Output:

[1] 1

[1] 2

Loops:

The function of a looping statement is to execute a block of code, several times and to provide various control structures that allow for more complicated execution paths than a usual sequential execution.

Repeat Loop:

A repeat loop is one of the control statements in R programming that executes a set of statements in a loop until the exit condition specified in the loop, evaluates to TRUE.

Syntax

```
repeat{
        Statements
        if(exit_condition){
        break
}
}
```

Example:

```
x <- 1
repeat {
        print(x)
        x = x+1
        if (x == 6){
                break
        }
}
```

Output:

[1] 1
[1] 2
[1] 3
[1] 4
[1] 5

While Loop:

A while loop is one of the control statements in R programming which executes a set of statements in a loop until the condition (the Boolean expression) evaluates to TRUE.

```
while(Boolean expression)
{
        Statement
}
```

Example:

```
i<- 1
while (i< 6) {
        print(i)
        i = i+1
}
```

Output:

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

For Loop:

A for loop is the most popular control flow statement. A for loop is used to iterate a vector. It is similar to the while loop. There is only one difference between for and while, i.e., in while loop, the condition is checked before the execution of the body, but in for loop condition is checked after the execution of the body.

Syntax

```
for (value in vector) {
        statements
}
```

Example:

```
x <- c(2,5,3,9,8,11,6)
count<- 0
for (val in x) {
if(val %% 2 == 0)  count = count+1
}
print(count)
```

Output:

```
[1] 3
```

Practice Programs:

1. Write a R program to accept an integer and check if it is even or odd.
2. Write a R program, which accepts annual basic salary of an employee and calculates and displays the Income tax as per the following rules.
   Basic: < 1,50,000 Tax = 0
   Basic: 1,50,000 to 3,00,000 Tax = 20%
   Basic: > 3,00,000 Tax = 30%
3. Writet a R program to accept character from the user and check whether the character is a vowel or consonant.
4. Write a R program accept any year as input and check whether the year is a leap year or not.

SET A:

1. Write a R program to display the first 10 Fibonacci numbers.
2. Write a program to calculate sum of digits of a given input number.
3. Write program to check whether a input number is Armstrong number or not.
4. Write a program to check whether a input number is perfect number of not.
5. Write a R script to display multiplication table of a given input number.

SET B:

1. Write a R program to get all prime numbers up to a given number.
2. Accept the cost price and selling price from the keyboard. Find out if the seller has made a profit or loss and display how much profit or loss has been made.
3. Accept the x and y coordinate of a point and find the quadrant in which the point lies.
4. Write a program to check whether a input number is palindrome or not.
5. Write a program to accept a number and count number of even, odd, zero digits within that number.

SET C:

1. Use a while loop to simulate one stock price path starting at 100 and random normally distributed percentage jumps with mean 0 and standard deviation of 0.01 each period. How long does it take to reach above 150 or below 50?
2. Implement a multiplication game. A while loop that gives the user two random numbers from 2 to 12 and asks the user to multiply them without using * operator.

Assignment Evaluation

0: Not Done [ ]	**1: Incomplete [ ]**	**2: Late Complete [ ]**
3: Need Improvement [ ]	**4: Complete [ ]**	**5: Well Done [ ]**

Signature of Instructor

Assignment 3: String and Function in R Programming

String:

Any value written within a pair of single quote or double quotes in R is treated as a string. Internally R stores every string within double quotes, even when you create them with single quote.

Example:

```
string1 <- “This is a string”
```

R String Manipulation Functions

1. **grep()**

It is used for pattern matching and replacement. grep, grepl, regexpr, gregexpr and regexec search for matches with argument pattern within each element of a character vector. Here we subsitute the first and other matches with sub and gsub. sub and gsub perform replacement of the first and all matches.

Example:

```
g <- grep("iconHomicideShooting", homicides)
length(g)
```

2. **nchar()**

With the help of this function, we can count the characters. This function consists of a character vector as its argument which then returns a vector comprising of different sizes of the elements of x. nchar is the fastest way to find out if elements of a character vector are non-empty strings or not.

Example

```
# Using nchar() function
nchar("hel'lo")
```

3.substr():

It is the substrings of a character vector. The extractor replaces substrings in a character vector.

4. str_length()

The length of strings indicate the number of characters present in the string.The function **str_length()** belonging to the **‘stringr’** package or **nchar()** inbuilt function of R can be used to determine the length of strings in R.

Example

```
# Importing package
```

```
library(stringr)

# Calculating length of string
str_length("hello")
```

5. cat() function

Different types of strings can be concatenated together using the **cat()**) function in R, where sep specifies the separator to give between the strings and file name, in case we wish to write the contents onto a file.

Syntax:

```
cat(..., sep=" ", file)
```

Example:

```
# Concatenation using cat() function
str<- cat("learn", "code", "tech", sep = ":")
print (str)
```

6. Conversion to upper case

All the characters of the strings specified are converted to upper case.

Example:

```
print(toupper(c("Learn Code", "hI")))
```

7. Conversion to lower case

All the characters of the strings specified are converted to lower case.

Example:

```
print(tolower(c("Learn Code", "hI")))
```

8. Character replacement

Characters can be translated using the chartr(oldchar, newchar, …) function in R, where every instance of old character is replaced by the new character in the specified set of strings.

Example:

chartr("a", "A", "An honest man gave that")

Output:

"An honest mAngAvethAt"

9. Splitting the string

A string can be split into corresponding individual strings using ” ” the default separator.

Example:

strsplit("Learn Code Teach !", " ")

Output:

[1] "Learn" "Code" "Teach" "!"

10. Working with substrings

substr(…, start, end) or substring(…, start, end) function in R extracts substrings out of a string beginning with the start index and ending with the end index. It also replaces the specified substring with a new set of characters.

Example:

substr("Learn Code Tech", 1, 4)

Output:

"Lear"

Function in R:

A **function**, in a programming environment, is a set of instructions. A programmer builds a function to avoid **repeating the** same task, or reduce **complexity.**

A function should be

- written to carry out a specified a tasks
- may or may not include arguments
- contain a body
- may or may not return one or more values.

Syntax

```
func_name<- function (argument) {

        statement

}
```

Example:

```
Pow <- function(x,y){
# function to print x raised to the power y
Result <- x^y
Print(paste(x,"raised to the power", y,"is",result))
}
```

Output:

```
>pow(8,2)
[1] "8 raised to the power 2 is 64"
```

1. Default Values for Arguments

We can assign default values to arguments in a function in R.

Example

```
pow<- function(x, y = 2) {
# function to print x raised to the power y
result<- x^y
print(paste(x,"raised to the power", y, "is", result))
}
```

Output:

```
>pow(3)
[1] "3 raised to the power 2 is 9"
```

3. **Return Value from Function**

Many a times, we will require our functions to do some processing and return back the result. This is accomplished with the return() function in R.

Syntax

```
return(expression)
```

Example:

```
check<- function(x) {
        if (x > 0) {
        result<- "Positive"
        }
        else if (x < 0) {
```

```
        result<- "Negative"
    }
    else {
        result<- "Zero"
    }
        return(result)
    }
```

Output:

```
>check(1)
[1] "Positive"
>check(-10)
[1] "Negative"
```

3. Recursive Function:

A function that calls itself is called a recursive function and this technique is known as recursion.This special programming technique can be used to solve problems by breaking them into smaller and simpler sub-problems.

Example:

```
# Recursive function to find factorial
recursive.factorial<- function(x) {
    if (x == 0)    return (1)
    else          return (x * recursive.factorial(x-1))
}
```

Output:

```
>recursive.factorial(0)
[1] 1
>recursive.factorial(5)
[1] 120
```

In-built Functions:

These functions in R programming are provided by R environment for direct execution, to make our work easier.

Function Name	Description	Example
seq()	To create a sequence of numbers	print(seq(1,9)) O/P [1] 1 2 3 4 5 6 7 8 9
sum()	To find the sum of numbers	print(sum(25,50)) O/P [1] 75
mean()	To find the mean of numbers	print(mean(41:68)) O/P [1] 54.5
paste()	To combine vectors after converting them to characters	paste(1,"sam",2,"rob",3,"max") O/P [1] "1 sam 2 rob 3 max"
min()	To return the minimum value from a vector.	x<-c(3,45,6,7,89,9)print(min(x)) O/P [1] 3
max()	To return the maximum value from a vector	x <- c(3,45,6,7,89,9) print(max(x)) O/P [1] 89

Practice Programs:

1. Write a R program to accept a string from user and display the length of the string.
2. Write a R program to accept a string in lowercase and display it uppercase and vice versa
3. Write a program to check whether a input number is prime number or not using user defined function.

SET A:

1. Write a program to calculate factorial of a input number using user defined function.
2. Write R program to find the factors of a given number using user defined function
3. Write a R Program to connect two different strings.

SET B:

1. Write a program to calculate x^y using user defined function (Use default parameters)
2. Write R program to accept a string and character from user and replace all occurrences of that character from string with other character.
3. Write a recursive function in R to calculate multiplication of all digits of a given input number.

4. Write a function which accepts one number. Function should return 1 if the number is Perfect No, otherwise function should return 0.
5. Write a function isPrime, which accepts an integer as parameter and returns 1 if the number is prime and 0 otherwise.

SET C:

1. Write a R program to print the numbers from 1 to 100 and print "Fizz" for multiples of 3, print "Buzz" for multiples of 5, and print "FizzBuzz" for multiples of both.
2. Write a program to calculate sum of following series up to n terms using user defined function

$$Sum=X+X^2/2!+X^3/3!+\cdots\cdots$$

Assignment Evaluation

0: Not Done [ ]	**1: Incomplete [ ]**	**2: Late Complete [ ]**
3: Need Improvement [ ]	**4: Complete [ ]**	**5: Well Done [ ]**

Signature of Instructor

Assignment 4: Vector and List in R programming

Introduction:

The Vector is the most basic Data structure in R programming. R Vector can hold a collection of elements of similar types. A vector supports logical, integer, numeric, character, complex, or raw data type. The elements which are contained in vector known as components of the vector. We can check the type of vector with the help of the typeof() function. The length is an important property of a vector. A vector length is basically the number of elements in the vector, and it is calculated with the help of the length () function. Vector is classified into two parts, i.e., Atomic vectors and Lists.

There is only one difference between atomic vectors and lists. In an atomic vector, all the elements are of the same type, but in the list, the elements are of different data types.

Creating vector in R:

In R, we use c () function to create a vector. This function returns a one-dimensional array or simply vector. The c() function is a generic function which combines its argument. All arguments are restricted with a common data type which is the type of the returned value.

For Example:

1. **# Numeric Vector:** a=c(1, 2, 3, 4)
2. **# Character vector:** b=c("India", "China", "Japan", "Russia“)
3. **#Boolean vector:** d=c(TRUE, FALSE, FALSE, TRUE, TRUE)
4. **# Mixed Vector and its Type will be Character** : e= c("India", 2, "China", 1, TRUE)
5. **# Placing or Nesting One Vector inside the another** : f = c("UK", "USA", TRUE, FALSE, b) where b is another Vector

There are various other ways to create a vector in R, which are as follows:

1. **Create R Vector using Range:** In R programming, there is a special operator called Range or Colon, and this will help to create a vector. For example
    1. **# Vector with Range :** i =1 : 10
    2. **Vector with Decimal Range :** j =1.5 : 5.5
    3. **# Vector with Decimal Range :** n = -10 : -20
    4. **Letter Vector with Range:** l =letters[1:6]
2. **Using the seq() function:** In R, we can create a vector with the help of the seq() function. A sequence function creates a sequence of elements as a vector. For example
    1. v1=seq(1,5) #v1 will be 1,2,3,4,5

2. v2=seq(1,10,by=2) #v2 will be 1,3,5,7,9
3. v3=seq(1,4,length.out=6) #v3 will be 1.0,1.6,2.2,2.8,3.4,4.0

Atomic vectors in R:

Atomic vectors are created with the help of **c()** function. In R, there are four types of atomic vectors. These atomic vectors are numeric vector, integer vector, character vector and logical vector.

1. **Numeric vector:** The decimal values are known as numeric data types in R. If we assign a decimal value to any variable d, then this d variable will become a numeric type. A vector which contains numeric elements is known as a numeric vector. For example

```
> d=c(10.5, 24.5,7)
> d                    #prints 10.5 24.5 7.0 at console
>class(d)              #prints  "numeric"
```

2. **Integer Vector:** A non-fraction numeric value is known as integer data. There is two way to assign an integer value to a variable, i.e., by using as.integer() function and appending of L to the value. A vector which contains integer elements is known as an integer vector. For example

```
>int_vec=as.interger(c(1,2,3,4,5) )
>int_vec1=c(1L,2L,3L,4L,5L)
>d=as.integer(5)
>e=5L
```

3. **Character Vector:** In R, there are two different ways to create a character data type value, i.e., using as.character() function and by typing string between double quotes("") or single quotes('). A vector which contains character elements is known as an character vector. For example

```
>f=c("shubham","arpita","nishka","vaishali")
>g=as.character(c(123, 234))
>d='shubham'
>e="Arpita"
```

4. **Logical vector:** The logical data types have only two values i.e., True or False. These values are based on which condition is satisfied. A vector which contains Boolean values is known as the logical vector. For example

```
> a=10
> b=4
> c=8
> log_vec=c(a>b, b<c, c>a, c<a)
> log_vec                    # it prints TRUE TRUE FALSE TRUE
```

Accessing elements of vectors:

We can access the elements of a vector with the help of vector indexing. Indexing denotes the position where the value in a vector is stored. Indexing will be performed with the help of integer, character, or logic.

1. **Indexing with integer vector:** On integer vector, indexing is performed in the same way as we have applied in C. There is only one difference, i.e., in C the indexing starts from 0, but in R, the indexing starts from 1. we perform indexing by specifying an integer value in square braces [] next to our vector. For example

```
> d=c(10,20,30,40,50)
> d                    #Prints 10 20 30 40 50
> d[2]                 # Prints 20
> d[3]                 # Prints 30
> d[2:4]               #Prints 20 30 40
```

2. **Indexing with a character vector:** In character vector indexing, we assign a unique key to each element of the vector. These keys are uniquely defined as each element and can be accessed very easily. For example

```
>  Stud=c("Rollno"=101, "Marks"=80.74)
 > Stud["Rollno"]                  #prints  101
>Stud["Marks"]                          #prints 80.74
>Stud[c("Rollno","Marks")]        #prints 101 80.74
```

3. **Indexing with a logical vector**: In logical indexing, it returns the values of those positions whose corresponding position has a logical vector TRUE. For example

```
> vec=c(1,2,3,4,5,6)
>vec[c(TRUE,TRUE,FALSE,FALSE,TRUE,FALSE)]                    #It
```

prints 1 2 5

4. **Access using Vector**: In this example, we will show how to access the Vector elements using another Vector in R. for example

```
 >a =c("India", "China", "Japan", "UK", "USA", "Russia", "Sri Lanka")
> b =c(2, 4, 6)
>print(a[b])                 # It  prints      China  UK      Russia
>print(a[c(5, 7)])           # It  prints USA         Sri Lanka
>print(a[c(7, 4, 1)])        # It  prints Sri Lanka   UK      India
```

5. **Using Negative Values in R:** We can access the Vector elements using Negative values and the Boolean values. In R Vectors, Negative index position is used to omit those values. For example

```
>a=c("India", "China", "Japan", "UK", "USA", "Russia")
>print(a[-3])                #it prints India China  UK      USA    Russia
>b =c(-3, -6)
>print(a[b])                 #it prints India China  UK      USA
```

```
>print(a[c(-4, -6)])          #it prints "India" "China" "Japan" "USA"
```

Manipulate R Vector Elements:

In R Programming, we can manipulate the Vector elements in following ways:

```
>a <- c(10, 20, 30, -15, 40, -25, 60, -5)
>a[7] =77                    #it update the 7th element of vector to 77
>a[a < 0]=99   #it modifies all the elements of vector to 99 if the element is less than 99
>a= a[1:5]     # it truncates all the elements of vector except elements 1 to 5
> a= NULL     #it deletes the vector a
```

Vector Operation:

1. **Combining Vectors:** The c() function is not only used to create a vector, but also it is also used to combine two vectors. By combining one or more vectors, it forms a new vector which contains all the elements of each vector.

```
>a=c(1,2,3)
>b=c("p","q","r")
>c=c(a,b)
>c                          #prints "1" "2" "3" "p" "q" "r“
>d=(b,a)
>d                          #prints "p" "q" "r" "1" "2" "3"
```

2. **Arithmetic operations:** We can perform all the arithmetic operation on vectors. The arithmetic operations are performed member-by-member on vectors. We can add, subtract, multiply, or divide two vectors. For example

```
> a=c(8,4,10)
> b=c(2,6,5)
> a+b                       #10 10 15
> a-b                       #6 -2 5
> a*b                       #16 24 50
> a/b                       #4.0000000 0.6666667 2.0000000
> a%%b                      #0 4 0
> a%/%b                     # 4 0 2
```

Important Functions of Vector:

1. **typeof(Vector):** This method tells you the data type of the vector.
2. **Sort(Vector):** This method helps us to sort the items in the Ascending order.
3. **length(Vector):** This method counts the number of elements in a vector.
4. **head(Vector, limit):** This method return the top six elements (if you Omit the limit). If you specify the limit as 4 then, it returns the first 4 elements.
5. **tail(Vector, limit):** It returns the last six elements (if you Omit the limit). If you specify the limit as 2, then it returns the last two elements.

R List:

Lists are the objects of R which contain elements of different types such as number, vectors, string and another list inside it. It can also contain a function or a matrix as its elements.

A list is a data structure which has components of mixed data types. We can say, a list is a generic vector which contains other objects.

In R, the list is created with the help of list() function.

```
> a_vec=c(1,2,3)
> b_vec=c("Pune","Mumbai")
>list_var=list(2.3,45L,"R Programming", TRUE, a_vec,b_vec)
> print(list_var)
```

Here print(list_var) will display the content of list on console as ouput

Giving a name to list elements:

R provides a very easy way for accessing elements, i.e., by giving the name to each element of a list. After creating list we can assign a name to the list elements with the help of names() function. For example

```
> list_var=list(101L,"Nilesh",80.45)
> names(list_var)=c("Roll No","Name","Marks")
> print(list_var)
        $`Roll No`
                [1] 101
        $Name
                [1] "Nilesh“
        $Marks
                [1] 80.45
```

Accessing List Elements:

R provides two ways through which we can access the elements of a list. First one is the indexing method performed in the same way as a vector. In the second one, we can access the elements of a list with the help of names. It will be possible only with the named list.

```
> list1=list (10, 20, 30)
> list1[1]                    # display  10
> list1[2]                    #display 20
> for(i in list1)
        + print(i)            # display 10 20 30
> list2=list(101,"Nilesh",80.57)
>names(list2)=c("Rollno", "Name", "Marks")
> list2["Rollno"]                    #display $Rllono 101
```

Manipulation of list elements:

R allows us to add, delete or update elements in the list. We can update an element of a list from anywhere, but elements can add only at the end of the list. To remove an element from a specified index, we will assign it a NULL value.

We can update the element of a list by overriding it from the new value.

```
> list1=list (10, 20, 30)
>list1                  #display 10 20 30
>list1 [4] =40
>list1                  #display 10 20 30 40
>list1[2]=200
>list1                  #display 10 200 30 40
>list1[4]=NULL
>list1                  #display 10 200 30
```

Converting list to vector:

There is a drawback with the list, i.e., we cannot perform all the arithmetic operations on list elements. To remove this, drawback R provides unlist() function. This function converts the list into vectors. In some cases, it is required to convert a list into a vector so that we can use the elements of the vector for further manipulation.The unlist() function takes the list as a parameter and change into a vector.

```
> list1=list(2:5)              #list1=(2,3,4,5)
> list2=list(11:14)            #list2=(11,12,13,14)
> list1+list2                  #Error
> v1=unlist(list1)             #converting list1 to vector v1
> v2=unlist(list2)             # converting list1 to vector v2
> v1+v2                        #display 13 15 17 19
```

Merging List:

R allows us to merge one or more lists into one list. Merging is done with the help of the list () function also. To merge the lists, we have to pass all the lists into list function as a parameter, and it returns a list which contains all the elements which are present in the lists.

```
>even=list (2, 4, 6)
>odd=list (1, 3, 5)
> mix=list(even, odd)
> print (mix)                  #display 2 4 6 1 3 5
```

Sample R Scripts using Vector and List

Q.) Write an R program to find the maximum and the minimum value of a given vector.

Solution:

```
n=as.integer(readline(prompt="How many nos do u want to strore in vector="))
```

```
vec=c()
a=1
while(a<=n)
{
        num=as.integer(readline(prompt="Enter Elemnt="))
        vec[a]=num
        a=a+1
}
cat("\n Original Vector=")
print(vec)
cat("\n Maximum elemennt of vector=",max(vec))
cat("\n Minimum elemennt of vector=",min(vec))
```

Q.) Write an R program to sort a list of 10 strings in ascending and descending order.

```
n=as.integer(readline(prompt="How many strings u want to store in list="))
lst=list()

for(a in seq(1,n))
{
  lst[a]=readline(prompt="Enter any String=")
}

b=unlist(lst)
b=sort(b)
lst=list(b)
cat("\n List in Ascending Order=")
print(lst)

b=sort(b,decreasing=TRUE)
lst=list(b)
cat("\n List in Descending Order=")
print(lst)
```

Practice Programs:

1. Write a R program to create a vector of a specified type and length. Create vector of numeric, complex, logical and character types of length 6.
2. Write a R program to add, multiply and divide two vectors of integers type and length 3.
3. Write a R program to create a list containing strings, numbers, vectors and a logical values.

4. Write a R program to list containing a vector, a matrix and a list and give names to the elements in the list. Access the first and second element of the list.

SET A:

1. Write an R program to sort a Vector in ascending and descending order.
2. Write an R program to find Sum, Mean and Product of a Vector.
3. Write a R program to sort a Vector in ascending and descending order.
4. Create a list containing a four vectors and give names to the elements in the list
5. Write a R program to merge two given lists into one list.
6. Write a R program to convert a given list to vector.
7. Write a R program to create a list named s containing sequence of 15 capital letters, starting from 'E'.

SET B:

1. Write a R program to find all elements of a given list that are not in another given list.
2. Write a R program to extract all elements except the third element of the first vector of a given list.
3. Write a script in R to create a list of cities and perform the following
    a. Give names to the elements in the list.
    b. Add an element at the end of the list.
    c. Remove the last element.
    d. Update the 3rd Element
4. Write a script in R to create a list of students and perform the following
    a. Give names to the students in the list.
    b. Add a student at the end of the list.
    c. Remove the first Student.
    d. Update the second last student.
5. Write a script in R to create a vector of numbers and perform the following
    a. Search for specific element
    b. Count the occurrences of specific element
    c. Access the last element of given vector

SET C:

1. Write a R program to extract every n^{th} element of a given vector.
2. Write a R program to select second element of a given nested list.

Assignment Evaluation

0: Not Done [ ]	**1: Incomplete [ ]**	**2: Late Complete [ ]**
3: Need Improvement [ ]	**4: Complete [ ]**	**5: Well Done [ ]**

Signature of Instructor

Assignment 5: Arrays and Matrices in R programming

R Arrays:

In R, arrays are the data objects which allow us to store data in more than two dimensions. In R, an array is created with the help of the array() function.

Syntax: array_name <- array (data, dim = (row_size, column_size, matrices), dim_names))

Where,

1. **data:** The data is the first argument in the array() function. It is an input vector which is given to the array.
2. **row_size:** This parameter defines the number of row elements which an array can store.
3. **column_size:** This parameter defines the number of columns elements which an array can store.
4. **Matrices:** This parameter defines number of arrays to create.
5. **dim_names:** This parameter is used to change the default names of rows and columns. It is list of 3 vectors, where first vector correspond to row names, second vector represent column names and third vector represent matrix name.

Ex:

```
>d=array(c(1,2,3,4,5,6,7,8,9),dim=c(3,3,1),dimnames=list(c("r1","r2","r3"),c("c1","c2","c3"),c("m1")))
>print(d)
, , m1

   c1 c2 c3
r1  1  4  7
r2  2  5  8
r3  3  6  9
```

Accessing R Array Elements:

In R programming, we can use the index position to access the array elements. Using the index, we can access or alter/change each and every individual element present in an array. Index value starts at 1 and an end at n where n is the size of a matrix, row, or column.

The syntax behind this R Array accessing is: *Array_Name[row_position, Column_Position, Matrix_Level]*.

For example, we declared an array of two matrices of size 6 rows * 4 columns.

To access or alter 1st value use Array_name[1, 1, 1], to access or alter 2nd-row 3rd column value at 1st Matrix level then use Array_name[2, 3, 1]

Ex. >A[2,3,1]=10

Accessing Subset of a Array Elements:

In our previous example, we show you how to access the single element from tan Array. In this example, we will show how to access the subset of multiple items from the Array. To achieve the same, we use the R array A as follows

```
>A =array(1: 24, dim = c(3, 4, 2))
# Access the elements of 1st, 3rd row and 2nd, 4th column in Matrix 1.
        >print(A[c(1, 3), c(2, 4), 1])
# Access all the element of 2nd and 3rd row in Matrix 2.
        >print(A[c(2, 3), , 2])
# Access all the element of 1st and 4th Column in Matrix 1.
        >print(A[ , c(1, 4), 1])
TIP: Negative index position is used to omit those values from an Array.
# Access all the element except 2nd row and 3rd Columm in Matrix 2.
        >print(A[-2, -3, 2])
```

R Array Addition and Subtraction:

In this example, we show how to use Arithmetic Operators on Matrices to perform arithmetic Operations on Array.

```
# Adding and Subtracting Elements of Array in R
    >vect1= c(10, 20, 40 )
    >vect2=c(55, 67, 89, 96, 100)
    >A=array(c(vect1, vect2), dim = c(3, 4, 2))
    >print(A)
    >mat.A=A[, , 1]
    >mat.B=A[, , 2]
    >print(mat.A + mat.B)
    >print(mat.B - mat.A)
```

R Matrix:

The Matrix in R is the most two-dimensional Data structure. In R Matrix, data is stored in row and columns, and we can access the matrix element using both the row index and column index. A matrix is created with the help of the vector input to the matrix function. On R matrices, we can perform addition, subtraction, multiplication, and division operation. In the R matrix, elements are arranged in a fixed number of rows and columns. In R, we use matrix function, which can easily reproduce the memory representation of the matrix. In the R matrix, all the elements must share a common basic type.

R provides the matrix() function to create a matrix.

Syntax of creating Matrix: *matrix(data, nrow, ncol, byrow, dim_name)*

Where,

1. **Data:** The first argument in matrix function is data. It is the input vector which is the data elements of the matrix.
2. **Nrow:** It is the number of rows in the matrix.
3. **Ncol:** It is the number of columns in the matrix.
4. **Byrow:** If its value is true, then the input vector elements are arranged by row.
5. **dim_name:** The dim_name parameter is the name assigned to the rows and columns.

For Example M1= matrix(c (11, 13, 15, 12, 14, 16), nrow =2, ncol =3, byrow = TRUE)

R = matrix(c(3:14), nrow = 4, byrow = TRUE, dimnames = list(row_names, col_names))

Different ways of creating Matrix in R:

1. # R Create Matrix

   >A=matrix(c(1:12), nrow = 3, ncol = 4) >print(A)
2. # Elements are arranged sequentially by column.

   >B=matrix(c(1:12), nrow = 3, ncol = 4, byrow = FALSE) >print(B)
3. # Elements are arranged sequentially by row.

   >D=matrix(c(1:12), nrow = 3, ncol = 4, byrow = TRUE)
4. # It will create a Matrix of 3 Rows and the remaining elements will be arranged Accordingly

   >A=matrix(c(1:12), nrow = 3)
5. # It will create a Matrix of 4 Columns and the remaining (row) elements will be arranged Accordingly

   >B=matrix(c(1:12), ncol = 4)
6. # It will create a Matrix of 3 rows and 4 Columns

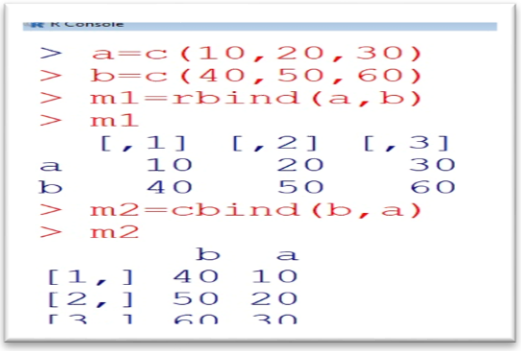
   > D=matrix(c(1:12), 3, 4)
7. # It will create a Matrix of 3 rows

   >E=matrix(c(1:12), 3)
8. # It will create a Matrix of 4 Rows. To create 4 Columns you have to specify *ncol = 4* explicitly

   >G=matrix(c(1:12), ncol=4)

Create R Matrix using cbind and rbind:

In this example, we will show you another way of creating a Matrix in R programming. ***cbind*** is used for binding vectors in Columns wise, and the ***rbind*** is used for binding vectors in Row wise.



Define Row names and Column names for matrix in R:

```
> M1=matrix(c(1,2,3,4,5,6),2,3,byrow=TRUE,dimnames=list(c("A","B"),c("P","Q","R")))
> M1
  P Q R
A 1 2 3
B 4 5 6
>
```

Accessing matrix elements in R:

There are three ways to access the elements from the matrix.

1. We can access the element which presents on nth row and mth column.
2. We can access all the elements of the matrix which are present on the nth row.
3. We can also access all the elements of the matrix which are present on the mth column.

E.g. A=matrix(c(1:12), nrow = 3, ncol = 4, byrow = TRUE)

I. # Access the element at 1st row and 2nd column.
>print(A[1, 2])

II. # Access the element at 3rd row and 4th column.
>print(A[3, 4])

III. # Access only the 2nd row.
>print(A[2,])

IV. # Access only the 4th column.
>print(A[, 4])

V. # Access Complete Matrix.
>print(A[,])

Accessing Subset of a Matrix in R: Following are the examples of accessing subset of a matrix

1. A =matrix(c(1:12), nrow = 3, ncol = 4, byrow = TRUE)
>print(A)
2. # Access the elements at 1st, 2rd row and 3nd, 4th column.

```
>print(A[c(1, 2), c(3, 4)])
```

3. # Access All the element at 2nd and 3rd row.
```
>print(A[c(2, 3), ])
```
4. # Access All the element at 1st and 4th Column.
```
>print(A[ , c(1, 4)])
```
5. # Access All the element except 2nd row.
```
>print(A[-2, ])
```
6. # Access All the element except 2nd row and 3rd Columm.
```
>print(A[-2, -3])
```
7. # Access All the element except 3rd and 4th Columm.
```
>print(A[, c(-3, -4)])
```

Accessing R Matrix Elements using Character Index:

By assigning the Row names and Columns Names can help us to extract the Matrix elements using the Row names or column names as the Index values.

```
> row.names=c("Row1", "Row2", "Row3")            #
>column.names=c("Col1", "Col2", "Col3", "Col4")
>B=matrix(c(1:12), nrow = 3, dimnames = list(row.names, column.names))
```

1. # Access the elements at 1st row and 3rd Column.
```
>print(B["Row1", "Col3"])
```
2. # Access only the 2nd row.
```
>print(B["Row2",])
```
3. # Access only the 4th column.
```
print(B[, "Col4"])
```
4. # Access the elements at 2nd row and 2, 3, 4th Column.
```
>print(B["Row2", 2:4])
```
5. # Access the elements at 1st, 3rd row and 1, 2, 3rd Column.
```
>print(B[c("Row1", "Row3"), 1:3])
```

Modify R Matrix Elements:

In R programming, We can use the index position to modify the elements in a Matrix. Using this index value, we can access or alter/change each and every individual element present in the vector. For example, if we declare a 3 * 4 matrix that stores 12 elements (3 rows and 4 columns). To access or alter 1st value use Matrix.name[1, 1], to access or alter 2nd row 3rd column value use Matrix.name[2, 3].

Modifying Matrix in R Programming

```
>A= matrix(c(1:9), nrow = 3, ncol = 3)
>A[2, 2] =100          #modify second row, second column element to 100
>A[A < 5] =222 # modifies all element to 222 if the element is less than 5
```

Matrix Arithmetic in R:

R Arithmetic Operators are used on Matrices to perform arithmetic Operations.

```
# Create 2x3 matrices.
>a=matrix( c(15, 34, 38, 44, 75, 93), nrow = 2)
>b=matrix( c(10, 20, 30, 40, 50, 60), nrow = 2)
```

1. # Addiing two Matrices
   ```
   >print(a + b)
   ```
2. # Subtraction One Matrix from another
   ```
   >print(a - b)
   ```
3. # R Matrix Multiplication
   ```
   >print(a * b)
   ```
4. # Matrix Division
   ```
   >print(a / b)
   ```

Practice Programs:

1. Write a R program to create an array of two 3x3 matrices each with 3 rows and 3 columns from two given two vectors.
2. Write a R program to create a blank matrix.
3. Write a R program to create a matrix taking a given vector of numbers as input. Display the matrix.
4. Write a R program to create a two-dimensional 5x3 array of sequence of even integers greater than 50.
5. Write a R program to convert a given matrix to a 1 dimensional array.

SET A:

1. Write a R program to create a matrix taking a given vector of numbers as input. Display the matrix.
2. Write a R program to create a matrix taking a given vector of numbers as input and define the column and row names. Display the matrix.
3. Write a R program to access the element at 3rd column and 2nd row, only the 3rd row and only the 4th column of a given matrix.
4. Write an R program to create three vectors a,b,c with 3 integers. Combine the three vectors to become a 3×3 matrix where each column represents a vector. Print the content of the matrix.
5. Write an R program to create a list of elements using vectors, matrices and a functions. Print the content of the list.

SET B:

1. Write a R program to create an array of three 3x2 matrices each with 3 rows and 2 columns from two given two vectors of different length.

2. Write a R program to create an array of two 3x3 matrices each with 3 rows and 3 columns from two given two vectors. Print the second row of the second matrix of the array and the element in the 3rd row and 3rd column of the 1st matrix.
3. Write a R program to access the element at 3rd column and 2nd row, only the 3rd row and only the 4th column of a given matrix.
4. Write a R program to create two 2x3 matrices and add, subtract, multiply and divide the matrix elements.
5. Write an R program to convert a given matrix to a list and print list in ascending order.

SET C:

1. Write a R program to combine three arrays so that the first row of the first array is followed by the first row of the second array and then first row of the third array.
2. Write a R program to find row and column index of maximum and minimum value in a given matrix.

Assignment Evaluation

0: Not Done [ ]	**1: Incomplete [ ]**	**2: Late Complete [ ]**
3: Need Improvement [ ]	**4: Complete [ ]**	**5: Well Done [ ]**

Signature of Instructor

Assignment 6: Factor and Data Frame in R language

R factors:

Factors are the data objects which are used to categorize the data and store it as levels. In order to categorize the data and store it on multiple levels, we use the data object called R factor. They are useful in the columns which have a limited number of unique values. Like "Male, "Female" and True, False etc. They are useful in data analysis for statistical modeling.

By default, R always sorts levels in alphabetical order.

The command used to create or modify a factor in R language is – factor() with a vector as input.

The two steps to creating a factor are:

1. Creating a vector
2. Converting the vector created into a factor using function factor()

```
# Creating a vector:
        >x=c("female", "male", "male", "female")
        >print(x)               #it prints "female" "male"   "male"   "female"

# Converting the vector x into a factor named gender
        >gender=factor(x)
        >print(gender)          #it prints      [1] female male   male   female
                                                Levels: female male
```

Levels can also be predefined by the programmer. For eample

```
# Creating a factor with levels defined by programmer:
>gender=factor(c("female", "male", "male", "female"), levels = c("female", "transgender", "male"));
```

Further one can check the levels of a factor by using function levels(). Function is.factor() is used to check whether the variable is a factor and returns “TRUE” if it is a factor.

```
        >gender=factor(c("female", "male", "male", "female"));
        >print(is.factor(gender))
```

Function class() is also used to check whether the variable is a factor and if true returns “factor”.

```
        >gender=factor(c("female", "male", "male", "female"));
        >class(gender)
```

Accessing elements of a Factor:

Like we acess elements of a vector, same way we acess the elements of a factor. If gender is a factor then gender[i] would mean acessing i th element in the factor.

Example:

```
        >gender =factor(c("female", "male", "male", "female"))
        >gender[4]                      #It prints   [1] female
                                                       Levels: female male
        >gender[c(2, 4)]                #It prints   [1] male   female
                                                     Levels: female male
```

For selecting all the elements of the factor gender except ith element, gender[-i] should be used.

>gender[-3]

Modification of a Factor

After a factor is formed, its components can be modified but the new values which needs to be assigned must be in the predefined level. For example

>gender[2]<-“female”

Data Frame in R:

The Data Frame in R is a table or two-dimensional data structure. In R Data Frames, data is stored in row and columns, and we can access the data frame elements using the row index and column index. A data frame is a list of variables, and it must contain the same number of rows with unique row names. The Column Names should not be Empty

The data frame’s data can be only of three types- factor, numeric and character type.

Data frame in R is created as follows

```
>Id=c(1:5)
>Name=c(“Nilesh”, “Suresh”, “Ramesh”, “Kamlesh”, “Rajesh”)
>Salary=c(80000, 70000, 90000, 50000, 60000)
>employee=data.frame(Id, Name,Salary)
> print(employee)       will print
        Id    Name Salary
1       1  Nilesh  80000
2       2  Suresh  70000
3       3  Ramesh  90000
4       4 Kamlesh  50000
5       5  Rajesh  60000
```

Create Named Data Frame in R:

```
# We are assigning new names to the Columns
>employee=data.frame("Empid" = Id, "Full_Name" = Name, "income" = Salary)
>print(employee)  will print
        Empid Full_Name income
1       1       Nilesh          80000
2       2       Suresh      70000
3       3       Ramesh     90000
4       4       Kamlesh    50000
5       5       Rajesh      60000

# Names function will display the Index Names of each Item >print(names(employee))
> print(names(employee))
```

[1] "Empid" "Full_Name" "income"

Access R Data Frame Elements:

In R programming, We can access the Data Frame item in multiple ways. In this example, we will show you how to access the data frame items using the index position. Using this index value, we can access each and every item present in the Data Frame. Index value starts at 1 and ends at n where n is the number of items in a data frame. For example

```
>Id=c(1:5)
>Name=c(“Nilesh”, “Suresh”, “Ramesh”, “Kamlesh”, “Rajesh”)
>Salary=c(80000, 70000, 90000, 50000, 60000)>
>employee=data.frame("Empid" = Id, "Full_Name" = Name, “Income" = Salary)
```

1. # Accessing all the Elements (Rows) Present in the Name Items (Column)
   ```
   >employee["Name"]
   ```
2. # Accessing all the Elements (Rows) Present in the 3rd Column (i.e., Occupation)
   ```
   >employee[3]
   ```
3. #Accessing Name column as vector
   ```
   > employee[[“Full_Name"]] or
   >employee[[2]]
   ```
4. # Accessing Element at 1st Row and 2nd Column
   ```
   >employee[1, 2]
   ```
5. # Accessing Element at 4th Row and 3rd Column
   ```
   >employee[4, 3]
   ```
6. # Accessing All Elements at 5th Row
   ```
   >employee[5, ]
   ```
7. # Accessing All Item of the 4th Column
   ```
   >employee[, 4]
   ```

Accessing Multiple Values from R Data:

1. # Accessing Item at 1st, 2nd Rows and 3rd, 4th Columns
   ```
   >employee[c(1, 2), c(3, 4)]
   ```
2. # Accessing Item at 2nd, 3rd, 4th Rows and 2nd, 4th Columns
   ```
   >employee[2:4, c(2, 4)]
   ```
3. # Accessing All Item at 2nd, 3rd, 4th, 5th Rows
   ```
   >employee[2:5, ]
   ```
4. # Accessing All Item of 2nd and 4th Column
   ```
   >employee[ ,c(2, 4)]
   ```
5. # Extract first three rows.
   ```
   >employee[(1:3), ]
   ```
6. #Adding column in data frame

> employee$dept=c(“Comp”, “Math”, “Ele”, “Stat”, “Eng.”)

Access R Data Frame Elements using $:

In R Programming, We can also access the Data frame elements using the **$** dollar symbol.

Ex. >employee$Empid

>employee$Full_Name

1. # Accessing Item at 2nd, 4th Rows of Full_Name Columns
   >employee$Full_Name[c(2, 4)]
2. # Accessing Item at 2nd, 3rd, 4th, 5th Rows of income Column
   >employee$income[2:5]
3. # Accessing Specific columns.
   >data.frame(employee$Full_Name, employee$income)

Modifying R Data Frame Elements:

In R programming, We can access the data frame elements using the index position. Using this index value, we can alter or change each and every individual element present in the data frame. For example

>Id=c(1:5)

>Name=c(“Nilesh”, “Suresh”, “Ramesh”, “Kamlesh”, “Rajesh”)

>Salary=c(80000, 70000, 90000, 50000, 60000)>

>employee=data.frame(Id, Name, Salary)

1. # Modifying Item at 2nd Row and 3rd Column
   >employee[2, 3] =100000
2. # Modifying All Item of 1st Column
   >employee[, 1]=c(10:15)

Adding Elements to Data Frame:

1. **cbind(Data Frame, Values):** This method is used to add extra Columns with values. In general, we prefer Vector as values parameter
   Ex. # Adding Extra Column
   >address=c(“Pimpri”, “Pune”, “Sangvi”, “Kalewadi”, “Nigdi”)
   >cbind(employee, address) where employee is a dataframe

2. **rbind(Data Frame, Values):** This method is used to add extra Row with values.
   Ex. # Adding Extra Row
   >rbind(employee, list(7, “Kamlesh", 8000))

Important Functions of Data Frame in R:

1. **typeof(Data Frame):** This method will tell you the type of Data Frame. Since the data frame is a kind of list, this function will return a list

2. **class(Data Frame):** This method will tell you the class of the Data Frame
3. **length(Data Frame):** This method will count the number of items (columns) in a Data Frame
4. **nrow(Data Frame):** This method will return the total number of Rows present in the Data Frame.
5. **ncol(Data Frame):** This method will return the total number of Columns available in the Data Frame.
6. **dim(Data Frame):** This method will return the total number of Rows and Columns present in the Data Frame.
7. **str(Data Frame):** This method returns the structure of the data present in the Data Frame.
8. **summary(Data Frame):** This R Programming method returns the nature of the data and the statistical summary such as Minimum, Median, Mean, Median, etc.

Use of Head and Tail Functions in R Data Frame:

1. **head(Data Frame, limit):** This method will return the top six elements (if you Omit the limit). If you specify the limit as 2 then, it will return the first 2 records. It is something like selecting the top 10 records.
2. **tail(Data Frame, limit):** This method will return the last six elements (if you Omit the limit). If you specify the limit as 4, it will return the last four records.

1. # No limit - It means Displaying First Six Records
   >head(emp)
2. # Limit is 4 - It means Displaying First Four Records
   >head(emp, 4)
3. # Limit is 10 - It means Displaying First Four Records
   >head(emp, 10)
4. # No limit - It means Displaying Last Six Records
   >tail(emp)
5. # Limit is 4 - It means Displaying Last four Records
   >tail(emp, 4)

Practice Programs:

1. Write a R program to find the levels of factor of a given vector.
2. Write a R program to create a data frame from four given vectors.
3. Write a R program to count the number of NA values in a data frame column.

SET A:

1. Write a R program to change the first level of a factor with another level of a given factor.

2. Write a R program to create a data frame from four given vectors and display the structure and statistical summary of a data frame.
3. Write a R program to display second row using row index and third column using column name of a data frame.
4. Write a R program to create a data frame using two given vectors and display the duplicated elements and unique rows of the said data frame.
5. Write a R program to call the (built-in) dataset airquality. Remove the variables 'Solar.R' and 'Wind' and display the data frame.

SET B:

1. Write an R program to concatenate two given factor in a single factor and display in descending order.
2. Write a R program to extract the five of the levels of factor created from a random sample from the LETTERS.
3. Write a R program to compare two data frames to find the row(s) in first data frame that are not present in second data frame.
4. Write a R program to create a data frame from four given vectors and perform the following
    a. add a new column in a given data frame
    b. add new row to data frame.
    c. drop specific column by name from a given data frame.
    d. drop row by number from a given data frame.
5. Write a R program to create a data frame from four given vectors and perform the following
    a. Extract 3rd and 5th rows with 1st and 3rd columns from a given data frame.
    b. Sort and display given data frame by specific column.

SET C:

1. Write a R program to create inner, outer, left, right join(merge) from given two data frames.
2. Write a R program to save the information of a data frame in a file and display the information of the file.

Assignment Evaluation

0: Not Done [ ]	**1: Incomplete [ ]**	**2: Late Complete [ ]**
3: Need Improvement [ ]	**4: Complete [ ]**	**5: Well Done [ ]**

Signature of Instructor

Assignment 7: Data Analysis

R CSV Files:

A Comma-Separated Values (CSV) file is a plain text file which contains a list of data. These files are often used for the exchange of data between different applications. These files can sometimes be called character-separated values or comma-delimited files. They often use the comma character to separate data. The idea is that we can export the complex data from one application to a CSV file, and then importing the data in that CSV file to another application. R allows us to read data from files which are stored outside the R environment. The file should be present in the current working directory so that R can read it. We can also set our directory and read file from there.

Getting and setting the working directory:

In R, getwd() and setwd() are the two useful functions.

- The getwd() function is used to check on which directory the R workspace is pointing.
- And the setwd() function is used to set a new working directory to read and write files from that directory.

1. # Getting and printing current working directory.
   ```
   >print(getwd())
   ```
2. # Setting the current working directory.
   ```
   >setwd("C:/Users/ajeet")
   ```

Creating a CSV File:

A text file in which a comma separates the value in a column is known as a CSV file.

Let's start by creating a CSV file with the help of the data from ms excel, which is mentioned below and by saving the file (save as) with .csv extension

```
id,name,marks
1,Nilesh,80.67                    #student.csv file
2,Shubham,90
3,Rajesh,65
```

Reading a CSV file:

R provides read.csv() function, which allows us to read a CSV file available in our current working directory.

Syntax: *data=read.csv(file, header = , sep = , quote =)*

some of the most useful arguments in read csv function:

1. **file:** You have to specify the file name, or Full path along with file name.
2. **header:** If the csv contains Columns names as the First Row then please specify TRUE otherwise, FALSE
3. **sep:** It is a short form of separator. You have to specify the character that is separating the fields. **" , "** means data is separated by comma

4. **quote:** If your character values (FirstName, Education column tc) are enclosed in quotes then you have to specify the quote type. For double quotes we use: quote = **"\""** in r read.csv function
5. **nrows:** It is an integer value. You can use this argument to restrict the number of rows to read. For example, if you want top 5 records, use nrows = 5
6. **skip:** Please specify the number of rows you want to skip from file before beginning the csv read. For example, if you want to skip top 2 records, use skip = 2

R Read csv File from Current Working Directory:

In this example, we will show you, How to read data from the csv (comma separated values) file that is present in the current working directory in R Programming.

```
>setwd("E:\R")
>data <- read.csv("student.csv")
>print(data)
```

Accessing csv file Data:

- In R programming, read.csv function will automatically convert the data into Data Frame.
- So, all the functions that are supported by the Data Frame can be used on csv data.
- While we are working with csv files or read from csv files in R programming, the following functions are the common functions used for data analysis.

1. **max:** This method will return the maximum value within the column
2. **min:** This method will return the minimum value within the column
3. **subset(data, condition):** This method will return the subset of data, and the data depends on the condition.

Examples:

1. # Creating a data frame by reading csv file
   ```
   >csv_data=read.csv("student.csv")
   ```
2. #Accessing all the Elements (Rows) Present in the marks Column
   ```
   >print(csv_data$marks)
   ```
3. # Getting the maximum marks from data frame.
   ```
   >Max_Marks=max(csv_data$marks)
   >print(Max_Marks)
   ```
4. # Accessing Element at 4th Row and 3rd Column
   ```
   >print(csv_data[4, 3] )
   ```
5. # Accessing Item at 1st, 2nd 4th Rows and 4th, 5th, 6th, 7th Columns
   ```
   >csv_data [c(1, 2, 4), c(4:7)]
   ```
6. #Getting the details of the Student who score minimum marks
   ```
   >Details=subset(csv_data, marks==min(marks))
   >print(Details)
   ```

7. #Getting the details of all the students whose name is Nilesh
   ```
   >details=subset(csv_data,name==“Nilesh")
   >print(details)
   ```
8. #Getting the details of all the student whose name is Nilesh and rollno is 5
   ```
   >details=subset(csv_data,rollno==5 & name==“Nilesh”)
   ```
9. #Getting the details of all the students who score more than 60 marks
   ```
   > details=subset(csv_data,marks>60)
   >print(details)
   ```
10. #using inbuilt dataset mtcars find the number of cars of each gear type
    ```
    >data=mtcars
    >f=factor(data$gear)
    >print(table(f))
    ```

11. #using inbuilt dataset mtcars find the number of cars having 3 gear and 2 carburetor
    ```
    >data=mtcars
    >d=subset(data,gear==3 & carb==2)
    >print(nrow(d))
    ```

Data Frame and dplyr package:

The data frame is a key data structure in statistics and in R. dplyr package is very very helpful for managing data frames. The dplyr package was developed by Hadley Wickham of RStudio and is an optimized and distilled version of his plyr package. The dplyr package does not provide any “new” functionality to R, in the sense that everything dplyr does could already be done with base R, but it greatly simplifies existing functionality in R. Filtering, re-ordering, and collapsing, can often be tedious operations in R whose syntax is not very intuitive. The dplyr package is designed to mitigate a lot of these problems and to provide a highly optimized set of routines specifically for dealing with data frames.

Installing the dplyr package:

```
>install.packages("dplyr")
```

After installing the package it is important that you load it into your R session with the library() function.

```
>library(dplyr)
```

Some of the key functions provided by the dplyr package are:

1. **select:** Select columns with select(). It returns a subset of the columns of a data frame.

   Ex.
   ```
   df=iris
   x<-select(df,c(Species,Sepal.Length))
   head(x)
   ```

2. **filter:** Filter rows with filter().It extracts a subset of rows from a data frame based on logical conditions.

   Ex x<-filter(iris, Sepal.Length > 5.843)
   head(x)

3. **arrange:** Arrange rows with arrange(). It helps to reorder rows of a data frame
   Ex x<-arrange(mtcars, cyl)
   Print(x)
   y<-arrange(mtcars, desc(cyl))
   print(y)
4. **group_by:**
   The group_by() function first sets up how you want to group your data.
   The general operation here is a combination of splitting a data frame into separate pieces defined by a variable or group of variables (group_by()), and then applying a summary function across those subsets (summarize()).
   Ex cyl <- group_by(mtcars, cyl)
   summarise(cyl, mean(disp), mean(hp))

Practice Programs:

1. Using inbuilt dataset women perform the following
   a. display all rows of dataset having height greater than 120
   b. display all rows of dataset in ascending order of weight
2. Using the inbuilt mtcar dataset perform the following
   a. Display all the cars having 4 gears
   b. Display all the cars having 3 gears and 2 carburetor.
3. Using inbuilt PlantGrowth dataset perform the following
   a. Find the flowers of each type of group
   b. Display all rows of type “ctrl” having weight greater than 5.0

SET A:

1. Using the inbuilt mtcar dataset perform the following
   a. Display all the cars having mpg more than 20
   b. Subset the dataset by mpg column for values greater than 15.0.
2. Using the inbuilt airquality dataset perform the following
   a. Find the temperature of day 30 of month 8
   b. Display the details of all the days if the temperature is greater than 90
3. Using the inbuilt airquality dataset perform the following
   a. Subset the dataset for the month July having Wind value greater than 10
   b. Find the number of days having temperature less than 60

SET B:

1. Using iris inbuilt dataset perform the following
   a. Find the flowers of each type of species

b. Find the Sepal length and width of the flower of type setosa having maximum petal length

2. Using iris inbuilt dataset perform the following
    a. Display details of all flowers of type virginica in ascending order of petal length. (use order function)
    b. Display details of first five flowers of type setosa having maximum petal length.
3. Using inbuilt PlantGrowth dataset perform the following
    a. Display details of all plant having weigth greater than 5.80
    b. Display details of all Plants of group trt1 in ascending order of their weight.

SET C:

1. Using inbuilt ToothGrowth dataset perform the following
    a. Find supplement (supp) wise maximum and minimum length of tooth
    b. Display details of first 3 tooth having minimum length for supplement OJ for dose 1.0

Assignment Evaluation

0: Not Done [ ]	**1: Incomplete [ ]**	**2: Late Complete [ ]**
3: Need Improvement [ ]	**4: Complete [ ]**	**5: Well Done [ ]**

Signature of Instructor

Assignment 8: Data Visualization

Introduction:

Data visualization is an efficient technique for gaining insight about data through a visual medium. With the help of visualization techniques, a human can easily obtain information about hidden patterns in data that might be neglected.

By using the data visualization technique, we can work with large datasets to efficiently obtain key insights about it.

In R, we can create visually appealing data visualizations by writing few lines of code.

Advantages of Data Visualization in R:

1. **Understanding:** It is easier to understand through graphics and charts than a written document with text and numbers. Thus, it can attract a wider range of audiences. Also, it promotes the widespread use of business insights that come to make better decisions.
2. **Efficiency:** Its applications allow us to display a lot of information in a small space. Although, the decision-making process in business is inherently complex and multifunctional, displaying evaluation findings in a graph can allow companies to organize a lot of interrelated information in useful ways.

R Bar Charts:

A bar chart is a pictorial representation in which numerical values of variables are represented by length or height of lines or rectangles of equal width. A bar chart is used for summarizing a set of categorical data. In bar chart, the data is shown through rectangular bars having the length of the bar proportional to the value of the variable.In R, we can create a bar chart to visualize the data in an efficient manner.

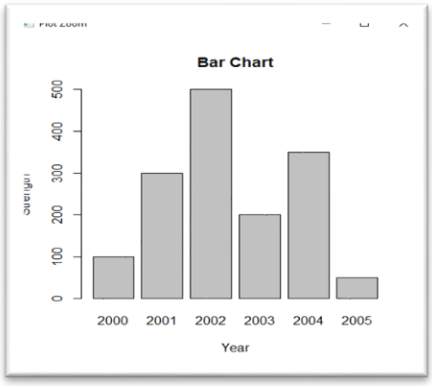
For this purpose, R provides the barplot() function, which has the following syntax:

Syntax: barplot(h, xlab, ylab, main, names.arg, col) where

1. **h:** A vector or matrix which contains numeric values used in the bar chart.
2. **xlab:** A label for the x-axis.
3. **ylab:** A label for the y-axis.
4. **main:** A title of the bar chart.
5. **names.arg:** A vector of names that appear under each bar.
6. **Col:** It is used to give colors to the bars in the graph.

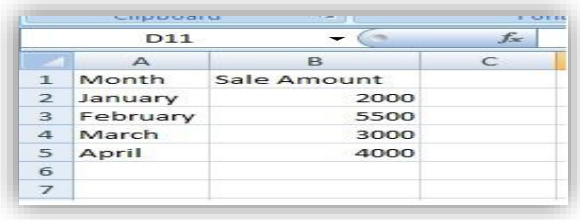
Example: # Creating the data for Bar chart

```
> h=c(100,300,500,200,350,50)
> barplot(h, xlab="Year", ylab="Strength", main="Bar Chart", names.arg = c (2000, 2001, 2002,2003,2004,2005))
```



Creating a barplot in R by reading data from CSV file:

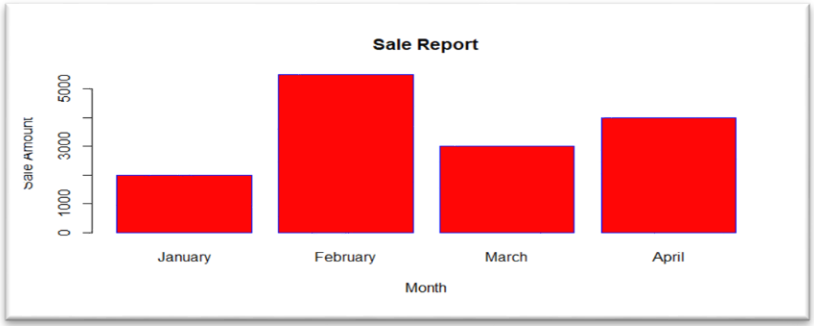
```
>setwd("F:/SYBBA/R/")                          #setting the path where csv file is stored
>d=read.csv("SaleReport.csv", header=TRUE)      #reading csv file data in data frame d
```



	A	B	C
1	Month	Sale Amount	
2	January	2000	
3	February	5500	
4	March	3000	
5	April	4000	
6			
7			

SaleReport.csv File

```
>print(d)                                      #printing data frame
>barplot(d$Sale.Amount, xlab="Month", ylab="Sale Amount", main="Sale Report", names.arg=
d$Month, col="red“, border="blue")
```



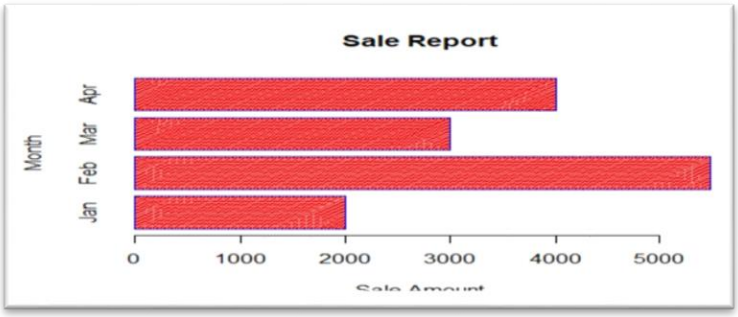
Horizontal Bar Chart in R Programming:

In this example, we change the default vertical bar chart into a horizontal bar chart using ***horiz*** argument in R.

We also change the bar density using ***density*** argument in R barplot

```
>setwd("F:/Yogesh/R/")
>getwd()
>d=read.csv("SaleReport.csv",header=TRUE)
>print(d)
```

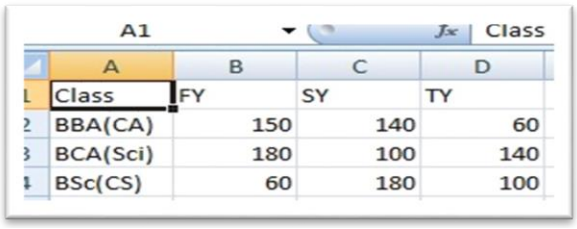
>barplot(d$Sale.Amount, xlab="Month", ylab="Sale Amount", main="Sale Report", names.arg=d$Month, col="red", border="blue", horiz = TRUE, density=100)



Creating Stacked Bar Plot using Matrix:

We can create bar charts with groups of bars and stacks using matrices as input values in each bar. One or more variables are represented as a matrix that is used to construct group bar charts and stacked bar charts.

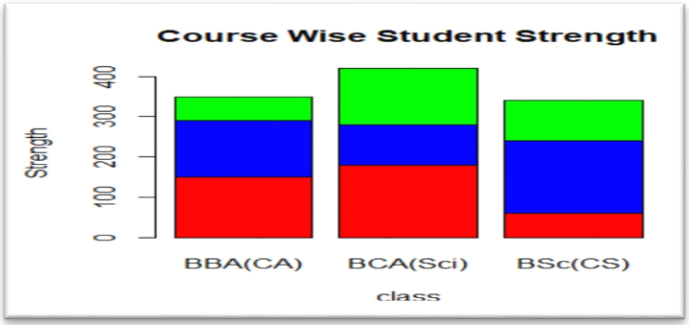
>setwd("F:/Yogesh/R/")
>d=read.csv("class.csv",header=TRUE)
>print(d)



	A	B	C	D
1	Class	FY	SY	TY
2	BBA(CA)	150	140	60
3	BCA(Sci)	180	100	140
4	BSc(CS)	60	180	100

Class.csv File

>fy=d$FY
>sy=d$SY
>ty=d$TY
>data=matrix(c(fy,sy,ty),ncol=3,byrow=TRUE)
>barplot(data, xlab="class", ylab="Strength", main="Course Wise Student Strength", names.arg=d$Class, col= c ("red", "blue", "green"))



R Scatterplots:

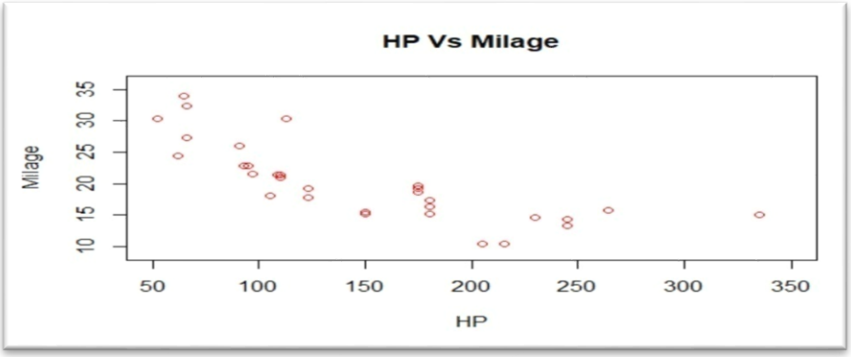
In a scatterplot, the data is represented as a collection of points. Each point on the scatterplot defines the values of the two variables. One variable is selected for the vertical axis and other for the horizontal axis.

The scatter plots are used to compare variables. A comparison between variables is required when we need to define how much one variable is affected by another variable.
Scatterplot is created using plot() function. The syntax is as follows

Syntax: plot(x, y, main, xlab, ylab, xlim, ylim, axes) where,

1. **X**- It is the dataset whose values are the horizontal coordinates.
2. **Y**- It is the dataset whose values are the vertical coordinates.
3. **Main**- It is the title of the graph.
4. **Xlab**- It is the label on the horizontal axis.
5. **Ylab**- It is the label on the vertical axis.
6. **Xlim**- It is the limits of the x values which is used for plotting.
7. **Ylim**- It is the limits of the values of y, which is used for plotting.
8. **axes**- It indicates whether both axes should be drawn on the plot.

Example:
Following scatterplot show the relationship between HP and MPG attribute of mtcars dataset
>plot(mtcars$hp, mtcars$mpg, xlab="HP", ylab="Milage", xlim=c(50,350), ylim=c(9,36), main= "HP Vs Milage", col="red")



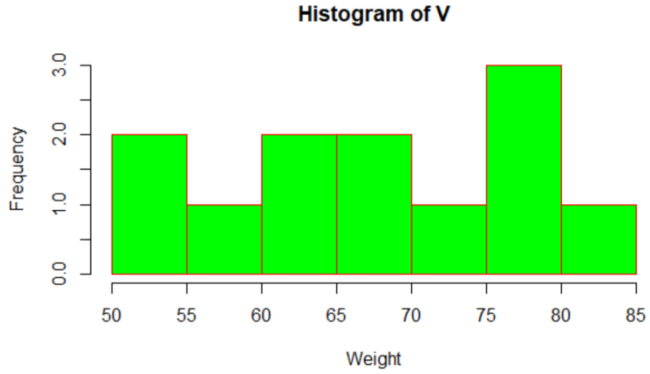
R Histogram:
A histogram is a type of bar chart which shows the frequency of the number of values which are compared with a set of values ranges. The histogram is used for the distribution, whereas a bar chart is used for comparing different entities.
In the histogram, each bar represents the height of the number of values present in the given range.
For creating a histogram, R provides hist() function, which takes a vector as an input and uses more parameters to add more functionality.
There is the following syntax of hist() function:

Syntax: hist(v, main, xlab, ylab, xlim, ylim, breaks, col, border) where

1. **V:** It is a vector that contains numeric values.
2. **Main:** It indicates the title of the chart.
3. **Col:** It is used to set the color of the bars.
4. **Border:** It is used to set the border color of each bar.
5. **Xlab:** It is used to describe the x-axis.
6. **Ylab:** It is used to describe the y-axis.
7. **Xlim:** It is used to specify the range of values on the x-axis.
8. **Ylim:** It is used to specify the range of values on the y-axis.
9. **Breaks:** It is used to mention the width of each bar.

Example: Consider Vector V which consists of weight of different students

```
> V=c(55,67,78,82,57,62,74,80,52,64,76,66)
>hist(v, xlab = "Weight", ylab="Frequency", col = "green", border = "red")
```



R Boxplot:

Boxplots are a measure of how well data is distributed across a data set. This divides the data set into three quartiles. This graph represents the minimum, maximum, average, first quartile, and the third quartile in the data set. Boxplot is also useful in comparing the distribution of data in a data set by drawing a boxplot for each of them.

R provides a boxplot() function to create a boxplot. There is the following syntax of boxplot() function

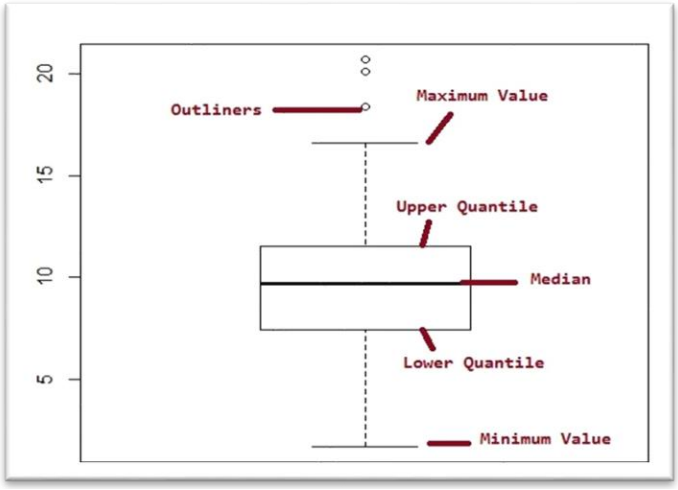
Syntax: boxplot(data or formula, xlab, ylab, main, names, col) where,

1. **data:** DataFrame, or List that contains the data to draw boxplot.
2. **Xlab:** It is used to describe the x-axis.
3. **Ylab:** It is used to describe the y-axis.
4. **Main:** It is used to give a title to the graph.
5. **Names:** It is the group of labels that will be printed under each boxplot.

Creating a Boxplot in R Programming:

In this example, we create a Boxplot using the ***airquality*** data set

```
>a=airquality
> boxplot(a$Wind)
```



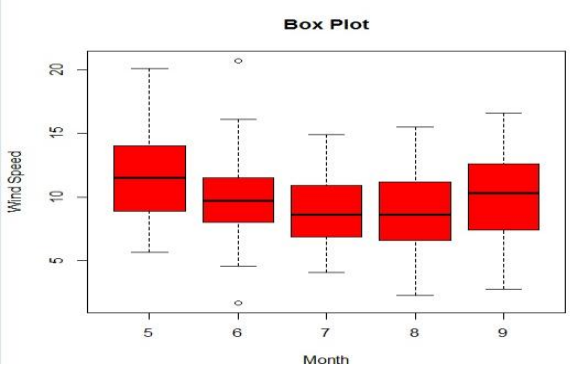
Use Formula to create a Boxplot in R:

In this example, we create a Boxplot using the formula argument

formula: It should be something like ***value~group,*** where value is the vector of numeric values, and the group is the column you want to use as a group by.

e.g., if you want to draw a boxplot for Monthwise wind speed, then ***value*** = Wind and ***group*** = Month

```
>a=airquality
>boxplot(a$Wind~a$Month, xlab="Month", ylab="Wind Speed", main="Box Plot",
col="red")
```



Practice Programs:

1. Write an R program to draw an empty plot and an empty plot specifies the axes limits of the graphic.
2. Using inbuilt airquality dataset make a scatter plot to compare Wind speed and temperature.
3. Using inbuilt iris dataset create Histogram for Petal.length values
4. Using iris dataset draw horizontal bar plot for Petal length values for species setosa

SET A:

1. Using inbuilt mtcars dataset
    a) Create a bar plot for attribute mpg for all cars having 3 gears
    b) Create a Histogram to show number of cars per carburetor type whose mpg is greater than 20
2. Using airquality dataset
    a) Create a scatter plot to show the relationship between ozone and wind values by giving appropriate value to color argument
    b) Create a bar plot to show the ozone level for all the days having temperature greater than 70

SET B:

1. Using inbuilt mtcars dataset
    a. Create a bar plot that shows the number of cars of each gear type.

   b. Draw a scatter plot showing the relationship between wt and mpg for all the cars having 4 gears
2. Using airquality dataset
   a. Show the statistical summary using box plot for Temprature value of month June
   b. Using histogram show the frequency of number of days for Temp values of month August

SET C:

1. Using inbuilt mtcars dataset show a stacked bar graph of the number of each gear type and how they are further divided out by cyl
2. Draw boxplot to show the distribution of mpg values per number of gears

Assignment Evaluation

0: Not Done [ ]	**1: Incomplete [ ]**	**2: Late Complete [ ]**
3: Need Improvement [ ]	**4: Complete [ ]**	**5: Well Done [ ]**

Signature of Instructor

Section-V

Block Chain

Assignment No 1: Working with Blockchain

Blocks are like ledger pages while the whole record-keeping book is the **blockchain**. A block is a file that stores unalterable data related to the network. It hold all the records of valid cryptocurrency transactions. They are hashed and encoded into a hash tree or Merkle tree. Every block has the cryptographic hash of the block that precedes it in the blockchain**.**

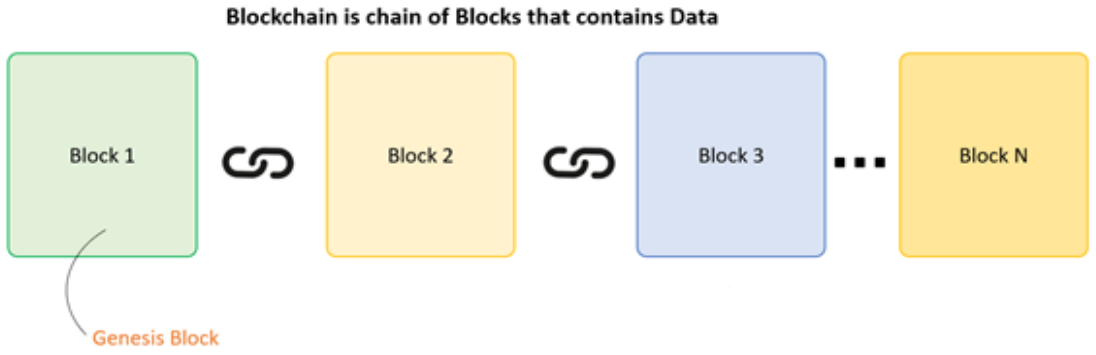
A block, plus all the blocks before and after it, form a block chain.

The first block in a blockchain is the **Genesis Block**. It is the only one with no data on the previous block because there is no block before it.

Blockchain

A blockchain is a continuously growing list of records, called blocks, which are linked and secured using cryptograph.

For use as a distributed ledger a blockchain is typically managed by a peer-to-peer network collectively adhering to a protocol for validating new blocks. The data recorded in any given block cannot be altered without the alteration of all subsequent blocks and a collusion of the network majority.



The structure of blockchain program:

Step1: Import SHA256 Hash Function
Step2: Create a Block class
{ //Consructor
//Hash function
}
Step 3:Create a Bslockchain Class
{ //Constructor
//function to create genesis block
//function to get latest block
//function to add block
}
Step 4:
Display blockchain information

1: Importing SHA256 hash function:

For hashing we are using **SHA256 hash function**. SHA256 hash is not available in JavaScript and requires you to use an external library. **crypto-js** library contains secure implementations for different hash functions. To **install crypto-js** using following command on command prompt i.e. **Terminal**

```
npm install --save crypto-js  (Node Package Manager)
```

You will then see the following **output**:

```
npm WARN saveError ENOENT: no such file or directory, open
'/Users/spenserhuang/Desktop/js-blockchain/package.json'
npm WARN enoent ENOENT: no such file or directory, open
'/Users/spenserhuang/Desktop/js-blockchain/package.json'
npm WARN js-blockchain No description
npm WARN js-blockchain No repository field.
npm WARN js-blockchain No README data
npm WARN js-blockchain No license field.
+ crypto-js@3.1.9-1
updated 1 package in 1.75s
```

Now **crypto-js** is installed successfully.

Then afterwards we can import it in our **".js"** file using following statement.

```
const SHA256 = require("crypto-js/sha256");
```

2: Creating a Block class: create a basic block structure using Block class:

When you create a new block, you need to pass it a timestamp, some data and the hash of the block that went before it:

```
class Block
{
constructor(index,timestamp, data, previousHash = '')
 {
this.index=index;
this.timestamp = timestamp;
this.data = data;
this.previousHash = previousHash;
this.hash=this.calculateHash()
}
}
```

Here's what each property means:

- ✓ **Timestamp**: tells us when the block was created. You can use any format you like (in this example we'll use a UNIX timestamp)

- ✓ **Data:** parameter can include any type of data that you want to associate with this block. If you want to build a cryptocurrency you can store transaction details in here like sender/receiver and the amount of money that was transferred.
- ✓ **previousHash:** is a string that contains the hash of the previous block. This is what will create the chain of blocks and will be very important to ensure the integrity of our blockchain later.
- ✓ **Hashes:** Each block points towards the previous block (that's why we have the **previousHash** attribute). That means that each block needs a hash.

3: Calculating hash using function:

Now that we have our **calculateHash()** method, let's use it in the constructor of our Block:

```
calculateHash()
{
return
SHA256(this.index+this.previous+this.timestamp+JSON.stringify(this.data)).toString();
}
```

4: Creating a Blockchain class:

This defines the structure of a Blockchain should look like. So let's create a new class for that:

```
class Blockchain
{
constructor()
 {
this.chain = [this.createGenesisBlock()];
}
```

In this case, the blockchain is a very simple object that contains a property chain. This is an array containing all the blocks on the chain. Before we can add new blocks, we have to add a "genesis block". This is the first block on the chain and it's a bit special because it cannot point to a previous block.

5: Function for creating Genesis Block:

The following method is used to create a Genesis Block

```
createGenesisBlock()
{
return new Block(0,"01/01/2017", "Genesis block", "0");
}
```

Back in the constructor of our Blockchain class, we can now add the genesis block Whenever we create a new Blockchain instance:

6: Function for getting latest block

```
getLatestBlock(){
```

```
return this.chain[this.chain.length - 1];
}
```

7. Function for adding block: The **addBlock** method is a bit more complicated. Before we can add a new block to our chain, we have to correctly set the **previousHash** property of that block. It has to be set to the hash of the latest block on our chain. And we also have to calculate the hash of the new block:

```
addBlock(newBlock){
newBlock.previousHash=this.getLatestBlock().hash;
newBlock.hash=newBlock.calculateHash();
this.chain.push(newBlock);
}
```

8. Display block information: In reality, adding blocks to a blockchain requires you to "mine" it.

```
let savjeeCoin = new Blockchain();
```

And let's add a few blocks:

```
savjeeCoin.addBlock(new Block("20/07/2017", { amount: 4 }));
savjeeCoin.addBlock(new Block("22/07/2017", { amount: 10 }));
```

There — we have created two new blocks. Let's see what our blockchain now looks like. We will stringifySavjeeCoin and use four spaces to format it:

```
console.log(JSON.stringify(savjeeCoin, null, 4));
```

Note: After combining all the above steps the final program will look like following:

Program name: main.js

```
const SHA256 = require("crypto-js/sha256");
class Block
{
constructor(index,timestamp, data, previousHash = '')
{
this.index=index;
this.timestamp = timestamp;
this.data = data;
this.previousHash = previousHash;
this.hash=this.calculateHash()
}
calculateHash()
{
```

```
return
SHA256(this.index+this.previous+this.timestamp+JSON.stringify(this.data)).toString();
}
}

class Blockchain
{
constructor()
{
this.chain = [this.createGenesisBlock()];
}
createGenesisBlock()
{
return new Block(0,"01/01/2017", "Genesis block", "0");

}
getLatestBlock()
{
return this.chain[this.chain.length - 1];
}
addBlock(newBlock)
{
newBlock.previousHash=this.getLatestBlock().hash;
newBlock.hash=newBlock.calculateHash();
this.chain.push(newBlock);
}
}
let savjeeCoin = new Blockchain();
savjeeCoin.addBlock(new Block(1,"20/07/2017", { amount: 4 }));
savjeeCoin.addBlock(new Block(2,"22/07/2017", { amount: 10 }));
console.log(JSON.stringify(savjeeCoin, null, 4));
```

Building A Small Blockchain Application :

1. Installation:

- Download the **Visual Studio Code installer** for Windows.
- Install VS Code and install additional component latest version of **Node.js**

2. Write a code for Building Blocks:

Step 1: Create a folder on any drive where you can save your blockchain program

Step 2: Open Visual Studio Code

Step 3: Select **File → Open Folder** (Select your Blockchain program folder)

Step 4: In the **EXPLORER** section your folder is displayed. There are 4 icons near to your Folder Name. Click on **new file icon and give a file name with ".js" extension**

Step 5: Write your program in the ".js" file

Step 6: Select **Run→Run without Debugging→Select Environment→Node.js** And you will get the output

OR

Step 6:

1. **Select Terminal→New Terminal**
   (a new terminal is open with our folder Eg: D:\Blockchain>**)**

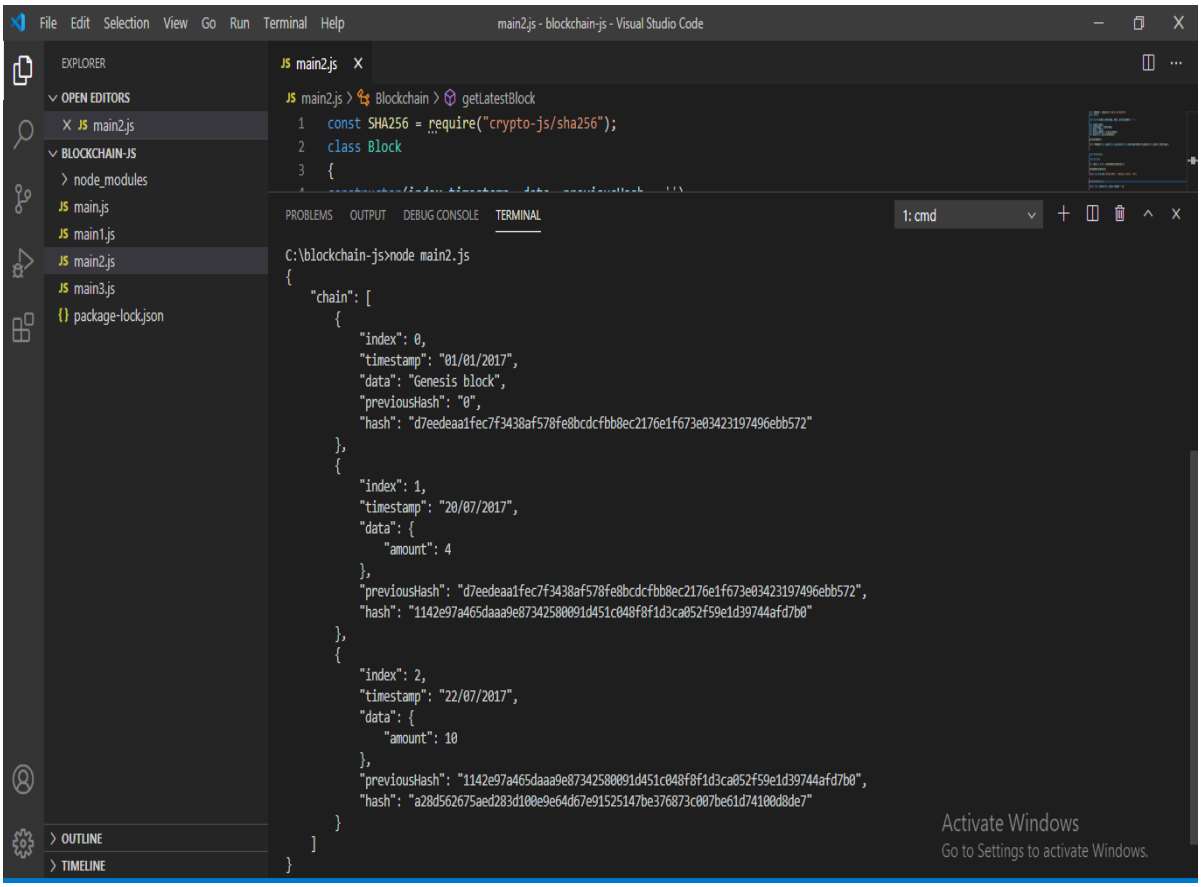
2. **Type following command**
   D:\Blockchain>**npm install --save crypto-js**

3. **Execute the program using following command**

D:\Blockchain>**node FileName.js (Press Enter)**

D:\Blockchain->node Demo.js

- **The output is what our blockchain looks like:**



Output of Blockchain

Solved Programs

1. Write a blockchain application in JavaScript for the creation of Transaction block for the account holder.

```
const SHA256 = require("crypto-js/sha256");
class Block
{
	constructor(index,timestamp,transaction, data, previousHash = '')
	{	this.index=index;
		this.timestamp = timestamp;
		this.transaction=transaction;
		this.data = data;
		this.previousHash = previousHash;
		this.hash=this.calculateHash()
	}
	calculateHash()
	{
    return
SHA256(this.index+this.previous+this.timestamp+JSON.stringify(this.data)).toString();
	}
}

classBlockchain
{
	constructor()
	{ this.chain = [this.createGenesisBlock()];}

	createGenesisBlock()
	{
		return new Block(0,"01/06/2020", "Genesis block", "0");

	}
	getLatestBlock(){
		returnthis.chain[this.chain.length - 1];
	}
	addBlock(newBlock)
	{
		newBlock.previousHash=this.getLatestBlock().hash;
		newBlock.hash=newBlock.calculateHash();
		this.chain.push(newBlock);
	}
}
class Transaction
{
	constructor(Ac_name, Address, Acc_type)
```

```
        {       this.Ac_name = Ac_name;
                this.Address = Address;
                this.Acc_type = Acc_type;
            }
createTransaction(transaction){
this.pendingTransactions.push(transaction);
    }
 }

letsavjeeCoin = new Blockchain();
savjeeCoin.addBlock(new Block(1,"20/06/2020", { Ac_name: "Mr.Shivaji", Address:"Bhor",
Acc_type: "Seving"}));
savjeeCoin.addBlock(new Block(1,"20/06/2020", { Ac_name: "Mahi", Address:"Natambi",
Acc_type: "Current"}));
console.log(JSON.stringify(savjeeCoin, null, 4));
```

Output:

```
C:\blockchain-js>node transaction_saving.js
{
   "chain": [
      {
         "index": 0,
      "timestamp": "01/06/2020",
         "transaction": "Genesis block",
         "data": "0",
         "previousHash": "",
         "hash": "ae3c2415351a43b9cca02321d379b0d8ea0287f230cf019010914a1a150d7786"
      },
      {
         "index": 1,
      "timestamp": "20/06/2020",
         "transaction": {
            "Ac_name": "Mr.Shivaji",
            "Address": "Bhor",
            "Acc_type": "Seving"
         },
         "previousHash":
"ae3c2415351a43b9cca02321d379b0d8ea0287f230cf019010914a1a150d7786",
         "hash": "917fa664664b36b0521f35e7b254b929743536227714b241db2e3f2709c800d5"
      },
      {
         "index": 1,
       "timestamp": "20/06/2020",
         "transaction": {
```

```
            "Ac_name": "Mahi",
            "Address": "Natambi",
            "Acc_type": "Current"
        },
        "previousHash":
"917fa664664b36b0521f35e7b254b929743536227714b241db2e3f2709c800d5",
        "hash": "917fa664664b36b0521f35e7b254b929743536227714b241db2e3f2709c800d5"
    }
  ]}
```


2. Write a blockchain application in JavaScript to calculate hash code for the transaction.

```
const SHA256 = require("crypto-js/sha256");
class Block
{
        constructor(index,timestamp,transaction, data, previousHash = '')
        {
                this.index=index;
                this.timestamp = timestamp;
                this.transaction=transaction;
                this.data = data;
                this.previousHash = previousHash;
                this.hash=this.calculateHash()
        }
        calculateHash()
        {
        return
        SHA256(this.index+this.previous+this.timestamp+JSON.stringify(this.data)).toString();
        }
}

class Blockchain
{
constructor()
        {
                this.chain = [this.createGenesisBlock()];
        }
        createGenesisBlock()
        {
                return new Block(0,"01/06/2020", "Genesis block", "0");

        }
        getLatestBlock()
        {
                returnthis.chain[this.chain.length - 1];
        }
        addBlock(newBlock)
```

```
        {
                newBlock.previousHash=this.getLatestBlock().hash;
                newBlock.hash=newBlock.calculateHash();
                this.chain.push(newBlock);
        }
}
class Transaction
{
        constructor(fromAddress, toAddress, amount)
        {
                this.fromAddress = fromAddress;
                this.toAddress = toAddress;
                this.amount = amount;
           }
        createTransaction(transaction){
                this.pendingTransactions.push(transaction);
            }
 }

letsavjeeCoin = new Blockchain();
savjeeCoin.addBlock(new Block(1,"20/06/2020", { fromAdeess: "pune", toAddress: "Bhor" ,
amount: 4 }));
savjeeCoin.addBlock(new Block(2,"30/06/2020", { fromAdeess: "Satara", toAddress: "Bhor"
,amount: 4 }));
console.log(JSON.stringify(savjeeCoin, null, 4));
```

Output: C:\blockchain-js>node Transaction.1.js


```
{
   "chain": [
      {
         "index": 0,
      "timestamp": "01/06/2020",
         "transaction": "Genesis block",
         "data": "0",
         "previousHash": "",
         "hash": "ae3c2415351a43b9cca02321d379b0d8ea0287f230cf019010914a1a150d7786"
      },
      {
         "index": 1,
         "timestamp": "20/06/2020",
         "transaction": {
            "fromAdeess": "pune",
            "toAddress": "Bhor",
            "amount": 4
```

```
        },
        "previousHash":
"ae3c2415351a43b9cca02321d379b0d8ea0287f230cf019010914a1a150d7786",
        "hash": "917fa664664b36b0521f35e7b254b929743536227714b241db2e3f2709c800d5"
      },
      {
        "index": 2,
        "timestamp": "30/06/2020",
        "transaction": {
          "fromAdeess": "Satara",
          "toAddress": "Bhor",
          "amount": 4
        },
        "previousHash":
"917fa664664b36b0521f35e7b254b929743536227714b241db2e3f2709c800d5",
        "hash": "335dfad8d59a2cbae099ea6593f2a05784622c5a39c15d9d1756914319f2a1ef"
      }
   ]
}
```


3. Write a JavaScript code for the implementation of block chain technology.(At least two block).

```
const SHA256 = require("crypto-js/sha256");
class Block
{
        constructor(index,timestamp, data, previousHash = '')
        {
                this.index=index;
                this.timestamp = timestamp;
                this.data = data;
                this.previousHash = previousHash;
                this.hash=this.calculateHash()
        }
        calculateHash()
        {
        return
        SHA256(this.index+this.previous+this.timestamp+JSON.stringify(this.data)).toString();
        }
}

classBlockchain
{
        constructor()
        {
                this.chain = [this.createGenesisBlock()];
        }
```

```
createGenesisBlock()
{
        return new Block(0,"01/06/2020", "Genesis block", "0");
}
getLatestBlock()
{
        returnthis.chain[this.chain.length - 1];
}
addBlock(newBlock)
{
        newBlock.previousHash=this.getLatestBlock().hash;
        newBlock.hash=newBlock.calculateHash();
        this.chain.push(newBlock);
}  }
letsavjeeCoin = new Blockchain();
savjeeCoin.addBlock(new Block(1,"20/06/2020", { amount: 4 }));
savjeeCoin.addBlock(new Block(2,"22/06/2020", { amount: 10 }));
console.log(JSON.stringify(savjeeCoin, null, 4));
```

Output:

```
C:\blockchain-js>node main2.js
{
   "chain": [
      {
         "index": 0,
         "timestamp": "01/06/2020",
         "data": "Genesis block",
         "previousHash": "0",
         "hash": "d7eedeaa1fec7f3438af578fe8bcdcfbb8ec2176e1f673e03423197496ebb572"
      },
      {
         "index": 1,
      "timestamp": "20/06/2020",
         "data": {
            "amount": 4
         },
         "previousHash":
"d7eedeaa1fec7f3438af578fe8bcdcfbb8ec2176e1f673e03423197496ebb572",
         "hash": "1142e97a465daaa9e87342580091d451c048f8f1d3ca052f59e1d39744afd7b0"
      },
      {
         "index": 2,
      "timestamp": "22/06/2020",
         "data": {
            "amount": 10
```

```
        },
        "previousHash":
"1142e97a465daaa9e87342580091d451c048f8f1d3ca052f59e1d39744afd7b0",
        "hash": "a28d562675aed283d100e9e64d67e91525147be376873c007be61d74100d8de7"
      }
   ]
}

C:\blockchain-js>
```


Practice Programs:

1. Write a blockchain application in JavaScript to create bitcoin wallet.
2. Write a blockchain application in JavaScript for the simple transaction.
3. Write a blockchain application in JavaScript for the implementation of SHA256() function.
4. Write a decentralized block chain application in JavaScript to calculate fine for the students those have submitted books late.

SET A:

1. Write a blockchain application in JavaScript for the creation of Transaction block for the account holder.
2. Write a blockchain application in JavaScript to calculate hash code for the transaction.
3. Write a JavaScript code for the implementation of block chain technology.(At least two block).

SET B:

1. Write a blockchain application in JavaScript for the simple transaction.
2. Write a decentralized block chain application in JavaScript for the bank transaction
3. Write a blockchain application in JavaScript to transfer cryptocurrency from one account to another account.

SET C:

1. Write a decentralized block chain application in JavaScript for the bank transaction system.
2. Write a blockchain application in JavaScript for the generation of bitcoin after completion of transaction.

Assignment Evaluation

0: Not Done [ ]	**1: Incomplete [ ]**	**2: Late Complete [ ]**
3: Need Improvement [ ]	**4: Complete [ ]**	**5: Well Done [ ]**

Signature of Instructor

Assignment No 2: Implementation of Smart Contracts and Hyperledger

Solidity:
Solidity is an object-oriented programming language for writing smart contracts. It is used for implementing smart contracts on various blockchain platforms, most notably, Ethereum.

Smart Contract:
Smart contracts help you exchange money, property, shares, or anything of value in a transparent, conflict-free way while avoiding the services of a middleman. They showed the world how the blockchain can evolve from a simple payment mechanism to something far more meaningful and powerful. Smart contracts are automated contracts. They are self-executing with specific instructions written in its code which get executed when certain conditions are made.

Smart contracts are how things get done in the Ethereum ecosystem. When someone wants to get a particular task done in Ethereum they initiate a smart contract with one or more people. Every single transaction that you are doing, the smart contracts will get recorded and updated by the network. It keeps everyone involved with the contract responsible for their actions.

Smart contracts are implemented by using SOLIDITY Programming Language.
Solidity is a programming language , smart contracts are series of instructions that write in 'solidity'. Language is set of instruction if the first set of instructions are done then execute the next instruction and after that the next and keep on repeating until you reach the end of the contract. Each and every step that you take acts like a trigger for the next step to execute itself like a vending machine.
Some important point of solidity language.

- ✓ Solidity is a typed language; the type of each variable (state and local) needs to be specified at compile-time. Solidity provides several elementary types which can be combined to form complex types.
- ✓ Solidity is a contract-oriented, high-level language for implementing smart contracts. It was partially by C++, Python and JavaScript and is designed to target the Ethereum Virtual Machine (EVM).
- ✓ Solidity supports inheritance, libraries and complex user-defined types among other features.

1. Structure of solidity Program

A Solidity source files can contain any number of contract definitions, import directives and **pragma** directives.

```
pragma contract >=0.4.0<0.6.9
import "filename"
contract ContractName
{
```

```
<variable declaration>
<mappings>
<constructor>
<functions>
<modifiers>
}
```

- **Pragma:** The first line is a pragma directive which tells that the source code is written for Solidity version 0.4.0 or anything newer that does not break functionality up to, but not including, version 0.6.9. You also write this but it specific compile

  Pragma solidity ^0.4.0;
- **Contract:** A Solidity contract is a collection of code like function and data that resides at a specific address on the Ethereum blockchain.
- **Importing Files:** The following statement imports all global symbols from "filename".

  import "filename";
- **Variable declaration:** Variables are the names you give to computer memory locations
- **Mapping:** Mapping is a reference type as arrays and structs.
- **Constructor:** is a special kind of function. It uses to initialize the variables, data.
- **Function:** Functions are the executable units of code within a contract.
- **Function modifiers:** can be used to amend the semantics of functions in a declarative Function Modifiers are used to modify the behavior of a function.

2. Comment: Any text before you write in solidity language // the end of a line and also In between /* ….. */ and the end of a line is treated as a comment and is ignored by Solidity Compiler.

```
// This is a comment. It is similar to comments in C++
  /*
    * This is a multi-line comment in solidity
    * It is very similar to comments in C Programming
  */
```

3. Reserved Keywords: Following are the reserved keywords in Solidity

abstract	after	alias	apply
auto	case	catch	copyof
default	define	final	immutable
implements	in	inline	let
macro	match	mutable	null

Of	override	partial	promise
reference	relocatable	sealed	sizeof
static	supports	switch	try
typedef	typeof	unchecked	

4. Data Type: Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory**.**

Data Type used in Solidity:

bool, integer (int8-int256/uint8-uint256), address, bytes, string, hex, enum

5. Variable: Variables are the names you give to computer memory locations which are used to store values in a computer program.

Solidity supports three types of variables.

- ✓ **State Variables** − Variables whose values are permanently stored in a contract storage. That declared into contract.
- ✓ **Local Variables** − Variables whose values are present till function is executing. That declared inside a function.
- ✓ **Global Variables** − Special variables exists in the global namespace used to get information about the blockchain. This are specific variables that are exist in global workplace.

Scope of local variables is limited to function in which they are defined but State variables can have three types of scopes with names public ,private, internal.

e.g. uint public x = 40;

6. Operators:

Solidity supports the following types of operators.

- ✓ **Arithmetic Operators:** +, -, *, / , %, ++, --
- ✓ **Comparison Operators:** ==, <, >, <=, >=,!=
- ✓ **Assignment Operators:** =, += ,-=, *=, /=, %=
- ✓ **Logical (or Relational) Operators:**&&, ||,!
- ✓ **Assignment Operators:**&, | , << , >>, ~, ^, >>>
- ✓ **Conditional (or ternary) Operators:**?:

7. Loops:

- **while loop** is to execute a statement or code block repeatedly as long as an **expression** is true. Once the expression becomes **false,** the loop terminates.

```
Syntax:
while (conditional expression) {
  Statement(s) to be executed if conditional expression is true
}
```

- **Do…while loop:**The **do...while** loop is similar to the **while** loop except that the condition check happens at the end of the loop

```
Sytax:    do {
            Statement(s) to be executed;
          } while (conditional expression);
```

- **For Loop :**It is a iterative loop. Its contains initialization, condition, iteration statement

Syntax:

```
for (initialization; condition; iteration statement) {

  Statement(s) to be executed if condition is true }
```

8. Decision making statement: Decision making statements such as If statement, if else statement & If else if statement same as c programming language.

9. String: String is nothing but a sequence of characters. String literal using both double quote (") and single quote (')

eg: string myname = "Rajashri";

In Solidity we can assign String literal to a byte32 type variable easily.

byte32 myname = “Rajashri”;

We can convert byte to string following code

```
bytes memory bstr = new bytes(10);
string message = string(bstr);
```

10. Array: Arrays can have a compile-time fixed size, or they can have a dynamic size. The type of an array of fixed size k and element type T is written as T[k], and an array of dynamic size as A[].

- **Bytes and Strings as Arrays** Variables of type bytes and string are special arrays. A bytes is similar to byte[], but it is packed tightly in call data and memory. String is equal to bytes but does not allow length or index access.
- **Allocating Memory Array:** Memory arrays with dynamic length can be created using the new operator. As opposed to storage arrays, it is not possible to resize memory arrays (e.g. the .push member functions are not available). You either have to calculate the required size in advance or create a new memory array or copy every element.

```
pragma solidity >=0.4.16 <0.7.0;
```

```
contract C {
function f(uint len) public pure {
uint[] memory a = new uint[](7);
bytes memory b = new bytes(len);
assert(a.length == 7);
assert(b.length == len);
a[6] = 8;}
}
```

Array Members

1. **length**: Arrays have a length member that contains their number of elements. The length of memory arrays is fixed (but dynamic, i.e. it can depend on runtime parameters) once they are created.
2. **push(x):** Dynamic storage arrays and bytes (not string) have a member function called push(x) that you can use to append a given element at the end of the array. The function returns nothing. Syntax: x.push().t = 2 or x.push() = b.
3. **pop:** Dynamic storage arrays and bytes (not string) have a member function called pop that you can use to remove an element from the end of the array. This also implicitly calls delete on the removed element.

11. Enum Types: Enum can be used to create custom types with a finite set of 'constant values'

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.0 <0.7.0;
contract Purchase {
enum State { Created, Locked, Inactive } // Enum
}
```

Struct Types: Structs are custom defined types that can group several variables

```
pragma solidity >=0.4.0 <0.7.0;
contract Ballot {
struct Voter { // Struct
uint weight;
bool voted;
address delegate;
uint vote;
}
}
```

12. Function: Functions are the executable units of code within a contract. Function Calls can happen internally or externally and have different levels of visibility towards other contracts. Functions accept parameters and return variables to pass parameters and values between them.

Function syntax:
Function FunctionName(Arguments…) <visibility><state mutablitiy> returns (<return types>) {

// statement that write in
}

State mutablitiy:

view – not to modify the state
pure – not to read from or modify the state
payable- to receive ether

```
e.g. function hi() public pure returns (string memory) {
       return ("Hello World");
    }
```

13. Cryptographic Function: Solidity provides inbuilt cryptographic functions as well. Some inbuilt function like

- ✓ **keccak256 (bytes memory)** : this function is useful for to computes the Keccak-256 hash of the input.
- ✓ **ripemd160(bytes memory) returns (bytes20)**: this function computes compute RIPEMD-160 hash of the input the SHA-256 hash of the input.
- ✓ **sha256(bytes memory)** : This function is useful to compute the SHA-256 hash of the input.
- ✓ **recrecover(bytes32 hash, uint8 v, bytes32 r, bytes32 s) returns (address)** : This function is useful for recover address associated with the public key from elliptic curve signature or return zero on error.

e.g. function callKeccak256() public pure returns(bytes32 result{
return keccak256("ABC");
}

14. Function modifiers: can be used to amend the semantics of functions in a declarative Function Modifiers are used to modify the behavior of a function. The function body is inserted where the special symbol "_;" appears in the definition of a modifier.

```
//SPDX-License-Identifier: MIT

pragma contract ^5.0.0

contract Purchase {
address public seller;
modifier onlySeller() { // Modifier
require(
msg.sender == seller,
```

```
"Only seller can call this."
);
_;
}
function abort() public view onlySeller { // Modifier usage
// ...
}
}
```

15. Events: Events are convenience interfaces with the EVM logging facilities.Event is an inheritable member of a contract. An event is emitted, it stores the arguments passed in transaction logs. These logs are stored on blockchain and are accessible using address of the contract till the contract is present on the blockchain.

```
//Declare an Event
event Deposit(address indexed _from, bytes32 indexed _id, uint _value);

//Emit an event
emit Deposit(msg.sender, _id, msg.value);
```

16. Contract: Contract in Solidity is similar to a Class in C++. A Contract have following properties.

- **Constructor** − A special function declared with constructor keyword which will be executed once per contract and is invoked when a contract is created.
- **State Variables** − Variables per Contract to store the state of the contract.
- **Functions** − Functions per Contract which can modify the state variables to alter the state of a contract.

Visibility Quantifiers

Following are various visibility quantifiers for functions/state variables of a contract.

- **external** − External functions are meant to be called by other contracts. They cannot be used for internal call. To call external function within contract this.function_name() call is required. State variables cannot be marked as external.
- **public** − Public functions/ Variables can be used both externally and internally. For public state variable, Solidity automatically creates a getter function.
- **internal** − Internal functions/ Variables can only be used internally or by derived contracts.
- **private** − Private functions/ Variables can only be used internally and not even by derived contracts.

E.g. public contract MyContract {

// Statements

}

Contracts support inheritance like using is keyword

Contract C is D {

//statements

}

17. Mapping: Mapping is a reference type as arrays and structs. Mapping can only have type of **storage** and are generally used for state variables.Mapping used as public modifier. Solidity automatically creates getter for it.

Syntax:

mapping(_KeyType => _ValueType)

- **KeyType** − can be any built-in types plus bytes and string. No reference type or complex objects are allowed.
- **ValueType** − can be any type.

18. Address: The address type is to use for account address specially. comes in two flavors, which are largely identical:

- **address:** Holds a 20 byte value (size of an Ethereum address).
- **address payable:** Same as address, but with the additional members transfer and send. The idea behind this distinction is that address payable is an address you can send Ether to, while a plain address cannot be sent Ether

Members of Addresses

• balance and transfer:

It is possible to query the balance of an address using the property balance and to send Ether (in units of wei) to a payable address using the transfer function.

```
address payable x = address(0x123);
address myAddress = address(this);
if (x.balance < 10 && myAddress.balance >= 10) x.transfer(10);
```

• **send**: Send is the low-level counterpart of transfer. If the execution fails, the current contract will not stop with an exception, but send will return false.

```
recipient.send(1 ether);
```

• call, delegatecall and staticcall

In order to interface with contracts that do not adhere to the ABI, or to get more direct control over the encoding, the functions call, delegatecall and staticcall are provided.

```
bytes memory payload = abi.encodeWithSignature("register(string)", "MyName");
(bool success, bytes memory returnData) = address(nameReg).call(payload);
```

```
require(success);
```

Truffle:

Truffle is a development environment, testing framework and asset pipeline for Ethereum, aiming to make life as an Ethereum developer easier. With Truffle, we can Built-in smart contract compilation, linking, deployment and binary management.With Truffle, you get:

- Built-in smart contract compilation, linking, deployment and binary management.
- Automated contract testing with Mocha and Chai.
- Configurable build pipeline with support for custom build processes.
- Scriptable deployment & migrations framework.
- Network management for deploying too many public & private networks.
- Interactive console for direct contract communication.
- Instant rebuilding of assets during development.
- External script runner that executes scripts within a Truffle environment.

Web3:

Web3 is a collection of libraries which allow you to interact with a local or remote Ethereum node, using a HTTP or IPC connection. The web3 JavaScript library interacts with the Ethereum blockchain. It can retrieve user accounts, send transactions, interact with smart contracts, and more.

Web3.js API Type:

- eth: Ethereum blockchain related methods
- net: Node's network status
- personal: Account functions and sending
- db: Get/put for local LevelDB
- shh: P2P messaging using Whisper

Hyperledger

Hyperledger is an umbrella project, under the Linux Foundation. NodeJs, Alljoyn, Dronecode are some example projects that have adopted the "Linux Way", i.e. to interlace a community of developers who work on open source projects thus maintaining a cycle where a piece of code is constantly getting modified and redistributed.

Hyperledger Fabric:

Hyperledger Fabric is a popular Hyperledger project. It is used for developing solutions and applications with a modular architecture. Some of its key features include plug-and-play membership, consensus, and other features. This gives the industry a lot of options to work with blockchain technology and can be used in a plethora of industry use-cases. Also, its focus is to bring more scalability without the need to sacrifice privacy. It has two components, including State data and transaction log. Hyperledger Fabric supports to world state data store in LevelDB & CouchDB.

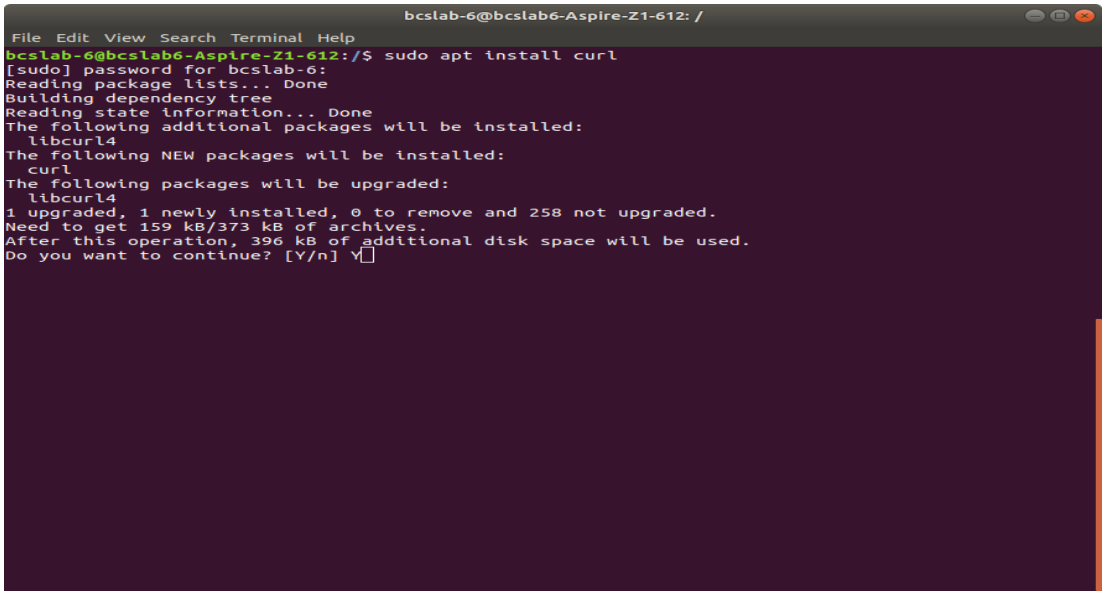
Fabric is the first distributed ledger platform to support smart contracts authored in general-purpose programming languages such as Java, Go and Node.js, rather than constrained domain-specific languages (DSL). This means that most enterprises already have the skill set needed to develop smart contracts, and no additional training to learn a new language or DSL is needed.

At the core, Hyperledger Fabric wants to provide a customized yet powerful and comprehensive enterprise blockchain solution.

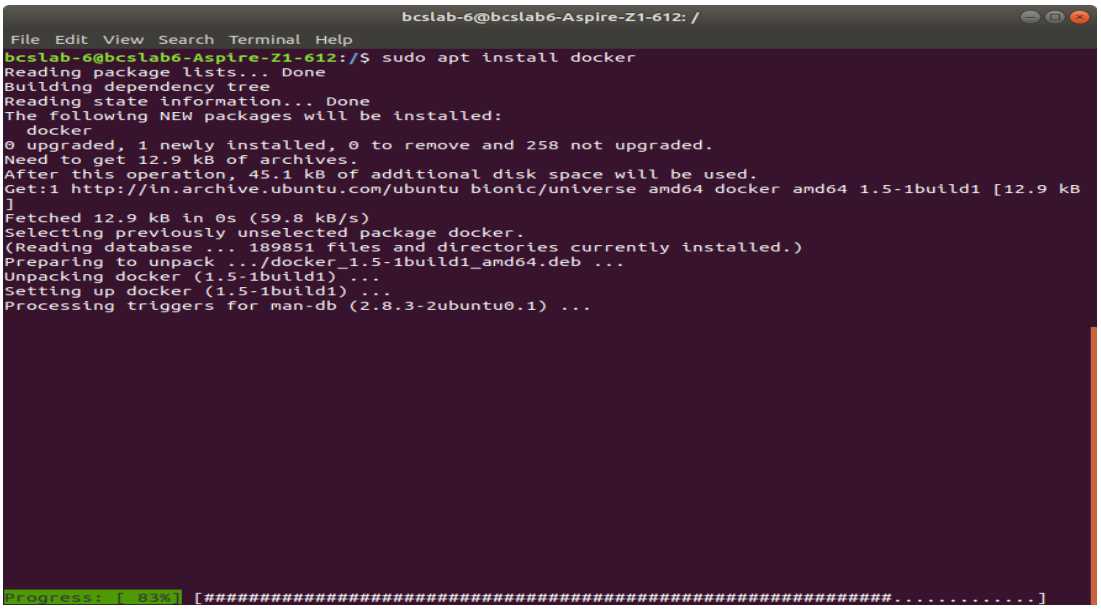
Steps of Installing Hyperledger fabric & build a Hyperledger on Ubuntu

Prerequisite:

Step 1: Installing cURL type on terminal: **sudo apt install curl**

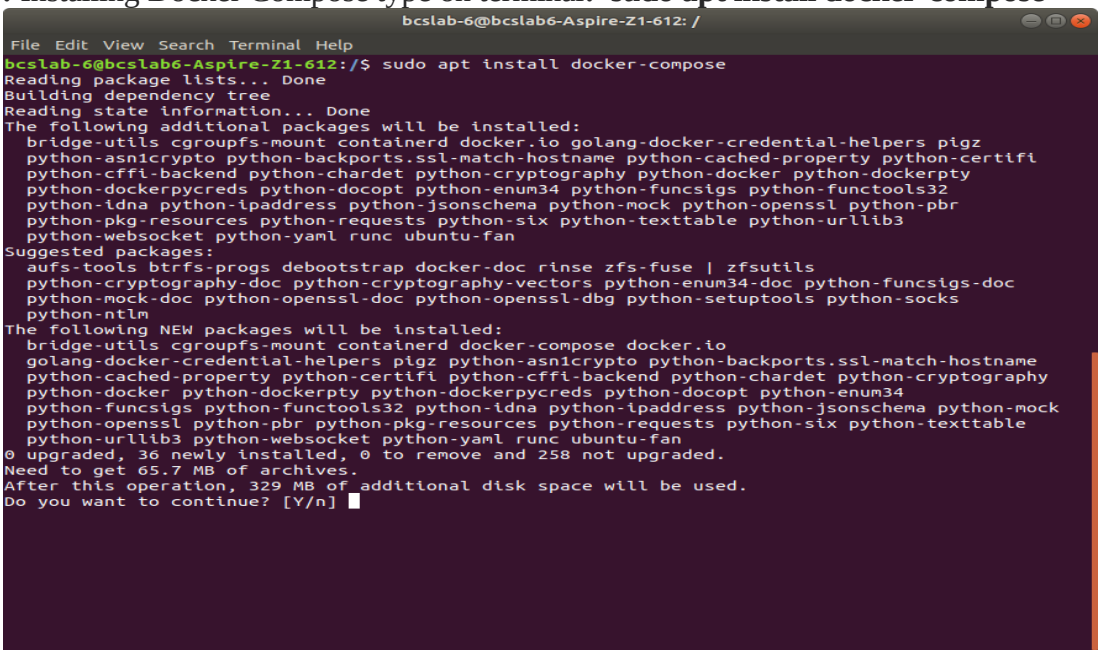


Step 2: Installing Docker type on terminal **: sudo apt install docker**



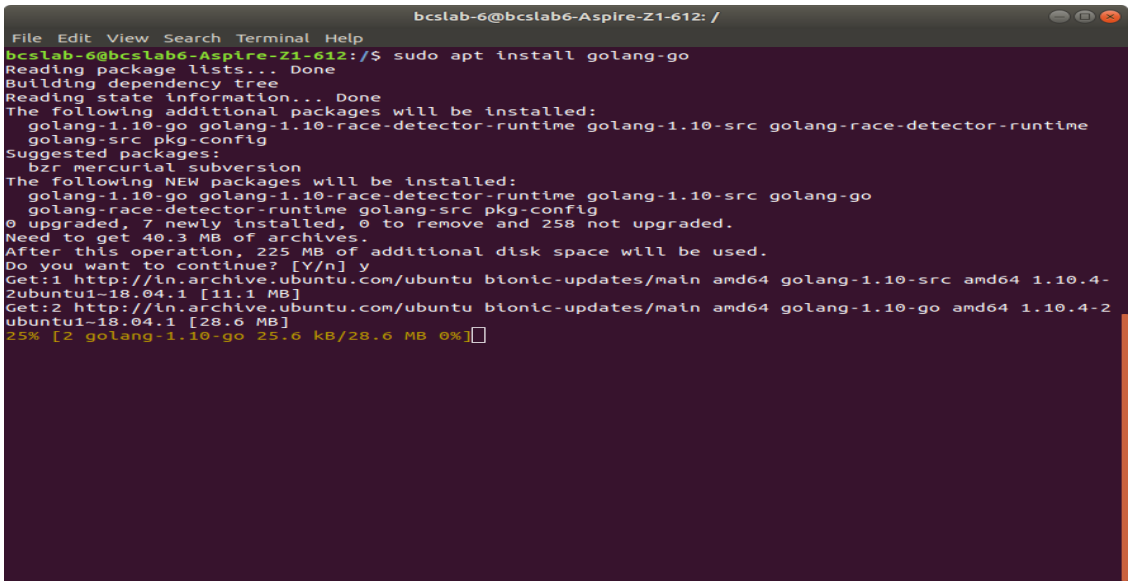
After installation you check the version type on terminal : **docker–version**

Step 3: Installing Docker Compose type on terminal**: sudo apt install docker-compose**

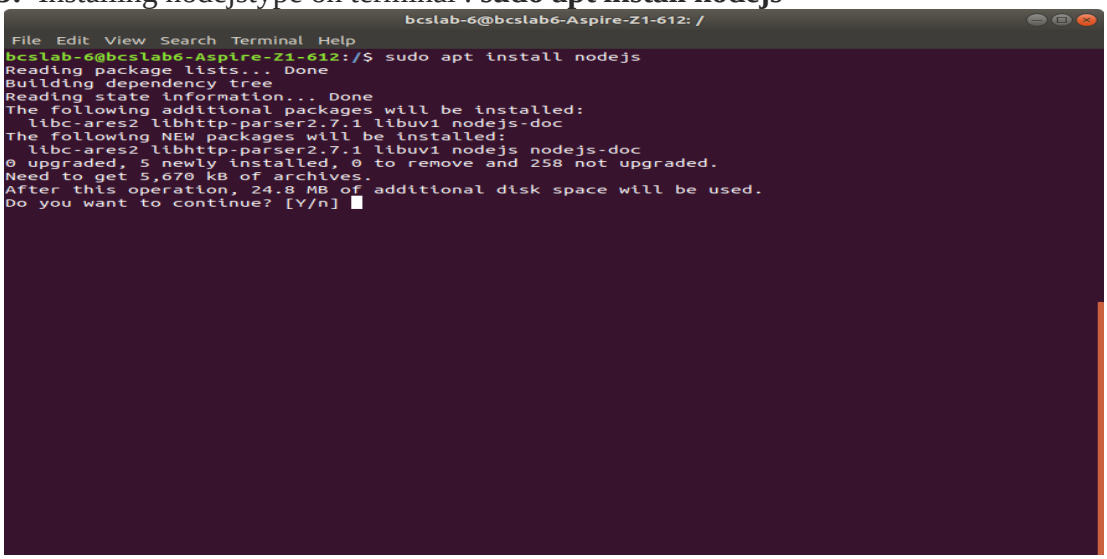


After installation you check thee version type on terminal: **docker-compose --version**

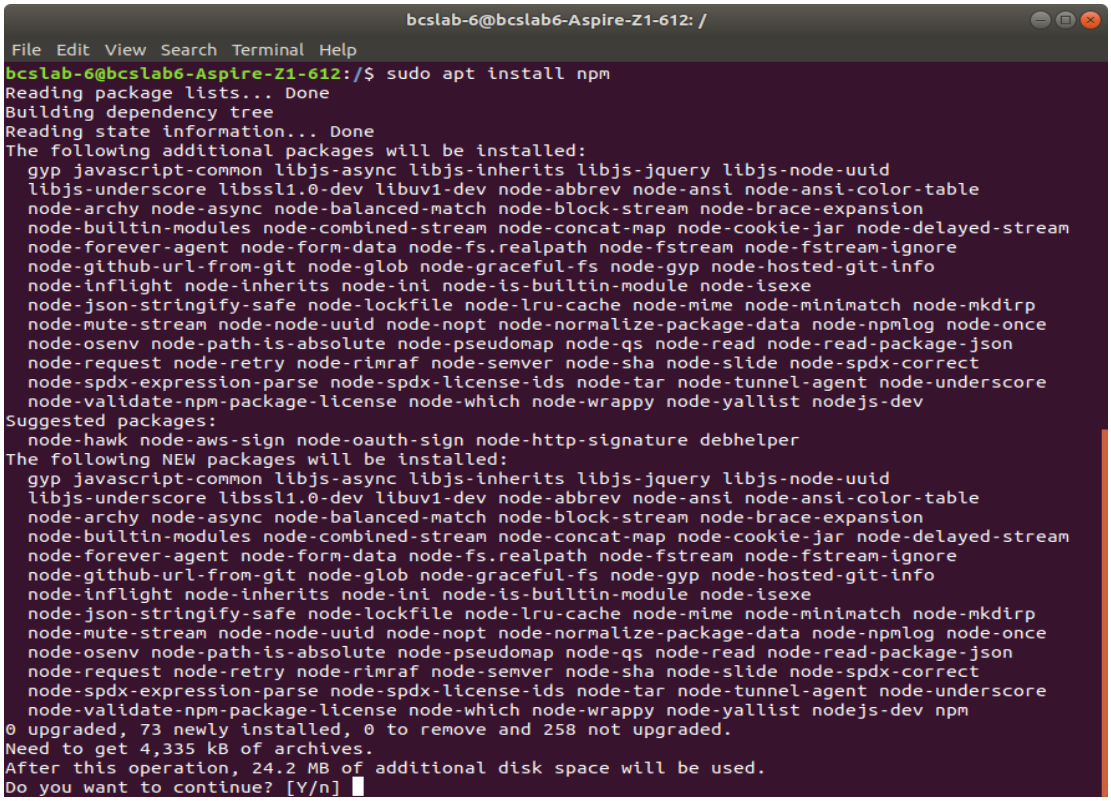
Step 4: Installing Go Language type on terminal **: sudo apt install golang-go**



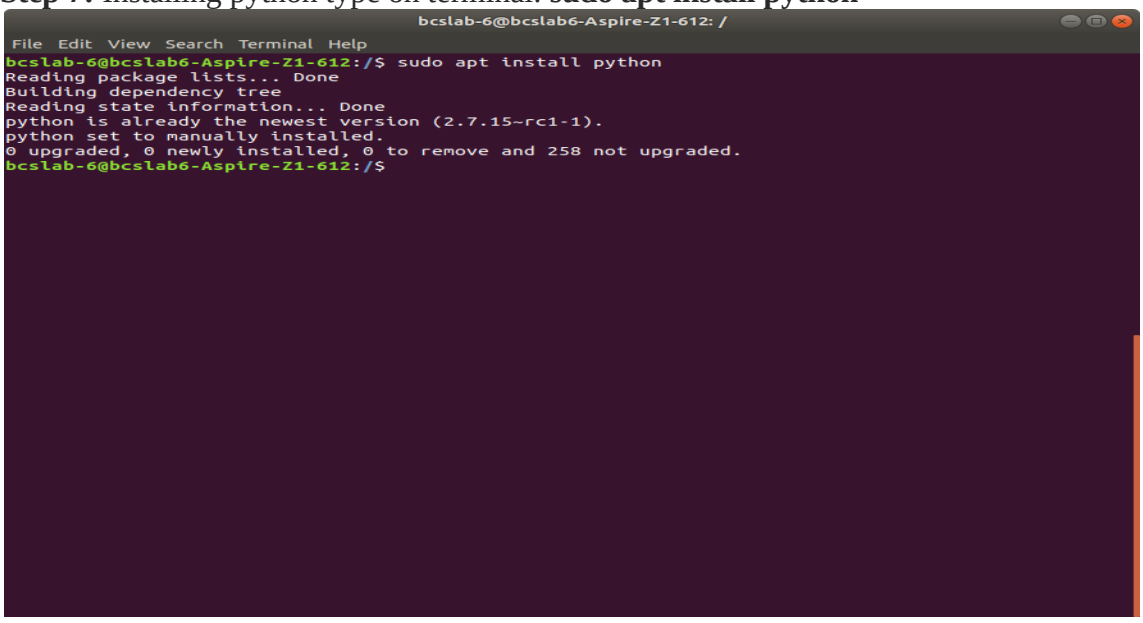
Step 5: Installing nodejstype on terminal : **sudo apt install nodejs**



Step 6: Installing npm type on terminal : **sudo apt install npm**



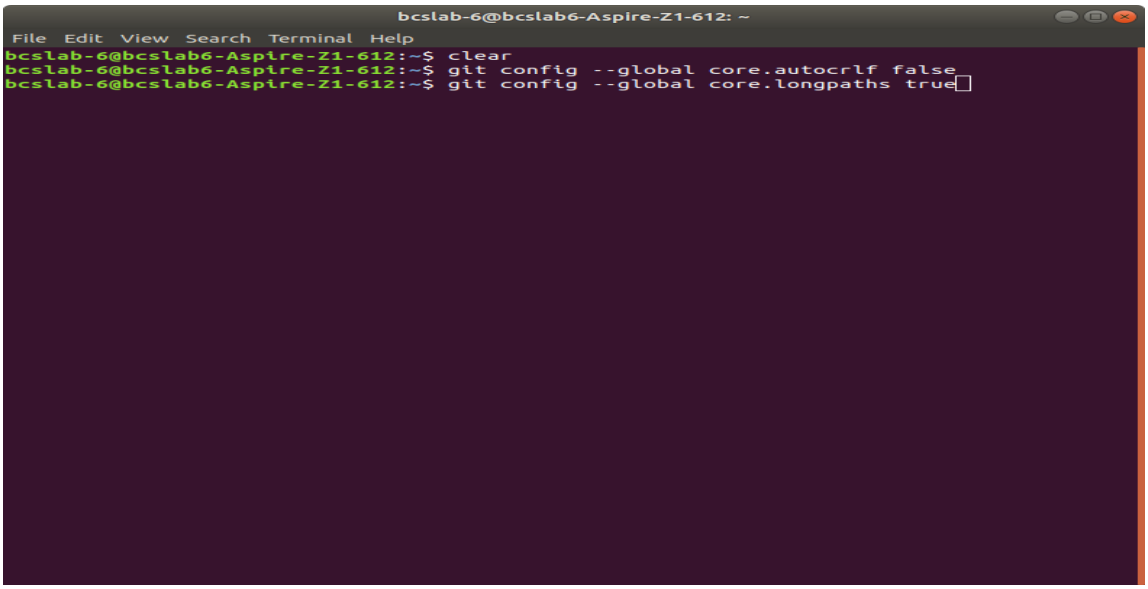
Step 7: Installing python type on terminal: **sudo apt install python**



Step 8: After Installation these two command type on

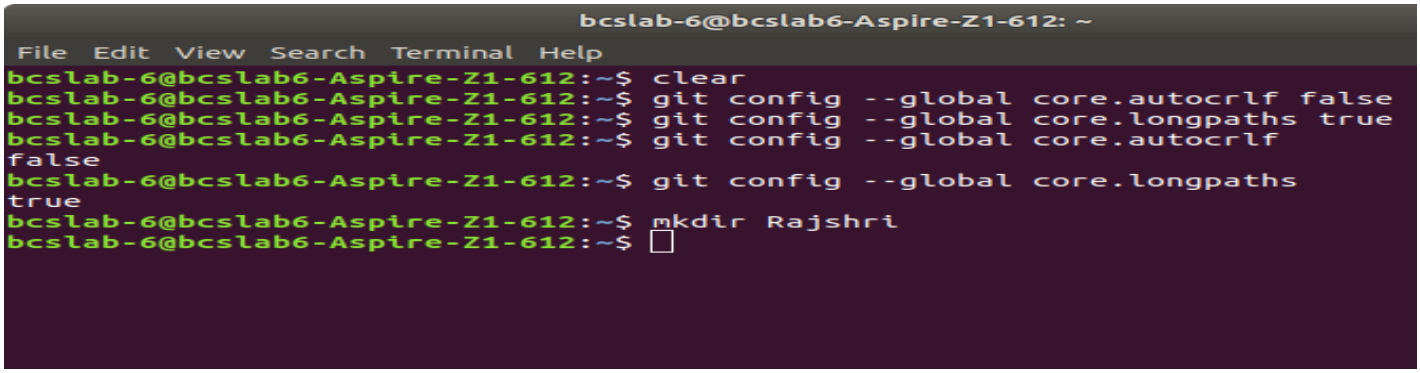
gitconfig --global core.autocrlf false

gitconfig --global core.longpaths true



For checking that command
gitconfig --global core.autocrlf
gitconfig --global core.longpaths

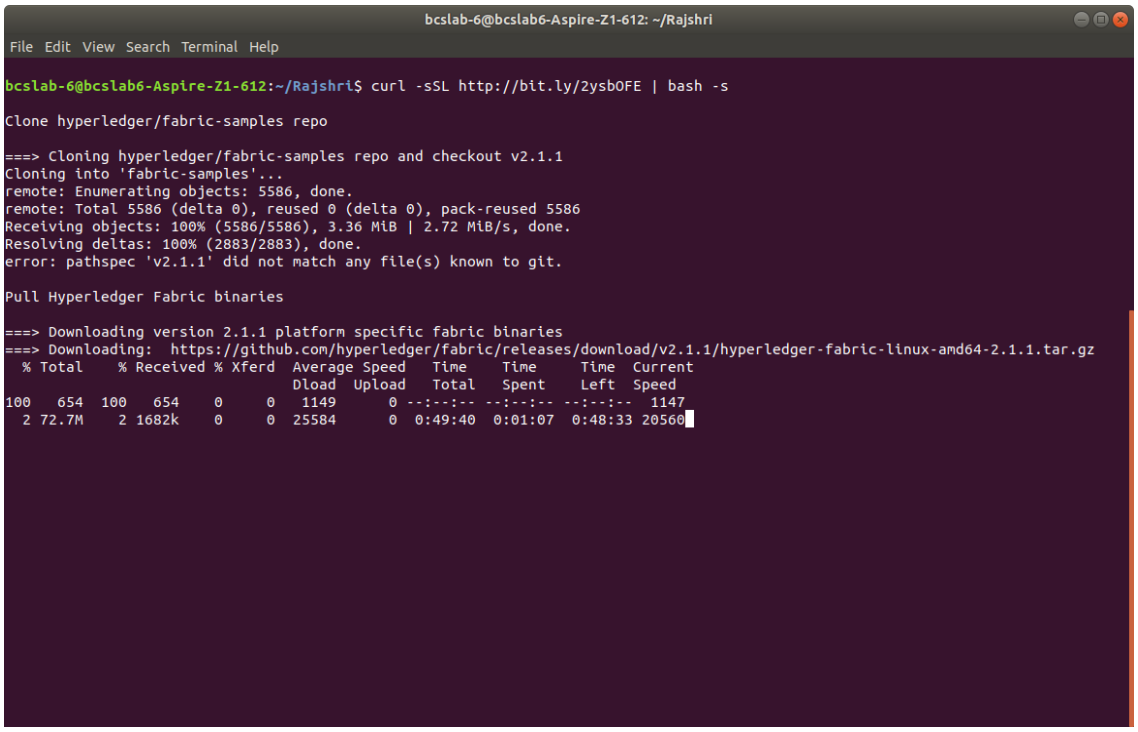
Step 9: Create your own directory type on terminal
mkdir <your directory name>
e.g. mkdir Rajshri



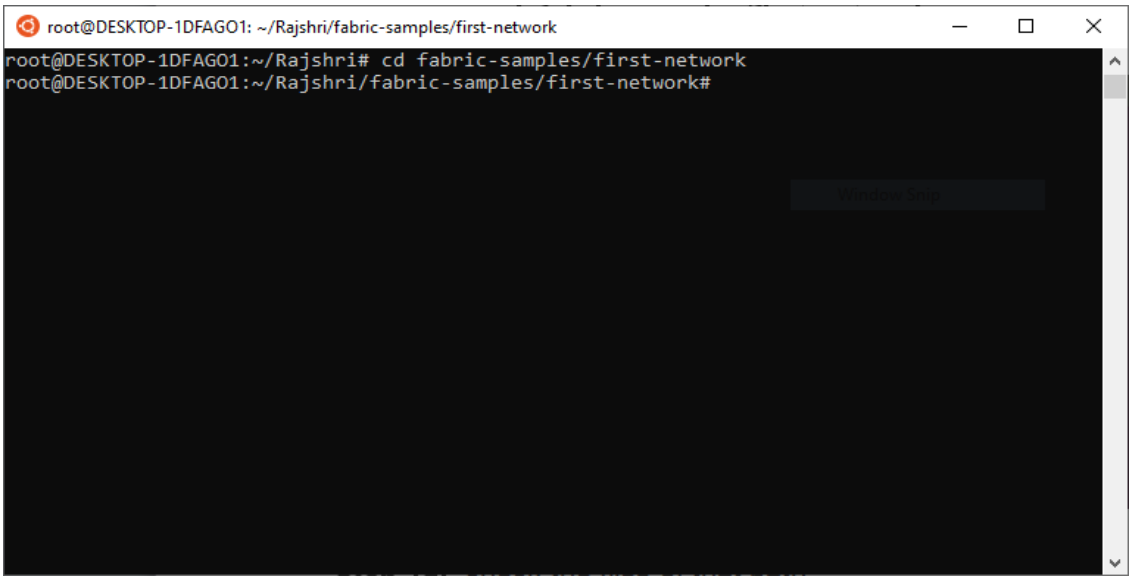
Step 10: Go to your created directory. Type on terminal
cd<your directory name> e.g.cd Rajshri

Step 11: Download files of fabric sample using this command type on terminal
curl -sSLhttps://goo.gl/6wtTN5 **| bash -s 1.1.0 1.1.0 0.4.6**

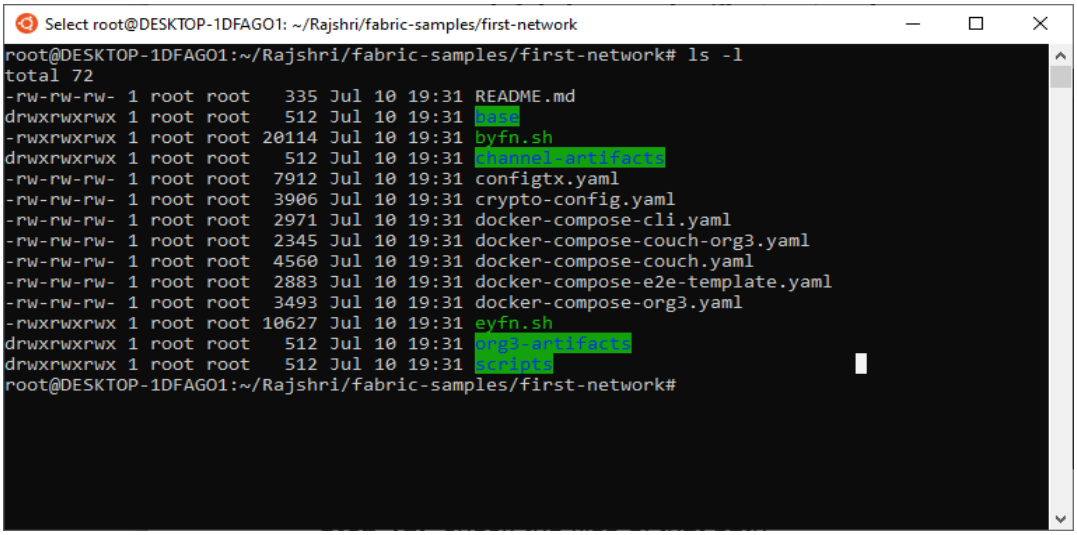
Download for latest version files of fabric samples
curl –sSLhttp://bit.ly/2ysbOFE **| bash -s**



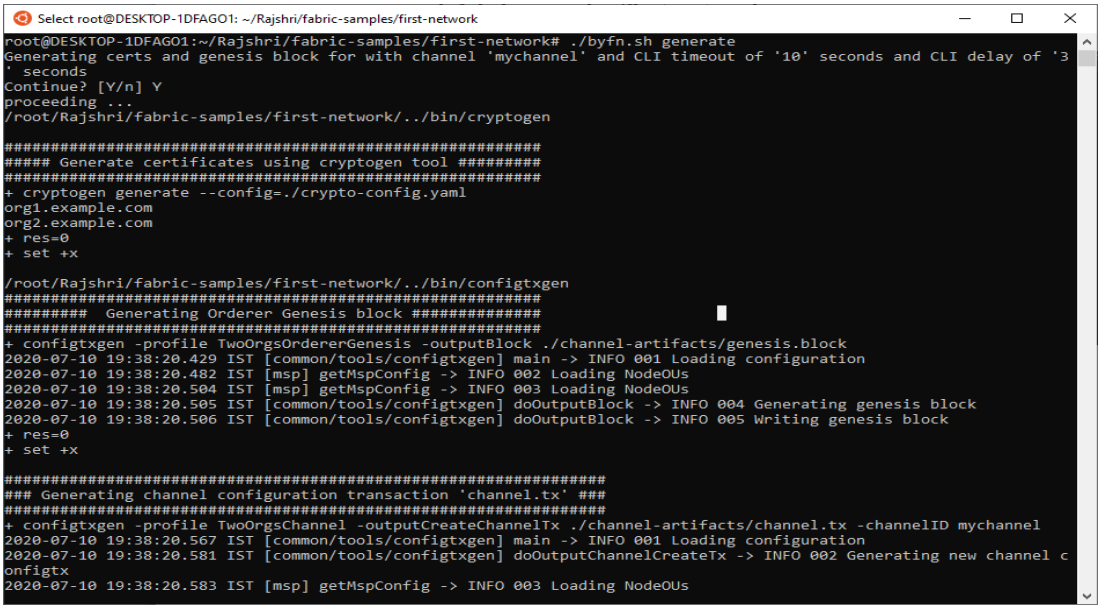
Step 12: Go to your downloaded directory. Type on terminal
cd fabric-samples/first-network



Step 13: check the files on that path Type on terminal: **ls -l**



Step 14: Then generate a network using to run file type on terminal
./byfn.sh generate



Step 15: Then start a network using type on terminal: **./byfn.sh up**

Step 16: Then end or stop a network using type on terminal: **./byfn.sh down**

Solved Programs

1. Write an Ethereum application in JavaScript for smart contracts.

Write Code in SmartContarct1.sol

```
//SPDX-License-Identifier: MIT
pragma solidity >= 0.5.0 < 0.7.0;

contract SmartContract1 {
  uint storedData;
  address payable[] recipient;
  constructor() public {
  }
  function set(uint x) public {
  }

  function get() public view returns(uint retVal){
    return storedData;
  }
}
```

Write Code in Deploy_smartContract.js

```
const SmartContract1  = artifacts.require("./SmartContract1.sol");

module.exports = function(deployer) {
  deployer.deploy(SmartContract1);
};
```

truffle compile

```
Compiling your contracts...
===========================
> Compiling .\contracts\SmartContarct1.sol
>Artifacts written to D:\SmartContracts\HelloWorld\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang
```

truffle migrate

```
Compiling your contracts...
===========================
> Compiling .\contracts\SmartContarct1.sol
>Artifacts written to D:\SmartContracts\HelloWorld\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang
 Starting migrations...
======================
```

```
> Network name:    'ganache'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)

deploy_SmartContarct1.js
========================

   Replacing 'SmartContarct1'
   ----------------------
>transaction
hash:    0xce20f7cdc9439852e3a7c5c214d0224b776bb09ec115786b99ae6face03d1db3
> Blocks: 0            Seconds: 0
>contract address:    0x980B5f555278Da4f02baee2B506dF941169bf257
>block number:        5
>block timestamp:     1593797676
>account:             0xD5d153C889d668DF699225674f4d2d91bE16f5F8
>balance:             99.99113358
>gas used:            94683 (0x171db)
>gas price:           20 gwei
>value sent:          0 ETH
>total cost:          0.00189366 ETH


> Saving migration to chain.
> Saving artifacts
   -------------------------------------
> Total cost:          0.00189366 ETH
```

2. Write an Ethereum application in JavaScript for HELLO World contract.

Write Code in helloworld.sol

```
//SPDX-License-Identifier: MIT
pragma solidity >= 0.5.0 < 0.7.0;

contract helloworld {
  constructor() public {
  }
  function hi() public pure returns (string memory) {
       return ("Hello World");
    }
}
```

Write Code in Deploy_helloworld.js

```
var HelloWorld=artifacts.require ("./helloworld.sol");
```

```
module.exports = function(deployer) {
     deployer.deploy(HelloWorld);
}
```

truffle compile

```
Compiling your contracts...
==========================
> Compiling .\contracts\ helloworld.sol
>Artifacts written to D:\SmartContracts\HelloWorld\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang
```

truffle migrate

```
Compiling your contracts...
==========================
> Compiling .\contracts\helloworld.sol
>Artifacts written to D:\SmartContracts\HelloWorld\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang
 Starting migrations...
======================
> Network name:    'ganache'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)
```

V3_Deploy_sendether.js

```
======================

   Replacing 'helloworld'
   ----------------------
>transaction
hash:    0x0e893fbfd316a7d984a108beeee3a2d98150a0879eecb89ccf6dcd637d593426
> Blocks: 0            Seconds: 0
>contract address:    0x11F11055C6cb9f16aC2035D628c98E14e32C6D3C
>block number:        7
>block timestamp:     1593797677
>account:             0xD5d153C889d668DF699225674f4d2d91bE16f5F8
>balance:             99.98383306
>gas used:            337685 (0x52715)
>gas price:           20 gwei
>value sent:          0 ETH
>total cost:          0.0067537 ETH


> Saving migration to chain.
> Saving artifacts
```

```
   -------------------------------------
> Total cost:          0.0067537 ETH

Summary
=======
> Total deployments:   4
> Final cost:          0.01422648 ETH
```

3. Write an Ethereum application in JavaScript to transfer currency from one account to another account.

Write a code under MySendEther.sol

```
 // SPDX-License-Identifier: MIT
pragma solidity >=0.4.0 <0.7.0;
contract MySendEther {
address public minter;
mapping (address => uint) public balances;
event Sent(address from, address to, uint amount);
constructor() public {
minter = msg.sender;
}
function mint(address receiver, uint amount) public{
require(msg.sender == minter);
require(amount < 1e60);
balances[receiver] += amount;
}
function send(address receiver, uint amount) public {
require(amount <= balances[msg.sender], "Insufficient balance.");
balances[msg.sender] -= amount;
balances[receiver] += amount;
emit Sent(msg.sender, receiver, amount);
}
}
```

Write a code under Deploy_MySendEther.js

```
const MySendEther = artifacts.require("./MySendEther.sol");

module.exports = function(deployer) {
  deployer.deploy(MySendEther);
};

truffle compile
Compiling your contracts...
===========================
> Compiling .\contracts\ MySendEther.sol
>Artifacts written to D:\SmartContracts\HelloWorld\build\contracts
```

```
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang

truffle migrate
Compiling your contracts...
===========================
> Compiling .\contracts\MySendEther.sol
>Artifacts written to D:\SmartContracts\HelloWorld\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang
Starting migrations...
======================
> Network name:    'ganache'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)
deploy_MySendEther.js
=====================
 Replacing 'MySendEther'
   ----------------------
>transaction
   hash:    0x0baefe4d8587feef205173fd316734a5d38df66f6ef9865b2fd20ae45ca61b02
> Blocks: 0            Seconds: 0
>contract address:    0xf484673A0BafcB54Dc061e725479C21C0C2C5557
>block number:        3
>block timestamp:     1593797675
>account:             0xD5d153C889d668DF699225674f4d2d91bE16f5F8
>balance:             99.99357406
>gas used:            114781 (0x1c05d)
>gas price:           20 gwei
>value sent:          0 ETH
>total cost:          0.00229562 ETH


> Saving migration to chain.
> Saving artifacts
   -----------------------------------
> Total cost:          0.00229562 ETH
```

Practice Programs:

1. Write a solidity program for the implementation of scope of the veriables (local, state, global).
2. Write a solidity program to perform setter and getter operations on State veriable.
3. Write a solidity program for the implementation of looping statement.
4. Write a solidity program to implement compile-time fixed size or a dynamic size array.
5. Write an Ethereum application in JavaScript for smart contracts.

6. Write an Ethereum application in JavaScript to transfer currency from one account to another account.
7. Write an Ethereum application in JavaScript for the implementation of blockchain technology.
8. Write a blockchain application in JavaScript to create a block by using hyperledger fabric.

SET A:

1. Write a solidity program to display "Hello Word"
2. Write a solidity program to create simple contract that you can get, increment and decrement the count store in this contract.
3. Write a solidity program for the implementation of primitive data types.

SET B:

1. Write a solidity program to convert Ether and Wei (Note: one dollar is equal to 100 cent, one ether is equal to 10^{18} wei)
2. Write a solidity program to calculate how much ether do you need to pay for a transaction?(Formula: gas spent * gas price amount of ether)
   Note:
   - gas is a unit of computation
   - gas spent is the total amount of gas used in a transaction
   - gas price is how much ether you are willing to pay per gas
3. Write a solidity program for the implementation of conditional statement.

SET C:

1. Write a solidity program to implement enumeration (enum).
2. Write a solidity program to implement the function.
3. Write an Ethereum application in JavaScript for HELLO World contract.

Assignment Evaluation

0: Not Done [ ]	**1: Incomplete [ ]**	**2: Late Complete [ ]**
3: Need Improvement [ ]	**4: Complete [ ]**	**5: Well Done [ ]**

Signature of Instructor

Savitribai Phule Pune University

S. Y. B. B. A. (C. A.) Semester-IV

(CBCS 2019 Pattern)

Object Oriented Concepts Through CPP, NODE JS and Advance PHP

CA-406: Lab Book

RollNo.:_____ **Division:**_____ **SeatNo.:**_____

Student Name:_____

CollegeName:_____

AcademicYear:_____

CERTIFICATE

This is to certify thatMr./Ms._____ Seatnumber_____ of S.Y.B.B.A.(C.A) Sem-IV has successfully completed Laboratory Course (Object Oriented Concepts Through CPP and NODE JS / Advance PHP) in the year ____. He/She has scoredmark out of 10 (for Labbook).

SubjectTeacher **H.O.D./Coordinator**

InternalExaminer **ExternalExaminer**

Editorial Board: Dr. D. Y. Patil Arts, Commerce and Science College, Pimpri.

Section-I: Object Oriented Concepts ThroughCPP

Mrs. Vidya Bankar.

Mrs. Reshma Masurekar.

Mrs. Ashwini Satkar.

Section-II: NODE JS

Mr. Bhushan Nikam.

Mr. Satish Mulgi.

Section-III: Advance PHP

Mrs. Madhuri Darekar.

Mrs. Trupti Kulkarni.

Reviewed By:

Dr. Ranjit Patil.

Mrs. SujataPatil.

Mrs. Sangeeta Nimbalkar.

Mrs. LeenaBhat.

Mr. Sudarshan Lakhdive.

Mrs. Shakila Siddavatam.

Mr. Shivendu Bhushan.

Introduction

1. About the workbook:

This workbook is intended to be used by S.Y.B.B.A. (C.A.) Semester-IV students for Object Oriented Concepts Through CPP, NODE JS and Advance PHP assignments as well as Add-on subject jQuerry. It is designed by considering all the practical topics mentioned in the syllabus.

2. The objectives of this workbookare:

- Defining the scope of thecourse.
- Tobring theuniformityin thepracticalconductionandimplementationinallcolleges affiliated toSPPU.
- To have continuous assessment of the course andstudents.
- Providing ready reference for the students during practicalimplementation.
- Providemoreoptionstostudentssothattheycanhavegood practicebefore facingthe examination.
- Catering to the demand of slow and fast learners and accordingly providing the practice assignments tothem.

3. How to use thisworkbook:

The workbook is divided into three sections. Section-I is related to CPP assignments, Section-IIisrelatedtoNODEJSassignmentsandSection-IIIisrelatedto AdvancePHP assignments.
Section-I CPP is divided into nine assignments.
Section-II NODE JS is divided into four assignments.
Section-III Advance PHP is divided into six assignments.
From Section-II and Section-III students have to perform practical assignments of selected elective subject only.
Each assignment of all sections has three SETs-A, B and C. It is mandatory for students to complete SET A and SET B in lab. Assignment also includes practice programs which are expected to be solved by students as home assignments and to be evaluated by subject teachers.

4. Instructions to thestudents

Please read the following instructions carefully and follow them during practical.

- Students are expected to carry this workbook every time they come to the lab for computerpractical.
- Students should prepare for the assignment by reading the relevant material which is mentioned in readyreference.
- Instructor will specify which problems to solve in the lab during the allotted slot and student should complete them and get verified by the instructor. However, student should spend additional hours in Lab and at home to cover all workbook assignments ifneeded.

- Students will be assessed for each assignment on a scale from 0 to5.

Not done	0
Incomplete	1
Late Complete	2
Needs improvement	3
Complete	4
Well Done	5

5. Instruction to theInstructors

Make sure that students should follow above instructions.

- Explain the assignment and related concepts in around ten minutes using whiteboard if required or by demonstrating thesoftware.
- Evaluate each assignment carried out by a student on a scale of 5 as specified above by ticking appropriatebox.
- The value should also be entered on assignment completion sheet of the respectivesection.

6. Instructions to the Labadministrator

You have to ensure appropriate hardware and software is made available to each student. The operating system and software requirements on server side and also client side areas given below:

- Operating System -Windows
- TurboC++
- WampServer
- Visual StudioCode

Assignment Completion Sheet

Section-I: Object Oriented Concepts Through CPP			
Sr. No.	**Assignment Name**	**Marks (out of 5)**	**Teacher's Sign**
1	Beginning with C++		
2	Operators and Functions in C++		
3	Classes and Objects		
4	Constructors and Destructors		
5	Inheritance		
6	Polymorphism		
7	Managing Console I/O operations		
8	Working with Files		
9	Templates		
Total (Out of 45)			
Total (Out of 5)			

Instructor Signature:

Section-II: NODE JS			
Sr. No.	**Assignment Name**	**Marks (out of 5)**	**Teacher's Sign**
1	Node.js web server, modules &npm		
2	File system		
3	Events in node.js		
4	Node.js with database		
Total (Out of 20)			
Total (Out of 5)			

'OR'

Section-III: Advance PHP			
Sr. No.	**Assignment Name**	**Marks (out of 5)**	**Teacher's Sign**
1	Introduction to Object Oriented Programming in PHP		
2	To study Web Techniques		
3	XML		
4	PHP with AJAX		
5	Connecting Database using PHP & AJAX		
6	PHP Framework - Druple		
Total (Out of 30)			
Total (Out of 5)			

Instructor Signature:

Section-I

Object Oriented Concepts Through CPP

Assignment No. 1: Beginning with C++

Introduction:

In 1982, Bjarne Stroustrup started to develop a successor to C with Classes at Bell labs, which he named "C++", as it is an extension to C programming language. C++ runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. The major purpose of C++ programming is to introduce the concept of object orientation to the C programming language.

Procedural programming is about writing procedures or functions that perform operations on the data, while object-oriented programming is about creating objects that contain both data and functions.

Object-oriented programming has several following advantages over procedural programming:

- OOP is faster and easier toexecute.
- OOP provides a clear structure for the programs.
- OOP makes the code easier to maintain, modify anddebug.
- OOP makes it possible to create full reusable applications with less code and shorter developmenttime.
- OOPmakesdevelopmentandmaintenanceeasierifcodegrowsasprojectsizegrows.
- OOP provide datahiding
- OOP provide ability to simulate real-world event much moreeffectively.

C++ is a general purpose, object oriented programming language. C++ has some additional facilities to those in C such as classes, data binding, data hiding, inheritance, encapsulation, polymorphism, default function argument etc. because of which it allows code to be reused and lowering development costs.

Real-World applications of C++:

- C++isclosetothehardware,caneasilymanipulateresourcesanditisfast,which makesit a primary choice to develop the **gamingsystems**.

- C++ can be used to develop most of the **GUI based and desktop applications** easily. Example: Adobe Photoshop, Win amp media player fromMicrosoft.

- C++ is also used in writing database management software. The two most popular databases **MySQL and Postgres** are written inC++.

- The fact that C++ is a strongly typed and fast programming language makes it an ideal candidate for writing **operating systems**. Apple OS X has some of its parts written in C++. Similarly, some parts of the iPod are also written in C++. Most of the software from MicrosoftisdevelopedusingC++(flavorsofVisualC++).ApplicationslikeWindows95, ME, 98; XP, etc. are written in C++. Apart from this, the IDE Visual Studio, Internet Explorer and Microsoft Office are also written inC++.

- **Browsers** are mostly used for rendering purposes. Rendering engines need to be faster in execution as most people do not like to wait for the web page to be loaded. With the fast performance of C++, most browsers have their rendering software written in C++.Mozilla Firefox internet browser is an open-source project and is developed completely in C++.Google applications like Google File System and Chrome browser are written in C++.

- C++ is useful in developing an **application that requires high-performance image processing, real-time physical simulations, and mobile sensor applications** that need high performance and speed. Maya 3D software from Alias system is developed in C++ and is used for animation, virtual reality, 3D graphics, andenvironments.

- **Compilers** of various high-level programming languages are written either in C or C++. The reason is that both C and C++ are low-level languages that are close to hardware and areabletoprogramandmanipulatetheunderlyinghardwareresources.

- C++ can be used for building higher-level applications with graphics libraries, **applicationstocommunicatewithnetworkdevices**andcomputernetworksimulatorsas well as remote device systems and networkmanagement.

C++ Data types:

Data types in C++ are mainly divided into three types:

1. **Primitive/Built-in Data Types**: These data types are built-in or predefined data types and can be used directly by the user to declare variables. Primitive data types available in C++are:
   - int
   - char
   - bool
   - float
   - double
   - void
2. **Derived Data Types:** The data-types that are derived from the primitive or built-in data types are referred to as Derived Data Types. These can be of four typesnamely:
   - Function
   - Array
   - Pointer
   - Reference
3. **Abstract or User-Defined Data Types**: These data types are defined by user itself.C++ provides the following user-defined data types:
   - Class
   - Structure
   - Union
   - Enumeration
   - Typedef definedDataType

Simple C++ Program:

Example: C++ Hello world program to simply print "Hello World" on computer screen.

```
// My first C++ program
#include <iostream.h>
int main()
{
        cout<< "Hello World!";
        return 0;
}
```

Fundamental components in C++ programs:

- **Comments: // My first C++program**
  In above C++ program first line with double slash symbol indicate single line comment and to indicate multiline comment programmer can enclose multiple lines in /* */ which means these lines inserted by the programmer has no effect on the behavior of the program. Programmers use comment to include short explanations or observations about program.

- **Header files: #include <iostream.h>**
  Lines beginning with '#' are directives which are read and interpreted by preprocessor before thecompilation of the C++ program begins. In above C++ program directive #include <iostream.h>, instructs the preprocessor to include header fileiostream, which allows to add the contents of the iostream.h file to the program to perform standard input and output operations, such as accepting input through keyboard and writing the output of program onscreen.

- **Main function: int main()**
  The function named main is a special function in all C++ programs; it is the function which called by operating system automatically when C++ program run. The execution of all C++ programs begins with the main function, regardless of where the function is actually located within the code.
  Proper way of writing the main function in C++ is to use int return type for mainfunction. C++ standards and specifications mention that the main function should always return integervaluewhich can be'0'or'1'where'0'isthestandard for"successfulexecutionof theprogram".

- **Operator:**
  In C++ input and output are performed in the form of a sequence of bytes or more commonly known as streams. Streams are of two types, if the direction of flow of bytes is from the device like keyboard to main memory then it is an Input Stream and if the direction of flow of bytes is from main memory to device like display screen then it is an output stream.

C++ isable to input and output the built-in data types using the stream extraction operator >> and the stream insertion operator << respectively. iostream stands for standard input-output stream in C++, this header file contains definitions to objects like cin, coutetc.

C++ Input and Output Operators:

InC++coutisapredefinedobjectwhichisaninstanceofostreamclass.Thecoutobjectis connected to the standard output device, which usually is the display screen. The cout is used in conjunction with output operator "<<" ("put to"), also known as stream insertion operator to direct a value to standard output i.e. to display output on thescreen.

cinisapredefinedobjectwhichisaninstanceofistreamclass.Thecin objectisconnected tothe standardinputdevice,whichusuallyisthekeyboard.The cinisusedin conjunction with input operator ">>" ("get from"), also known as stream extraction operator to read a value from standard input i.e. to accept input fromkeyboard.

Example: C++ program to illustrate use of input and output operators.

```
#include<iostream.h>
int main()
{
  int num1, num2;
  cout<<“\n Enter two numbers :”;
  cin>>num1>>num2;                                    //cascaded input operators
  cout<<“\n Two numbersare:”<<num1<<“ ”<<num2;       // cascaded outputoperators
  return0;
}
```

Cascading of Input / Output Operators in C++:

The cascading of the input and output operators refers to the multiple use of input or output operators in one statement. The statement using multiple output operators “<<” is saidtobecascadingoutputoperatorand the statementusing multiple inputoperators“>>” is said to be cascading inputoperator.

In above program, the cascaded input operators wait for the user to input two integer values, where values are assigned from left to right, means first input value will get assigned to num1 and second input value will get assigned to num2. The cascaded output operator first displays the message “Two numbers are:”, then displays the value ofnum1 and after that it displays the value of num2.

It is observed that cascading of the input/output operator improves the readability and reduces the size of the program.

- **C++ Program Execution:**

  Create a C++ program using editor in Turbo C++.Save the program usingF2, give a meaningful name to a source file which should reflects the purpose of the program, with extension ".cpp". Compile the program using Alt + F9.Execute your C++ program by pressingCtrl+F9.PressAlt+F5toviewtheoutputoftheprogramattheoutputscreen.

Practice Programs:

1. Write a C++ program to find factorial of a givennumber.
2. Write a C++ program to check whether a given number is even orodd.
3. Write a C++ program to check whether a given number is prime ornot.
4. Write a C++ program to check whether a given number is perfect ornot.
5. Write a C++ program to find largest and smallest number of 3 integer numbers. (Use cascading of I/Ooperators.)

Set A:

1. Write a C++ program to generate multiplicationtable.
2. Write a C++ program to display first ‘n’ numbers of Fibonacciseries.
3. Write a C++ program to reverse anumber.
4. Write a C++ program to display Armstrong numbers between twointervals.
5. Write a C++ program to accept two integers and an arithmetic operator(+, -, *, /) from user and performs the corresponding arithmetic operation and display the result. (Use switchstatement)

Set B:

1. WriteaC++programtoprintthefollowingpattern.
   A
   B C
   D E F
   G H I J

2. Write a C++ program to print the followingpattern

```
        *
    *   *
*   *   *
*   *   *
```

3. WriteaC++programtocalculatefollowingseries:
   (1*1)+(2*2)+(3*3)+ … +(n*n)

4. Write a C++ program to convert decimal number into binarynumber.

Set C:

1. Write a C++ program to print the followingpattern

```
    *
   * *
  * * *
 * * * *
* * * * *
```

2. WriteaC++programtocalculatefollowingseries:
   1/1! + 2/2! + 3/3! + …. +n/n!

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: WellDone [ ]

Signature of Instructor

Assignment No. 2: Operators and Functions in C++

Operators in C++:

C++ has a rich set of operators. All C operators are valid in C++ also. In addition to that C++ introduces some new operators. We have already seen two such operators namely, the insertion operator <<, and the extraction operator >>. Other new operators are:

Operator	Name of Operator	Function
::	Scope resolution operator	To access global version of a variable
::*	Pointer-to-member declarator	To declare a pointer to a member of a class
->*	Pointer-to-member operator	To access a member using a pointer to the object and a pointer to that member.
.*	Pointer-to-member operator	To access a member using object name and a pointer to that member.
new	Memory allocation operator	To allocate memory for object.
delete	Memory release operator	To free allocated memory of an object.
endl	Line feed operator	To insert a new line character similar to '\n'
setw	Field width operator	To sets the field width to be used on output operations.

Scope resolution operator:

In C++ the scope resolution operator (::) is used for several reasons, some of them are:

- Accessing a global variable when there is a local variable with samename
- Defining a function outside aclass
- Accessing a class's staticvariables
- Referring to a class inside anotherclass
- In case of multipleInheritance

Out of these, in this assignment we are going study first use of scope resolution operator. If the global variable name is same as that of local variable name, the scope resolution operator will be used to access the global variable.

Example: C++ program to illustrate use of Scope Resolution Operator (::)

```
#include<iostream.h>
int num=30;                                    // Initializing a global variablenum
intmain()
{       int num=10;                            //Initializing the local variable num
        cout<< "\nValue of global num is " <<::num;
        cout<< "\nValue of local num is " <<num;
        return 0;
}
```

Output:

Value of global num is 30
Value of local num is 10

In above program, we have two variables both named num with global & local scope. So, to access global num variable in main function we need to use scope resolution operator (i.e. ::num).

Memory management operators:

Allocating memory of a variable or an array run time is known as Dynamic Memory Allocation(DMA).In C, dynamic memory management is handled by malloc( ) and free( ) function, but in C++ dynamic memory management is handled by using operators called 'new' and 'delete', where 'new' operator replaces malloc( ) and 'delete' operator replaces free( ) in C. New and Delete operators manage memory effectively hence they are called as memory management operators. In C++, we need to deallocate the dynamically allocated memory manually after we have no use for the variable.

Syntax for anydatatype: pointer-variable = newdata-type;
delete pointer-variable;

Syntax foranarray: pointer-variable=newdata-type[size];
delete[size]pointer-variable;

Example: C++ program to illustrate use of memory management operators.

```
#include<iostream.h>
int main()
{       int*ptr;                    // declare an integerpointer
        ptr =newint;                // dynamically allocate memory for an int variable
        int *arr_ptr=newint[5];     // create a memory for an array of 5integers
        *ptr=100;                   // assign value to the memory
        cout<<*ptr<<endl;           // print thevalue stored in memory
        inti;               //variabledeclarationanywhereinthescopeisallowedinC++
        for(i=0;i<5;i++)
        {       arr_ptr[i]=i+1; //assign value to an arrayelements
        }
        cout<<arr_ptr[0];           // print first element of an array
        deleteptr;                  // deallocate the memory
        delete[]arr_ptr;            //deallocatememoryofanarray
        return0;
}
```

Output:

```
100
1
```

In above C++ program we have used pointer to allocate memory dynamically because newoperatorreturnstheaddressofmemorylocation.Incaseofanarray,thenewoperator returns the address of the first element of an array. Delete operator returns the memory to the operating system which is known as memorydeallocation.

Advantages of the new operator over malloc() function:

- It does not use the sizeof() operator as it automatically computes the size of the dataobject.
- It automatically returns the correct data type pointer, so does not need to use the typecasting.
- Itallowstoinitializethedataobjectwhilecreatingthememoryspaceforobject.

Manipulators:
Manipulators are operators that are used to change formatting parameters on streams and to insert or extract certain special characters, these are helping functions that can modify the input/output stream. It does not mean that we change the value of a variable, it only modifies the I/O stream using insertion (<<) and extraction (>>) operators. To use manipulators in C++ program we need to include header file iomanip.h.

Following are some of the most widely used C++ manipulators:

1) **endl:**
   endl is the line feed operator in C++. It acts as a stream manipulator whose purpose is to feed the whole line and then point the cursor to the beginning of the next line. We can use endl instead of '\n' (newline character) for the same purpose.
   Example:
   cout<<"Good"<<endl<<"Morning";
   This will display "Good" and "Morning" on two separate lines.
2) **setw:**
   setw manipulator function stands for set width. This manipulator is used to specify the minimum number of character positions on the output field a variable will consume, that is it sets the minimum field width on output. It is mostly used in output to right justify numbers.
   Example:
   Sum=123;
   cout<<setw(5) <<Sum;
   This sum value is right justified within the field.

		1	2	3

3) **setfill:**
   setfill is used after setw manipulator. If a value does not entirely fill a field, then the character specified in the setfill argument of the manipulator is used for filling the fields. It specifies a character that is used to fill the unused portion of a field.
   Example:
   cout<<setw(10)<<setfill('*')<<1234;
   This will give you output: ******1234
4) **setprecision:**
   The setprecision manipulator is used with floating point numbers. It is used to specify the number of digits to be displayed after the decimal point of a float value.
   Example:
   PI=3.14159;
   cout<<setprecision(2)<<PI;
   Output: 3.14

Functions in C++:

Function prototyping:
The function prototype describes the function interface and it is used to give details to the compiler about the number of arguments and about the required data types of a function parameter, it also tells about the return type of the function. Using these details, the compiler cross-checks the function signatures before calling it. If the function prototypes are not

mentioned, then the program may be compiled with some warnings. If some function is called somewhere in a program, but its body is not defined yet, that is defined after the current line, then it may generate problems. The compiler does not find what is the function and what is its signature. In that case, we need to use function prototyping. If the function is defined before, then we do not need to use prototypes.

Syntax:

return_type function_name (argument_list);

Example:

int multiplication (int x, int y, int z);

int addition (int, int, int); /*this is also acceptable at the place of declaration because at this stage, the compiler only checks for the type of arguments when the function is called. */

Call by reference:

Call by value means pass arguments by value to the function and call by reference means pass address of arguments to the function. In call by value, called function creates a new set of variable and copies the values of arguments into them. The function does not have access to the actual variables in the calling program. This mechanism is fine if the function does not want to alter the values of the original variables in calling program.

To change values of the original variables in calling program we have to use call by reference. In call by reference, address of the value is passed to the function, so actual and formal arguments share the same address space. Hence, value changed by called function will get reflected in calling function also.

Example: C++ program to illustrate use of call by reference.

```
void swap(int*, int*);
int main()
{       int a = 10, b=20;                   // initializevariables

        cout<< "Before swapping"<<endl;
        cout<< "a = " << a<<endl;
        cout<< "b = " << b<<endl;

        swap(&a,&b);                        // call function by passing variableaddresses

        cout<< "\nAfter swapping" <<endl;
        cout<< "a = " << a <<endl;
        cout<< "b = " << b <<endl;
        return 0;
}

// function definition to swap numbers
void swap(int* num1, int* num2)
{       intt;
        t = *num1;
        *num1 = *num2;
        *num2 = t;
}
```

Output:
Beforeswapping
a =10
b =20
After swapping
a =20
b =10

In above program we are using call by reference, when the function is working with reference or address it is actually working with original data.

Return by reference:
A function can also return a reference. A C++ program can be made easier to read and maintain byusingreferencesratherthanpointers.Whenafunctionreturnsareference,itreturnsanimplicit pointer to its return value. This way, a function can be used on the left side of an assignment statement.

Example: C++ program to illustrate use of return by reference.

```
#include<iostream.h>in
t n;
int& test();

int main()
{
        test()=10;
        cout<<n;
        return 0;
}
int& test()
{
        return n;
}
```

In above program return type of function test() is int& hence test() returns by reference. In program, test() will not return value of n, instead it returns reference of the variable n. Since test() is returning address of n it can be assigned a value, in our program it is 10. Hence program will display output: 10.

Inline Function:
When the program executes the function call instruction, the CPU stores the memory address of the instruction following the function call, copies the arguments of the function on the stack and finallytransferscontroltothespecifiedfunction.TheCPUthenexecutesthefunctioncode,stores the function return value in a predefined memory location/register and returns control to the callingfunction.
This can become overhead if the execution time of function is less than the switching time from thecallerfunctiontocalledfunction(callee).Forfunctionsthatarelargeand/orperformcomplex tasks,theoverheadofthe functioncallisusuallyinsignificantcomparedtotheamountoftimethe functiontakestorun.However,forsmall,commonly-usedfunctions,thetimeneededtomakethe

function call is often a lot more than the time needed to actually execute the function's code. This overhead occurs for small functions because execution time of small function is less than the switching time.

C++providesaninlinefunctionfeaturetoreducethefunction calloverhead.Italsosaveoverhead of arguments push/pop on the stack, while function calling. Inline function is a function that is expandedinlinewhenitiscalled.Ifa functionisInline, the compilerplacesa copyofthecode of that function at each point where the function is called at compile time and may make the program executionfaster.

To inline a function, place the keyword inline before the function name and define the function before any calls are made to the function. The compiler can ignore the inline qualifier, in case defined function is more than a line.

Syntax:

```
inline return-type function-name(argument list)
{
        //Function Body
}
```

Example: C++ program to illustrate use of inline function.

```
#include <iostream.h>
inline int square(int x)
{
        return (x*x);
}
int main()
{
        cout<< "Square (2): " <<square(2)<<endl;
        cout<< "Square (3): " <<square(3)<<endl;
        return0;
}
```

Output:

Square (2):4
Square (3):9

Default Arguments:

In C++ programming, we can provide default values for function parameters. A default argument is a value provided in a function declaration for function parameters. If a function with default arguments is called without passing arguments, then the default values are automatically assigned by the compiler during compilation of program. However, if arguments are passed while calling the function, the default arguments are ignored.

Example: C++ program to illustrate use of default arguments.

```
#include<iostream.h>
int sum(int a=10, int b=20);
int sum(int a, int b)
{
        return (a+b);
}
```

```
int main()
{
        cout<<sum()<<endl;
        cout<<sum(50)<<endl;
        cout<<sum(50,50)<<endl;
        return 0;
}
```

Output:

30
70
100

In above program for first function call a=10 and b=20, for second function call a=50 and b=20 and for third function call a=50 and b=50.

Rules for default argument:

- Adefaultargumentischeckedfortypeatthetime ofdeclarationandevaluatedatthetime ofcall.
- Onlytrailingarguments canbedefaultvaluesandthereforeadddefaultsfromrighttoleft.
- We cannot provide default value to a particular argument in the middle of an argument list.

Function declaration with default values:

```
int sum(int a, int b=20,intc=30);          //allowed
int sum(int a=10,intb);                    //notallowed
int sum(int a=10, int b,intc=30);          //notallowed
int sum(int a=10, int b=20,intc=30);       //allowed
```

Default arguments are useful in situation where some arguments always have the same value. For example bank interest may remain same for all customers for a particular period of deposit.

Practice Programs:

1. Write a C++ program to read two float numbers. Perform arithmetic operations like +, -, *, / on these numbers using Inline Function. (Use manipulators)
2. Write a C++ program to store percentage of ‘n’ students and display it where ‘n’ is the number of students entered by the user.(Use new and deleteoperator)
3. Write a C++ program to perform increment and decrement operation on integer number. (Use inlinefunction)

Set A:

1. Write a C++ program to accept length and width of a rectangle. Calculate and display perimeter as well as area of a rectangle by using Inlinefunction.
2. Write a C++ program to define power function to calculate x^y. (Use default value as 2 fory).
3. Write a C++ program to accept and display Bank_Account details as Acc_No, Acc_holder_name, Addr, Contact_Number and Balance. Perform deposit of some amount and display modified bank account details. (Usemanipulators)

Set B:

1. Write a C++ program to accept ‘n’ float numbers, store them in an array and print the alternate elements of an array. (Use dynamic memoryallocation)
2. WriteaC++programtomodifycontentsofanintegerarray.(UseCall byreference)
3. Write a C++ program to calculate area and circumference of a Circle. (Use default argument, scope resolution operator and manipulator.)

Set C:

1. Create a C++ program to maintain inventory of a book having details Title, Authors[], Price, Publisher and Stock. Book can be sold, if stock is available, otherwise purchase will be made. Write a menu driven program to perform followingoperation:
    - Accept bookdetails.
    - Salea book. (Sale contains number of copies to besold.)
    - Purchaseabook. (Purchasecontainsnumberofcopiestobepurchased)

    (Use dynamic memory allocation while accepting authordetails).

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No. 3: Classes and Objects

Class:

A class in C++ is just an extension of a 'structure' used in the 'C' language. Class is a user-defined data type. It actually binds the data and its related functions in one unit, they are called members of the class.

A structure and a class differ a lot as a structure has limited functionality and features as compared to a class. A structure is used to represent a record and a class can have both data members and functions also. C++ expands the role of structure to create a class.

The **Structure and Class**, are almost similar in all respect except the significant one difference that,structurebydefaulthaveallitsmemberas"public",andclassbydefaulthaveallitsmember "private". Both a structure and a class provide a way to create a customized data type which can be used further to create instances. Instance of structure is called 'structure variable' and instance of a class is called'object'.

Object:

An object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

When you define a class, you define a blueprint for a data type. This doesn't actually define any data,butitdoesdefinewhattheclassnamemeans,thatis,whatanobjectofthe classwillconsist of and what operations can be performed on such anobject.

Access Specifiers:

Access specifiers are used to implement an important feature of Object-Oriented Programming known as Data hiding. Access specifiers in a class define how the data members and functions of a class can be accessed. That is, it sets some restrictions on the class members not to get directly accessed by the outside functions. This access restriction to the class members is specified by the labeled public, private, and protected sections within the class body. The keywords public, private, and protected are called access specifiers.

- public - members are accessible from outside the class but within aprogram.
- private - members cannot be accessed or viewed from outside the class. Only the class and friend functions can access privatemembers.
- protected - members cannot be accessed from outside the class, however, they can be accessed in inheritedclasses.

But if we do not specify any access specifier for the members inside the class then by default the access specifier for the members will be private. Member functions of the class can access all the data members and other member functions of the same class (private, public or protected) directly by using their names.

Example: C++ program to demonstrate class, object, access specifiers and defining member function inside class definition.

```
#include<iostream.h>
classSquare                         //class
{
        public:                     //access specifier
        floatside;
        floatarea()                 //member function definition inside theclass
        {       return(side*side);
        }
};
intmain()                                   // mainfunction
{
        Squareobj;                          //object
        obj.side=5.5;               // accessing public data member outside class
        cout<<"Square side length is: " <<obj.side<<"\n";
        cout<< "Area of square is: " <<obj.area();
        return 0;
}
```

Output:

Square side length is: 5.5
Area of square is: 30.25

A class definition starts with the keyword **class** followed by the class name; and the class body, enclosedbya pairofcurlybraces.Aclassdefinitionmustbefollowedbyasemicolonoralistof declarations.

InC++**publickeyword**determinestheaccessattributesofthemembersoftheclassthatfollows it, in above program data member side and member function area arepublic.

A public member can be accessed from outside the class anywhere within the scope of the class object hence side is accessible in main function through object of square class. You can also specify the members of a class as **private** or **protected** as per the need.

Defining member functions inside and outside class definition:

Member functions are the functions, which have their declaration inside the class definition and works on the data members of the class. The definition of member functions can be inside or outside the definition of class. In both the cases, the function body remains the same; however, the function header is different.

Member function definition inside the class definition:

If the member function is defined inside the class definition it can be defined directly. Member function inside the class does not require to be declared first here we can directly define the function. Defining a member function within the class definition declares the function by default **inline**, even if you do not use the inline specifier. Above C++ program is an example of member function definition inside theclass.

Member function definition outside the class definition:

If the member function is defined outside the class, then we have to use the scope resolution operator '::' along with class name and function name. Function name in the function header is preceded by the class name and the scope resolution operator (: :).

The scope resolution operator informs the compiler what class the member belongs to. Defining a member function outside a class requires the function declaration (function prototype) to be provided inside the class definition.

Example:

```
#include<iostream.h>cl
ass Square
{
        public:
                float side;
                float area();
};
float Square::area()                    //member function definition outside theclass
{       return (side*side);
 }
int main()
{
        Square obj;
        obj.side=5.5;
        cout<< "Square side length is: " <<obj.side<< "\n";
        cout<< "Area of square is: " <<obj.area();
        return 0;
}
```

Output:

```
Square side length is: 5.5
Area of square is: 30.25
```

Static data members and Static member functions:

Static data members:

Static data members are class members that are declared using the static keyword. The normal variable is created when the function is called and its scope is limited, while the static variable is created once and destroyed at the end of the program. These variables are visible within the class but its lifetime is till the program ends. There is only one copy of the static data member in the class, even if there are many class objects. This is because all the objects share the static data member. To hold the count of objects created for a class, static data members are used.

The static data member is always initialized to zero when the first class object is created. While definingastaticvariable,someinitialvalue canalsobeinitializedtothevariable.Typeandscope of each static member variable must be defined outside the class definition using scope resolution operator. This is necessary because the static data members are stored separately rather than as a part of anobject.

Static data members are associated with the class itself rather than with any class object, hence they are also known as class variables.

Static member functions:

Like static data member, we can also have static member functions. A static member function can only access other static variables or functions present in the same class. To create a static member function we need to use the static keyword while declaring the function.

Since static member variables are class properties and not object properties, to access them we need to use the class name instead of the object name. A static member function can be called even if no objects of the class exist and the static functions are accessed using class name and the scope resolution operator ::. You could use a static member function to determine whether some objects of the class have been created or not.

Example: C++ program to illustrate use of static data member and static member function.

```
#include <iostream.h>
class StaticDemo
{
        private:
                staticintnum;                   //declaration of static datamember
        public:
                staticvoidDisplay()             //static member functiondefinition
                {
                        cout<<"Value of num is : "<<num<<endl;//accessing static data member
                }
};
int StaticDemo::num=10;      //staticdatamemberdefinitionandinitializationoutsideclass
intmain()
{
        StaticDemo::Display();                  //call tostatic member function
        return0;
}
```

Output:

Value of num is : 10

Array of objects:

An object of class represents a single record in memory, if we want more than one record of class type, we have to create an array of object. An array which contains the class type of element is called array of objects.

Array of objects contains the objects of the class as its individual elements. It is declared in the same way as an array of any built-in data type.

Example: C++ program to illustrate use of array of objects.

```
#include<iostream.h>cl
ass Employee
{
        int Emp_id;
        char Name[20];
        long Salary;
```

```
        public:
        void Accept()
        {
                cout<<"\n\tEnter Employee Id, Name and Salary : ";
                cin>>Emp_id>>Name>>Salary;
        }
        void Display()
        {
                cout<<"\n"<<Emp_id<<"\t"<<Name<<"\t"<<Salary;
        }
};
int main()
{
        int i;
        Employeeemp[3];          //CreatingArrayofobjectstostore3 Employeesdetails
        for(i=0;i<3;i++)
        {
                cout<<"\nEnter details of "<<i+1<<" Employee";
                emp[i].Accept();
        }
        cout<<"\nDetails of Employees";
        for(i=0;i<3;i++)
        emp[i].Display();
        return 0;
}
```

Above program will accept and display details of 3 employees using array of objects.

Objects as a function argument:

In C++ we can pass objects of a class as arguments, the same way how we pass other variables. To pass it we write the object name as the argument while calling the function. Object as function argument is normally used to communicate between two objects.

The objects of a class can be passed as arguments to member functions as well as non-member functions either by value or by reference.

Call by value: When an object is passed by value, a copy of the actual object is created inside the function, to pass entire object into another function. This copy is destroyed when the function terminates. Moreover, any changes made to the copy of the object inside the function are not reflected in the actual object.

Call by reference: In this method, only a reference to that object (not the entire object) is passed to the function. Thus, the changes made to the object within the function are also reflected in the actualobject.

Whenever an object of a class is passed to a member function of the same class, its data members can be accessed inside the function using the object name and the dot operator. However, the data members of the calling object can be directly accessed inside the function without using the object name and the dotoperator.

Function returning objects:

As we can pass entire object as an argument, similarly we can return object from the function. We can return entire object from function by specifying its return type as class name just like primary data-types. An object can be returned by a function using the return keyword.

Friend Function:

Data hiding is a fundamental concept of object-oriented programming. It restricts the access of private members from outside of the class. Similarly, protected members can only be accessed by derived classes and are inaccessible from outside. However, there is a feature in C++called **friend functions** that break this rule and allow us to access **private** and **protected** data of a class outside theclass.

For accessing the data, the declaration of a friend function should be done inside the body of a class starting with the keyword friend. A friend function of a class is defined outside that class' scope but it has the right to access all private and protected members of the class.

Even though the prototypes for friend functions appear in the class definition, friends are not member functions. The function can be defined anywhere in the program like a normal C++ function. The function definition does not use either the keyword friend or scope resolution operator.

Characteristics of a Friend Function:

- Friend function is not in the scope of the class to which it has been declared as afriend.
- It cannot be called using the object as it is not in the scope of thatclass.
- It can be invoked like a normal function without using theobject.
- It cannot access the member names directly and has to use an object name and dot membership operator with the membername.
- It can be declared either in the private or the publicpart.

Example: C++ program to illustrate use of objects as a function argument, function returning object and friend function.

```
# include <iostream.h>
class Demo
{
        int x,y;
        public:
        void Accept();
        // friend function declaration with objects as arguments and returning object
        friend Demo sum (Demo, Demo);
        void Display();
};
Demo sum(Demo obj1, Demo obj2)
{
        Demo obj3;
        obj3.x=obj1.x+obj2.x;
        obj3.y=obj1.y+obj2.y;
        returnobj3;                    //function returningobject
}
```

```
int main()
{
        Demo obj1, obj2, obj3;
        obj1.Accept();
        obj2.Accept();
        obj3=sum(obj1,obj2);        //call to a friendfunction
        obj3.Display();
        return 0;
}

voidDemo::Accept()
{
        cout<<"\nPlease enter value ofx and y :";
        cin>>x>>y;
}

void Demo::Display()
{
        cout<<"x= "<<x<<endl;
        cout<<"y= "<<y<<endl;
}
```

Output:

```
Please enter value of x and y : 1020
Please enter value of x and y : 1020
x=20
y=40
```

Above C++ example give us an idea about the concept of a friend function, but it doesn't show any meaningful use. In the above example, we could have made "sum" as a member function of the class instead of declaring it as a friend function to the class.

A more meaningful use would be operating on objects of two different classes. That's when the friend function can be very helpful. A friend function can act as a bridge between two classes as in the following example.

Example: C++ program to illustrate use of friend function for two classes.

```
#include <iostream.h>

class Square; // forward declaration of a class

class Rectangle
{
        int width, height;
        public:
        void setvalue(int w, int h){width=w; height=h;}
        friend void display(Rectangle &, Square &);
};
```

```
class Square
{
        int side;
        public:
        void setvalue(int s){side=s;}
        friend void display(Rectangle &, Square &);
};

void display(Rectangle &r, Square &s)
{
        cout<< "Rectangle Area: " <<r.width * r.height<<endl;
        cout<< "Square Area: " <<s.side * s.side<<endl;
}
int main ()
{
        Rectangle rec;
        rec.setvalue(5,10);
        Square sq;
        sq.setvalue(5);
        display(rec,sq);
        return 0;
}
```

Output:

Rectangle Area: 50
Square Area: 25

In above program friend function display() is friendly to Rectangle and Square class. It does not belong to any class, so it can be used to access private data of Rectangle and Square class.

Friend Class:

Like friend function, a class can also be a friend of another class. A friend class can access all the private and protected members of other class in which it is declared as friend. This is needed when we want to allow a particular class to access the private and protected members of a class. In order to access the private and protected members of a class into friend class we must pass on object of a class to the member functions of friendclass.

Example: C++ program to illustrate use of friend class.

```
#include <iostream.h>
class A
{
        int num;
        public:
        void setvalue(int i)
        {
                num=i;
        }
```

```
        friendclassB;           //making B class, a friend class of Aclass

};
class B
{
        public:
        void display(A &a)
        {
                cout<<"Value of num is : "<<a.num;
        }
};
int main()
{
        A a_obj;
        a_obj.setvalue(10);
        B b_obj;
        b_obj.display(a_obj);
        return0;
}
```

Output:

Value of x is :10

In the above example, B class is a friend class of A class. In order to access the private members of A class into B class we have explicitly pass an object of A class to the member functions of Bclass.

This is similar to passing an object as function argument but the difference is, an object a_obj we are passing as argument is of different class (A) and the calling object is of different class (B).

Practice Programs:

1. Write a C++ program to create a class Customer with data members ID, Name, Addr and Contact_No. Write member functions to accept and display customer information. (Use scope resolution operator while defining memberfunctions)
2. Write a C++ program to create a class Employee with data members Emp_id, Name, department, date_of_joining and Salary. Write member functions to accept and display details of 'n' employees. (Use array ofobjects)
3. Write a C++ program to add two float numbers of two different classes using friend function.

Set A:

1. Write a C++ program to create a class Student with data members Roll_No, Student_Name, Class. Write member functions to accept and display Student information alsodisplaycountofstudents.(UseStaticdatamemberandStaticmemberfunction)
2. Write a C++ program to calculate the average height of all the students of a class. The numberofstudentsandtheirheightsareenteredbyuser. (Usearrayofobjects)

3. Write a C++ program to calculate maximum and minimum of two integer numbers of two different classes.(Use friendfunction)

Set B:

1. Write a C++ program using class to accept and display ‘n’ Products information, also display information of a product having maximum price. (Use array of objects and dynamic memoryallocation)
2. Write a C++ program to create a class Distance with data members feet and inches. Write member functions for the following:
    a. To acceptdistance
    b. To displaydistance
    c. To add two distanceobjects
    (Use object as a function argument and function returning object)
3. Write a C++ program to create two classes Array1 and Array2 with an integer array as a data member. Write necessary member functions to accept and display array elements of boththeclasses.Findanddisplaymaximumofboththearray.(UseFriendfunction)

Set C:

1. Write a C++ program to calculate multiplication of two integer numbers of two different classes. (Use friend class)

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No. 4: Constructors and Destructors

Constructor:
Aconstructorisa‘special’memberfunctionwhosetaskistoinitializetheobjectsofitsclass.Itis called constructor because it constructs the values of data members of the class. Constructor is automatically called when object of class iscreated.

Characteristics of Constructor:

- Constructors are declared as public memberfunction.
- Constructors are automatically invoked when an object of class iscreated.
- Constructor has same name as the classname.
- Constructors don’t have any returntype.
- Constructors can have defaultarguments.
- Constructorscannotbeinherited,thoughaderivedclasscancallthebaseclassconstructor.
- Constructors cannot bevirtual.
- Constructors cannot refer to theiraddresses.
- Constructors can implicitly call new and delete operators when memory allocation is required.

Constructorscanbedefinedeitherinsidetheclassdefinitionoroutsideclassdefinition. If constructors are defined outside class definition, then they can be defined using class name and scope resolution::operator.

Example: To define Constructor inside the class.

```
class Number
{
        int n;
    public:
        Number()                    //Constructor defined inside theclass
        {
            n=10;
        }
};
```

Example: To define Constructor outside the class.

```
class Number
{
        int n;
    public:
        Number();                   //Constructordeclared
 };
 Number :: Number()                 //Constructor Defined outside theclass
{
        n=10;
 }
```

When a class contains a constructor, objects of the class will be initialized automatically.

Ex. Number Obj1;

Here Obj1 invokes constructor and initializes the data members of class Number.
If constructor is not defined in a class, C++ compiler generates a default constructor.

Types of Constructors:

1. **DefaultConstructors:**

   The constructor that accepts **no arguments** is called as Default Constructor.

   Example: To illustrate the use of Default Constructor.

```
#include<iostream.h>clas
s Number
{
        int n;
   public:
        Number()            //DefaultConstructor
        {
                n = 0;
        }
};
int main( )
{
        Number Obj1;
        return 0;
}
```

   Number Obj1 invokes Default constructor and initializes data member n to 0(zero).

2. **ParameterizedConstructors:**

   The constructor that **accepts arguments** is called as Parameterized constructor. These argumentsinitializeanobject,whenitiscreated.Theconstructorscanbecalledexplicitlyor implicitly.

   If more than one constructor is defined in a class, it is called as **Constructor Overloading.**

   Example: To illustrate the use of Parameterized Constructor.

```
#include<iostream.h>clas
s Number
{
        int n;
    public:
        Number(intx)        //ParameterizedConstructor
        {
                n = x;
        }
};
int main( )
```

```
{
        Number Obj1=Number(50);          // Explicitcall
        NumberObj2(100);                 // Implicitcall
}
```

Number Obj1 & Number Obj2 invokes parameterized constructor and initializes data member n to 50 & 100 respectively.

3. CopyConstructor:

A constructor that **initializes an object** using another object of the same class is called as copy constructor. It takes a reference of object of the same class as its argument. It copies data from one object to other by copying every member of an object with the member of object passed as argument.

Example: To illustrate the use of Copy Constructor.

```
#include<iostream.h>clas
s Number
{
        int n;
  public:
        Number (int x )
        {
                n = x;
        }
        Number(Number&N)          //CopyConstructor
        {
                n= N.n;
        }
};

int main( )
{
        Number Obj1(10), Obj2(Obj1);
        return 0;
}
```

Number Obj2(Obj1) defines the obj2 and at the same time initializes it to values of Obj1.

4. DynamicConstructor:

The constructor can be used to allocate memory while creating objects. Memory can be allocated using new operator. Allocation of memory to objects at the time of their construction is known as dynamic construction of objects.

Example: To illustrate the use of Dynamic Constructor.

```
#include<iostream.h>#include<string
.h>
```

```
class MyString
{
        Char *Str;
        int len;
  public:
        MyString ()
        {
                len=0;
                Str=new char [len+1];
        }
        MyString (char *S)
        {
                len=strlen(S);
                Str=new char [len+1];
                Strcpy(Str, S);
        }
        void Concatenate(MyString &S1, MyString &S2)
        {
                len=S1.len+S2.len;
                delete Str;
                Str=new char [len+1];
                Strcpy(Str, S1.Str);
                Strcat(Str, S2.Str);
                cout<<"String ="<<Str;
        }
};
int main( )
{
        MyString Obj1("Computer"), Obj2("Application"), Obj3;
        Obj3.Concatenate(Obj1, Obj2);
        return 0;
}
```

Constructors with default arguments

It is possible to define constructors with default arguments.

Example: To illustrate the use of Constructors with default arguments.

```
#include<iostream.h>clas
s Number
{
        int m, n;
    public:
        Number(int x, int y=100 )
        {
                m = x;
                n=y;
        }
```

```
};
int main( )
{
        Number Obj1(50);
}
```

Number Obj1 invokes constructor with default arguments and assigns the value 50 to the variable x and 100 to y.

Dynamic initialization of Objects:

Class objects can be initialized dynamically i.e. initial value of an object can be provided during run time. Dynamic initialization is used to provide various initialization formats, using overloaded constructors.

Example: To illustrate the use of Dynamic initialization of Constructor

```
#include<iostream.h>cl
ass Number
{
        int n;
    public:
        Number(int x)
        {
                n = x;
        }
        void display()
        {
                cout<<"n = " <<n;
        }
};
int main( )
{
        int a;
        cout<<"\n Enter the value of a:";
        cin>>a;
        Number Obj(a);
        Obj.display();
        return 0;
}
```

Destructor:

Destructor is a member function that destroys an object which has been created by constructor.Ifnewoperatorisusedtoallocatememoryintheconstructors,deleteoperatorisused to free memory in the destructor. Destructor can clean up the storage which is no longer accessible.
A destructor is invoked implicitly when the object goes out of scope like:

a. the functionends.
b. the programends.

c. a block containing local variablesends.
d. a delete operator iscalled.

Characteristics of Destructor:

- Destructors have same name as the class name preceded by a tilde(~).
- Destructors doesn't take any argument and doesn't return anyvalue.

Example: To illustrate the use of Destructor.

```
#include<iostream.h>clas
s Number
{
    public:
        Number ( )
        {
                cout<<"\n Constructor called";
        }
        ~Number ( )
        {
                cout<<"\n Destructor called";
        }

};
int main( )
{
        Number Obj1;
        {
                Number Obj2;
        }                                   //Destructor Ob2called
        return0;                            //Destructor Ob1called
}
```

Note: Objects are destroyed in the reverse order of creation.

Practice Programs:

1. Write a C++ program to create a class 'MyNumber' with three data members of type integer. Create and initialize the object using default constructor and parameterized constructor. Also define copy constructor to copy one object to another. Write a C++ program to illustrate the use of aboveclass.

2. Write a C++ program to create a class 'Fraction' with integer data members numerator and denominator. Create and initialize the object using parameterized constructor. Write a member function to display addition two fraction objects.(Use the concept of dynamic initialization ofobject)

3. Write a C++ program to create a class 'MyArray' which contains single dimensional integer array of given size. Write a member function to display array in ascending order. (UseDynamicConstructortoallocateandDestructortofreememoryofanobject)

Set A:

1. Write a C++ program to create a class 'MyNumber' with three data members of type integer. Create and initialize the object using default constructor, parameterized constructor and parameterized constructor with default value. Write a member function to display average of given three numbers for allobjects.

2. Write a C++ program to create a class MyDate with three data members as dd, mm, yyyy. Create and initialize the object by using parameterized constructor and display date in dd-mon-yyyy format. (Input: 19-12-2014 Output: 19-Dec-2014).(Use the concept of dynamic initialization ofobject)

3. WriteaC++programtocreatea class'MyPoint'withtwointegerdatamembersasx& y. Define copy constructor to copy one object to another. (Use Default and parameterized constructor to initialize the appropriate objects) Write a C++ program to illustrate the use of aboveclass.

Set B:

1. Write a C++ program to create a class 'MyArray' which contains single dimensional integer array of given size. Write a member function to display even and odd numbers from a given array. (Use Dynamic Constructor to allocate and Destructor to free memory of anobject)

2. Write a C++ program to create a class 'MyMatrix' which contains two dimensional integer array of size mXn. Write a member function to display sum of all elements of entered matrix. (Use Dynamic Constructor for allocating memory and Destructor to free memory of anobject)

3. Write a C++ program to create a class 'MyVector' with data members size & a pointer to integer. The size of the vector varies so the memory should be allocated dynamically. Create and initialize the object using default and parameterized constructor. Write a member function to display the vector in the format (10, 20,30,….)

Set C:

1. Create a C++ class 'Student' with data members Rollno, Name,Number of subjects, Marks of each subject (Number of subjects varies for each student). Writea parameterized constructor which initializes rollno, name & Number of subjects and creates the array of marksdynamically.Displaythedetailsofallstudentswithpercentageandclassobtained.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No. 5: Inheritance

Inheritance:
The mechanism of deriving a new class from an old class is called as **Inheritance**.
Inheritance allows a derived class to inherit the properties and characteristics from base class. A classcanalsoinheritpropertiesfrommorethanoneclassorfrommorethanonelevel.Inheritance supports the reusability as inheritance can extend the use of existing classes and eliminate redundantcode.

The class that inherits the properties from another class is called Sub class or **Derived Class**.
Theclasswhosepropertiesareinheritedbyderivedclassis calledSuperclassor**BaseClass.**
Syntax to define derived class:

```
class Derived_class_name : visibility_mode Base_class_name
        {
                //body of Derived class
        };
```

Where,
Derived_class_name is the name of the sub class/derived class.
visibility_mode specifies the mode in which derived class can be inherited. For example: public, private, protected. Default visibility mode is private.
Base_class_name is the name of the base class from which you want to inherit the sub class.

Modes of Inheritance

The following table represents the scope of the access specifier of the members of base class in the derived class when derived in private, public & protected modes:

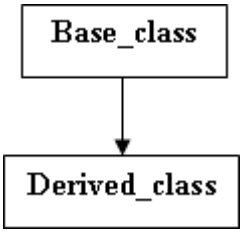
		Derived Class		
		private Mode	public Mode	protected Mode
Base Class Access specifiers	private	Not inherited	Not inherited	Not inherited
	public	private	public	protected
	protected	private	protected	protected

- **Private mode**: If a sub class is derived from a base class in private mode then both public memberandprotected membersofthebaseclassbecomesPrivateinderivedclass.Private members of the base class never get inherited in subclass.
- **Public mode**: If a sub class is derived from base class in public mode then the public memberofthebase classremainspublicinthederivedclassandprotectedmembersofthe base class remains protected in derived class. Private members of the base class never get inherited in subclass.
- **Protected mode**: If a sub class is derived from a base class in protected mode then both public member and protected members of the base class becomes protected in derived class. Private members of the base class never get inherited in subclass.

Types of Inheritance:

1. SingleInheritance:

A derived class with **only one base class** is called as Single Inheritance.



Syntax to define derived class:

```
class Derived_class: visibility_mode Base_class
{
      //Body of Derived class
};
```

Example: To illustrate the use of Single Inheritance using public derivation.

```
#include<iostream.h>cl
ass Base
{
        int x;                          //private; not inheritable
    public:
        int y;                          //public;inheritable
        void setValues()
        {
                x=10;
                y=20;
        }
        int getx()
        {
                return x;
        }
}
class Derived :publicBase               // publicderivation
{
        int z;
   public:
        void add()
        {
                Z=getx() + y;
        }
        void display()
        {
                cout<<"\n x= "<<getx();
                cout<<"\n y= "<<y;
                cout<<"\n Addition : "<<z
```

```
        }
}
int main()
{
        Derived D;
        D.setValues();
        D.add();
        D.display();
}
```

Derivedclassisapublicderivationofthebaseclass Base.So,Derivedclassinheritsallthepublic members of class Base and retains their visibility. Thus public members of the Base class are also public members of the Derived class. The private members of the Base class cannot be inherited by classDerived.

Example: To illustrate the use of Single Inheritance using private derivation.

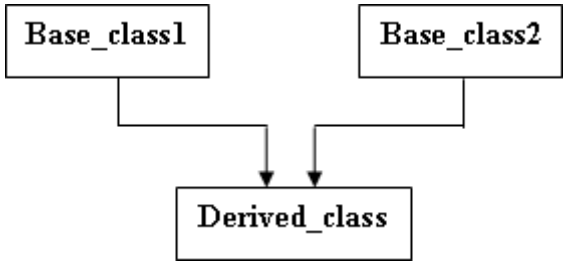
```
#include<iostream.h>cl
ass Base
{
        int x;                          //private; not inheritable
    public:
        int y;                          //public; ready forinheritance
        void setValues()
        {
                x=10;
                y=20;
        }
        int getx()
        {
                return x;
        }
}
class Derived :privateBase              // privatederivation
{
        int z;
   public:
        void add()
        {
                D.setValues();
                z=getx() + y;
        }
        void display()
        {
                cout<<"\n x= "<<getx();
                cout<<"\n y= "<<y;
                cout<<"\n Addition : "<<z
```

```
        }
}
int main()
{
        Derived D;
        //D.setValues();                wont work
        D.add()
        D.display();
}
```

Derived class is a private derivation of the base class Base. So, Derived class inherits only public members of base class Base as private and retains their visibility. The private members of the Base class cannot be inherited by class Derived.

2. MultipleInheritance:

A derived class with **several base classes** is called as Multiple Inheritance.



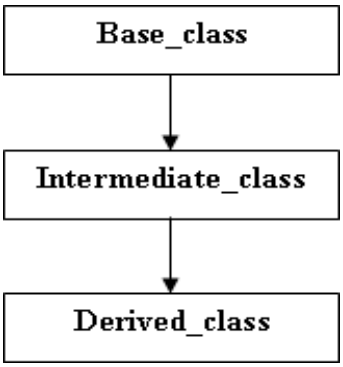
Syntax to define derived class:

```
class Derived_class : visibility_mode Base_class1, visibility_mode Base_class2,..
{
        //Body of Derived class
};
```

A class is derived with multiple base classes. The number of base classes are separated by a comma (‘, ‘). Visibility mode for every base class must be specified.

3. MultilevelInheritance:

The mechanism of **deriving a class from another derived class** is called as Multilevel inheritance.

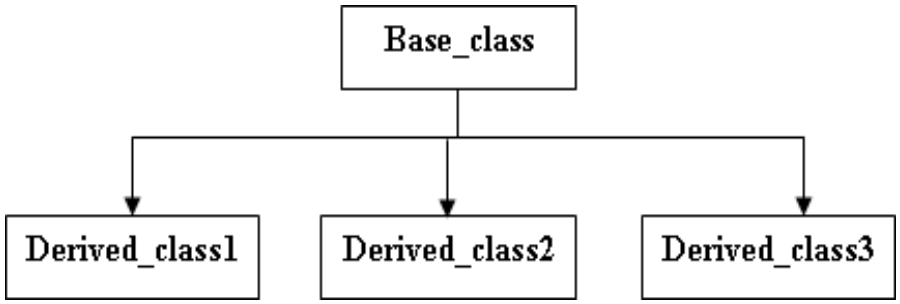


Syntax to define derived class:

```
class Intermediate_class : visibility_mode Base_ class
{
      //Body of Intermediate class
};
class Derived_class : visibility_mode Intermediate_class
{
      //Body of Derived class
};
```

4. HierarchicalInheritance:

More than one derived classes inherits the features from a **single base class** is called as Hierarchical Inheritance i.e. more than one derived classes are created from a single base class.



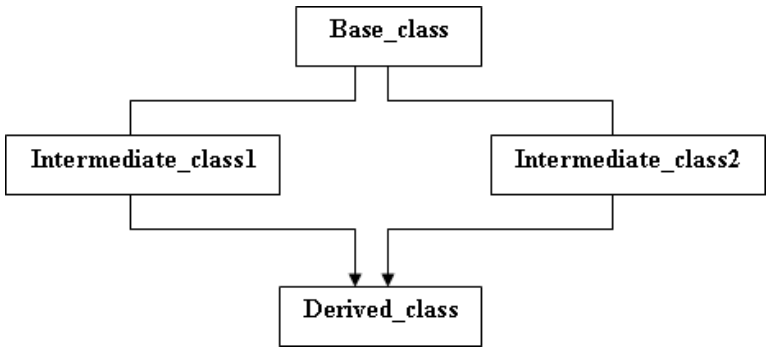
Syntax to define derived class:

```
class Derived_class1 : visibility_mode Base_class
{
      //Body of Derived class
};
class Derived_class2 : visibility_mode Base_class
{
      //Body of Derived class
};
class Derived_class3 : visibility_mode Base_class
{
      //Body of Derived class
};
```

5. HybridInheritance:

More than one type of inheritance is combined to form Hybrid Inheritance.

For Ex.: Combination of Hierarchical inheritance and Multiple Inheritance.



Syntax to define derived class:

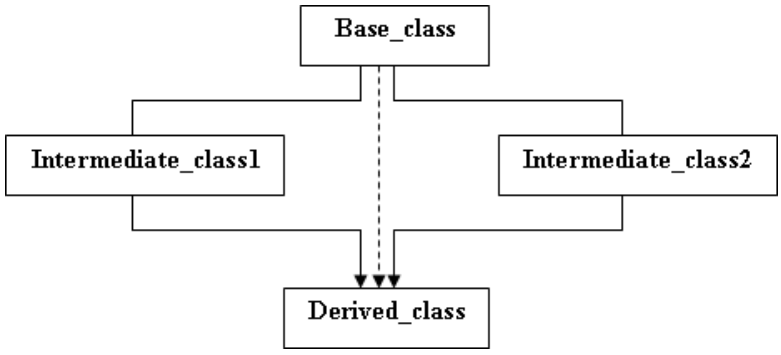
class **Intermediate_class1** : **visibility_mode** Base_ class
{
//Body of Intermediate class1
};
class **Intermediate_class2** : **visibility_mode** Base_ class
{
//Body of Intermediate class2
};
class **Derived_class** : **visibility_mode** Intermediate_class1, **visibility_mode** Intermediate_class2
{
//Body of Derived class
};

Virtual baseclass:

Severalpathsexisttoa derivedclassfromthesamebaseclassi.e.a derivedclasscanhave duplicate sets of members inherited from a single base class. This introduces **ambiguity and it should beavoided.**

Duplication ofinheritedmembersduetomultiplepathsisavoidedbymakingthecommon base class as **virtual base class**. This is achieved by preceding the base class name with the keyword**virtual**.

When a class is made a virtual base class, necessary care is taken so that only one copy of that class is inherited, regardless of the number of paths exist between virtual base class and a derived class.



Syntax:

```
class Base_class
{
        //Body of Base class
 };
class Intermediate_class1 : virtual visibility_mode Base_ class
{
         //Body of Intermediate class1
};
class Intermediate_class2 : visibility_mode virtual Base_ class
{
         //Body of Intermediate class2
};
class Derived_class : visibility_mode Intermediate_class1, visibility_mode Intermediate_class2
{
         //Body of Derived class
};
```

Note: Virtual and visibility mode can be used in either order.

Example: To illustrate the use of virtual base class.
Calculating marks and grade of student using virtual base class.

```
#include<iostream.h>
#include<conio.h>
#include<string.h>in
tn;
class student
{
  protected:
        int rno;
        char name[20];
  public:
        void acceptinfo()
        {
                cout<<"\nRoll no: ";
                cin>>rno;
                cout<<"Name: ";
                gets(name);
        }
        void displayinfo()
        {
                cout<<"\nRoll no: "<<rno <<"\nName: "<<name;
        }
};
```

```
class test:public virtual student
{
 protected:
        int marks1, marks2;
 public:
        void acceptmark()
        {
                cout<<"Mark 1: ";
                cin>>marks1;
                cout<<"Mark 2: ";
                cin>>marks2;
        }
        void displaymark()
        {
                cout<<"\nMark 1: "<<marks1<<"\nMark 2: "<<marks2;
        }
};
class sport : public virtual student
{
 protected:
        int score;
 public:
        void acceptscore()
        {
                cout<<"Score:";
                cin>>score;
        }
        voiddisplayscore()
        {
                cout<<"\nScore: "<<score;
        }
};
class result:public test,public sport
{
 protected:
        int totalmarks, perc;
        char grade[20];
 public:
        void calctotal();
        void accept()
        {
                acceptinfo();
                acceptmark();
                acceptscore();
                calctotal();
        }
```

```
        void display()
         {
                displayinfo();
                displaymark();
                displayscore();
                cout<<"\nTotal: "<<totalmarks
                        <<"\nPercentage: "<<perc<<" %"
                        <<"\nGrade: "<<grade<<"\n";
         }
        int gettotal()
         {
                return totalmarks;
         }
};
void result::calctotal()
{
         totalmarks=marks1+marks2+score;
         perc=(totalmarks*100)/300;
         if(perc>=75)
                strcpy(grade,"Distinction");
         else if(perc>=60 && perc <75)
                strcpy(grade,"First Class");
         else if(perc>=35 && perc<60)
                strcpy(grade,"Pass Class");
         else
                strcpy(grade,"Fail");
}

int main()
{
         int ch, i, j;
         clrscr();
         do{

                cout<<"\nMain Menu\n"
                        <<"\n1. Accept details"
                        <<"\n2. Display details in ascending order"
                        <<"\n3. Exit\n"
                        <<"\nEnter your option: ";
                cin>>ch;
                switch(ch)
                {
                        case 1: cout<<"\nDetails of how many students do you want ot enter: ";
                                cin>>n;
                                result r[10];
                                cout<<"\nEnter the following details";
```

```
                    for(i=0; i<n; i++)
                    {
                        r[i].accept();
                    }
                    break;
            case 2: cout<<"\nThe details are\n";
                    for(i=0; i<n; i++)
                    {
                        r[i].display();
                    }
                    getch();
                    break;
            case 3: exit(0);
        }
    }while(ch!=3);
    return 0;
}
```

Abstract class:

An abstract class is **not used to create objects**. An abstract class is designed only to acts as a base class.

Constructor in derived class:

While using constructors during inheritance, is that, as long as a base class constructor doesnottakeanyarguments,thederivedclassneednothaveaconstructorfunction.However,ifa base class contains a constructor with one or more arguments, then it is mandatory forthe derived class to have a constructor and pass the arguments to the base class constructor. While applying inheritance, we usually create objects using derived class. Thus, it makes sense for the derived class to pass arguments to the base class constructor. When both the derived and base class containsconstructors,thebaseconstructorisexecutedfirstandthentheconstructorinthederived class isexecuted.

In case of multiple inheritance, the base class is constructed in the same order in which they appear in the declaration of the derived class. Similarly, in a multilevel inheritance, the constructor will be executed in the order of inheritance.

The derived class takes the responsibility of supplying the initial values to its base class. The constructor of the derived class receives the entire list of required values as its argument and passesthemontothebaseconstructorintheorderin whichtheyaredeclaredinthederived class. A base class constructor is called and executed before executing the statements in the body of the derivedclass.

Syntax to define derived class constructor:

```
Derived-Constructor (ArgList2, ArgList2,…….ArgListN,ArgListD): Base1(ArgList1),
Base2(ArgList2)……. BaseN(ArgListN)
{
    // Body of Derived Constructor
}
```

Practice Programs:

1. Create a base class Employee(empcode, empname). Derive the classes Manager(designation, club_dues), Scientist(deptname, publications) and Labourer from Employee class. Write a C++ menu drivenprogram
   i. to accept the details of 'n'employees
   ii. to display theinformation
   iii. to display details of manager with designation as "GeneralManger".

2. Create two base classes Learning_Info(Roll_No, Stud_Name, Class, Percentage) and Earning_Info(No_of_hours_worked, Charges_per_hour). Derive a class Earn_Learn_info from above two classes. Write necessary member functions to accept and display Student information. Calculate total money earned by the student. **(Use constructor in derived class)**

Set A:

1. DesignabaseclassProduct(Product_Id,Product_Name,Price).DeriveaclassDiscount (Discount_In_Percentage) from Product. A customer buys 'n' products. Write a C++ program to calculate total price, totaldiscount.

2. DesignaBaseclassCustomer(name,phone-number).DeriveaclassDepositor(accno, balance) from Customer. Again derive a class Borrower (loan-no, loan-amt) from Depositor. Write necessary member functions to read and display the details of 'n' customers.

Set B:

1. Design two base classes Personnel (name, address, email-id, birth date) and Academic (marksintenth,marksintwelth,classobtained).DeriveaclassBio-data frombothese classes. Write a C++ program to prepare a bio-data of a student having Personnel and Academicinformation.

2. Design a base class Employee (emp-code, name). Derive two classes as Fulltime (daily rate, number of days, salary) and Parttime (number of working hours, hourly rate, salary) from Employee. Write a C++ menu driven programto:
   i. Accept the details of 'n' employees and calculate thesalary.
   ii. Display the details of 'n'employees.
   iii. Search a givenEmployee.

3. Create a base class Student(Roll_No, Name) which derives two classes Academic_Marks(Mark1, Mark2, Mark3) and Extra_Activities_Marks(Marks). Class Result(Total_Marks, Grade) inherits both Academic_Marks and Extra_Activities_Marks classes. (Use Virtual BaseClass)
   Write a C++ menu driven program to perform the following functions:
   i. Build a mastertable.
   ii. Calculate Total_marks andgrade.

Set C:

1. Create a base class Student(Roll_No, Name, Class) which derives two classes Internal_Marks(IntM1, IntM2, IntM3, IntM4, IntM5) and External_Marks(ExtM1 ExtM2, ExtM3, ExtM4, ExtM5). Class Result(T1, T2, T3, T4, T5) inherits both Internal_Marks and External_Marks classes. (Use Virtual BaseClass)
   Write a C++ menu driven program to perform the following functions:
   i. To Accept and display studentdetails
   ii. Calculate Subject wise total marksobtained.
   iii. Check whether student has passed in Internal and External Exam of each subject. Also check whether he has passed in respective subject or not and display result accordingly.

Assignment Evaluation

0: Not Done[]	1: Incomplete[]	2: Late Complete []
3: Needs Improvement[]	4: Complete[]	5: Well Done []

Signature of Instructor

Assignment No. 6: Polymorphism

Polymorphism:
Polymorphism means 'One name, multiple forms'.

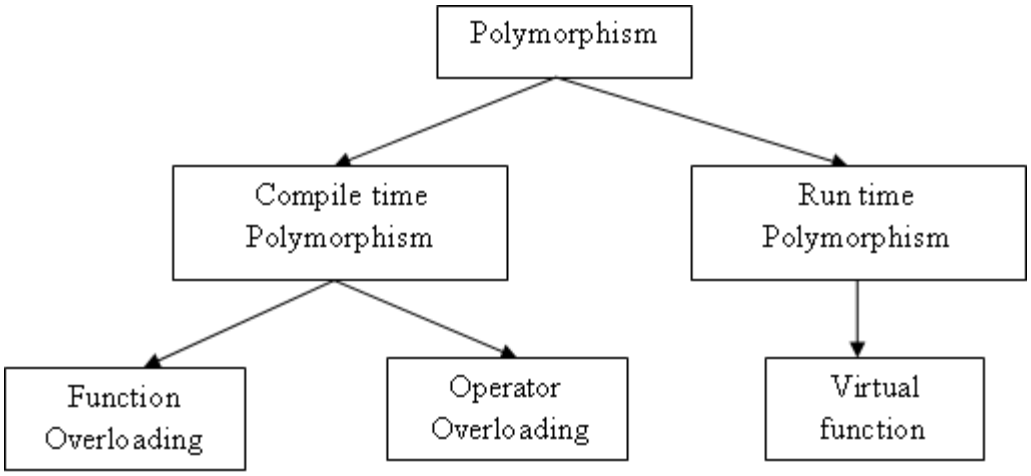


Fig.: Achieving Polymorphism

Compile time Polymorphism:

Compiler selects appropriate function for a particular call at compile time is called as **Compile time Polymorphism.** It is also called as early binding or static binding or static linking. Compile time Polymorphism is achieved by using function overloading and operator overloading.

Function Overloading:

Same function name is used to create a function that perform variety of different tasks is called as Function Overloading.

A family of functions can be designed with same function name but with different argument list. The function performs different operations depending on the argument list in the function call. The appropriate function to be invoked is determined by checking the number and type of arguments.

Example: C++ **program to find volume of cube, cylinder and rectangle using function overloading.**

```
#include<iostream.h>
#include<conio.h>int
volume(int);
double volume(double,int);
long volume(long,int,int);
int main()
{
        clrscr();
        int x,s,h,b;
```

```
        double y,r;
        long z,l;
        cout<<"\nEnter the value for s: ";
        cin>>s;
        x=volume(s);
        cout<<"\nVolume of cube: "<<x;

        cout<<"\nEnter the value for r and h: ";
        cin>>r>>h;
        y=volume(r,h);
        cout<<"\nVolume of cylinder: "<<y;

        cout<<"\nEnter the value for l,b and h: ";
        cin>>l>>b>>h;
        z=volume(l,b,h);
        cout<<"\nVolume of rectangle: "<<z;
        getch();
        return 0;
}
int volume(int s)
{
        return (s*s*s);
}
double volume(double r,int h)
{
        return (3.14*r*r*h);
}
long volume(long l,int b,int h)
{
        return (l*b*h);
}
```

Example: C++ program to find area of triangle, circle, and rectangle using function overloading.

```
#include<iostream.h>#i
nclude<conio.h> float
area(float r)
{
        return(3.14*r*r);
}

float area(float b,int h)
{
        return(0.5*b*h);
}
```

```
int area(int l,int b)
{
        return(l*b);
}
void disp(float m)
{
        cout<<"\nArea: "<<m;
}
int main()
{
        clrscr();
        int  m,n;
        floatl,a;
        cout<<"CIRCLE:\n";
        cout<<"Enter the Radius:";
        cin>>l;
        a=area(l);
        disp(a);
        cout<<"\n\nTRIANGLE\n:";
        cout<<"EntertheBaseandHeight:";
        cin>>l>>m;
        a=area(l,m);
        disp(a);
        cout<<"\n\nRECTANGLE:\n";
        cout<<"EntertheLengthandBreadth:";
        cin>>m>>n;
        a=area(m,n);
        disp(a);
        getch();
        return 0;
}
```

Operator Overloading:

In **Operator Overloading,** an operator is overloaded to give user defined meaning to it. Overloaded operator is used to perform operation on user-defined data type. Although semantics ofanoperatorcanbeextended,butcannotchangeitssyntax,thegrammaticalrulesthatgovernits use such as the number of operands, precedence andassociativity.

For example '+' operator can be overloaded to perform addition on various data types, like for integer, float etc.

Operator overloading is achieved using the **operator function**. The operator function is created using **operator** keyword.

Syntax of operator function:

```
returntype classname :: operator Op (argument List)
{
        //Function Body
}
```

where, **returntype** is the type of value returned by the specified operation.
op is the operator being overloaded. op is preceded by the keyword **operator**. 'operator op' is the function name.
The **argument list** will depend on whether the operator is unary or binary and whether the function is a member function or friend function.

The operator function can be either member function or friend function.

- A friend function will have one argument for unary operators and two for binary operators.
- A member function has no arguments for unary operators and only one for binary operators because the object used to invoke the member function is passed implicitly and therefore is available for memberfunction.

Restrictions on Operator overloading while implementing operator overloading:

1. Precedence and Associativity of an operator cannot bechanged.
2. Arity(numbersofOperands)cannotbechanged.Unaryoperatorremainsunary,binary remains binaryetc.
3. No new operators can be created, only existing operators can beoverloaded.
4. Cannotredefinethemeaningofaprocedure.Youcannot changehowintegersareadded.
5. There are few operators in C++ that cannot be overloaded suchas
    - ternary operator?:,
    - sizeof,
    - scope resolution operator::
    - membership operators . and .*.

Overloading Unary operators:
Unary operators are Increment, Decrement and unary minus which can be overloaded.

Example: To overload the operator unary minus- to negate the numbers.

```
#include<iostream.h>#i
nclude<conio.h> class
Numbers
{
        int x;
        int y;
   public:
        void accept(int a,int b)
        {
                x=a;
                y=b;
        }
        void display()
        {
                cout<<"x="<<x<<"\n";
                cout<<"y="<<y<<"\n";
        }
```

```
        void operator -()
        {
                x=-x;
                y=-y;
        }
};
int main()
{
        Numbers N;
        clrscr();
        N.accept(10,20);
        N.display();
        -N;
        cout<<"\nAfter unary minus handled variable are:"<<"\n";
        N.display();
        return(0);
}
```

Overloading Unary operators using friend function:

Example: To overload operator unary minus- to negate the numbers using friend function.

```
#include<iostream.h>#i
nclude<conio.h> class
Numbers
{
        int x;
        int y;
   public:
        void accept(int a,int b)
        {
                x=a;
                y=b;
        }
        void display()
        {
                cout<<"x="<<x<<"\n";
                cout<<"y="<<y<<"\n";
        }
        friend void operator -(Numbers &Obj)
        {
                Obj.x=-Obj.x;
                Obj.y=-Obj.y;
        }
};
```

```
int main()
{
        Numbers N;
        clrscr();
        N.accept(10,20);
        N.display();
        operator –(N);
        cout<<"\nAfter unary minus handled variable are:"<<"\n";
        N.display();
        return(0);
}
```

Overloading increment operator:

The increment operator ++ is used in two ways: pre-increment (++d) and post-increment(d++). To distinguish between pre and post increment operator overloading, dummy parameter of type int in the function heading of the post-increment operator function is used. Decrement operator can be overloaded similarly.

```
void operator++()
{
        ++x;
        ++y;
}
void operator++(int)
{
        x++;
        y++;
}
```

Overloading Binary operators:

Arithmetic operators are most commonly used operator in C++. Almost all arithmetic(+, - , * ,/) operators are overloaded to perform arithmetic operation on user-defined data type.

Example: To Overload Binary '+' operator using member function

```
#include<iostream.h>#i
nclude<conio.h>
classNumbers
{
        int x;
        public:
                void accept(int a)
                {
                        x=a;
                }
                void display()
                {
                        cout<<"x="<<x<<"\n";
                }
```

```
        Numbers operator +(Numbers Obj)
        {
            Numbers temp;
            temp.x=x+Obj.x;
            return temp;
        }
};
int main()
{
    clrscr();
    Numbers N1,N2,N3;
    N1.accept(100);
    N2.accept(200);
    cout<<"\nFirst number:";
    N1.display();
    cout<<"\nSecond number:";
    N2.display();
    cout<<"\nOperations:\n\n";
    cout<<"\nAddition:";
    N3=N1+N2;
    N3.display();
    getch();
    return(0);
}
```

Example: To Overload Binary operator '+' using friend function.

```
#include<iostream.h>#i
nclude<conio.h>
classNumbers
{
    int x;
    public:
        void accept(int a)
        {
            x=a;
        }
        void display()
        {
            cout<<"x="<<x<<"\n";
        }
        friend Numbers operator +(Numbers Obj1,Numbers Obj2)
        {
            Numbers temp;
            temp.x=Obj1.x+Obj2.x;
            return temp;
```

```
                }
};
int main()
{
        clrscr();
        Numbers N1,N2,N3;
        N1.accept(100);
        N2.accept(200);
        cout<<"\nFirst number:";
        N1.display();
        cout<<"\nSecond number:";
        N2.display();
        cout<<"\nOperations:\n\n";
        cout<<"\nAddition using friend function:";
        N3=operator+(N1,N2);
        N3.display();
        getch();
        return(0);
}
```

Overloading insertion and extraction operators:

Overloading insertion(<<) operator and extraction (>>) operator is used to input and output objects using stream class library in the similar way as built in data types. <<operator is overloaded with **ostream** class object **cout** to print primitive type value to the screen. Similarly <<operator is overloaded in class to print user-defined type to screen. >>operator is overloaded with **istream** class object **cin** to read primitive type values from the user. Similarly >>operator is overloaded in class to read user-defined type value.

Example: To Overload <<&>> operator.

```
        #include<iostream.h>
        #include<conio.h>
        #include<fstream.h>
        class Numbers
        {
                int x;
          public:
                friend ostream& operator <<(ostream &,Numbers &);
                friend istream& operator >>(istream &,Numbers &);
        };
        ostream & operator <<(ostream &out,Numbers &d)
        {
                out<<"\nValue of x:"<<d.x;
                return out;
        }
        istream & operator >>(istream &in,Numbers &d)
        {
```

```
        in>>d.x;
        return in;
    }
    int main()
    {
        Numbers N;
        cout<<"Input";
        cin>>N;              //invokes operator >>( )function
        cout<<"Output";
        cout<<N;             //invokes operator <<( )function

        getch();
        return 0;
    }
}
```

String manipulation using Operator Overloading:

Relational operators like ==, > ,>=, <, <=, !, != are used to compare two user-defined objects.

Example: To compare two strings are equal or not

```
#include<iostream.h>#i
nclude<conio.h>
#include<string.h>
classmystring
{
    char str[30];
    int len;
    public:
        mystring(char *s)
        {
            strcpy(str,s);
        }
        int operator ==(mystring ms)
        {
            if(strcmp(str,ms.str)==0)
                return 0;
            else
                return 1;
        }
};
int main()
{
    char s1[10],s2[10];
    clrscr();
    cout<<"Enter first string"<<"\n";
```

```
        cin>>s1;
        cout<<"Enter secondstring"<<"\n";
        cin>>s2;
        mystring obs1(s1),obs2(s2);
        if(obs1==obs2)
        cout<<"\nGiven strings are not same"<<"\n";
        else
        cout<<"\nGiven strings are same"<<"\n";
        getch();
        return(0);
}
```

Run time Polymorphism:

An appropriate member function is selected for a particular call while the program is running (at run time) is called as **Run time Polymorphism.** It is also called as late binding or dynamic binding or dynamic linking. **Run time Polymorphism is achieved by using Virtual Function**.

this pointer:

Keyword this is used to represent an object that invokes a member function. this is a pointer that pointstotheobjectfor whichthisfunctionwascalled.Thisuniquepointerisautomaticallypassed to a member function when it is called. The pointer 'this' acts as an implicit argument to all the memberfunctions.

Example1: To illustrate the use of this pointer.

```
#include<iostream.h>cl
ass Test
{
        int x;
public:
        void setX (int x)
        {
                this->x = x;
          }
        void print() { cout << "x = " << x << endl; }
};
int main()
{
        Test obj;
         int x = 20;
         obj.setX(x);
         obj.print();
        return 0;
}
```

Example2: To illustrate the use of this pointer.

```
#include<iostream.h>#i
nclude<conio.h> class
Test
{
        int x;
  public:
        Test(int x)
        {
                this->x = x;
        }
        Test& maximum(Test& T)
        {
                if(T.x >= x)
                        return T;
                else
                        return * this;
         }
        void print() { cout << "x = " << x << endl; }
};

int main()
{
  Testobj1(50),obj2(30);
  obj1.print();
  obj2.print();
  Testobj3=obj1.maximum(obj2);
  obj3.print();
  getch();
  return 0;
}
```

Note: return * this will return the object that invoked the function.

Virtual Function:

When same function name is used in both the base and derived classes, the function in baseclassdeclaredasvirtualusingthekeywordvirtual precedingitsnormaldeclarations.Whena function is made virtual, C++ determines which function to use at run time based on the type of object pointed to by the base pointer, rather than the type of the pointer. By making the base pointertopointtodifferentobjects,differentversionsofvirtualfunctionscanbeexecuted.

RuntimePolymorphismisachievedonlywhenaVirtualFunctionisaccessedthrougha pointer to the baseclass.

Example: To illustrate the use of virtual function.

```
class Base
{
   public:
```

```
        void Display()
        {
                cout<<"\n Display Base";
        }
        virtual void show()
        {
                cout<<"\n Show Base";
        }

};
class Derived: public Base
{
   public:
        void Display()
        {
                cout<<"\n Display Derived";
        }
        void show()
        {
                cout<<"\n Show Derived";
        }
};

int main()
{
        Base B;
        Derived D;
        Base*Bptr;

        Bptr =&B;
        Bptr->Display();            //Calls Baseversion
        Bptr->Show();               //Calls Baseversion

        Bptr = &D;
        Bptr->Display();            //Calls Baseversion
        Bptr->Show();               //Calls Derivedversion

        return(0);
}
```

Note: When Bptr is pointing to derived class object D, the statement

Bptr->Display();

calls only the function associated with Base; whereas the statement

Bptr->Show();

calls the Derived version of Show(). Because Show() function from the base class is declared as virtual.

Pure virtual function:

A virtual function equaled to zero is called as pure virtual function It is also called as **"do-nothing"** function. It is a function declared in a base class that has no definition relative to the base class.

Syntax:

virtual void display()=0;

A class containing such pure function is called as an **abstract class**.

Practice Programs:

1. WriteaC++program tosortintegerandfloatarrayelementsinascendingorderbyusing function overloading.

2. Create a class College containing data members as College_Id, College_Name, Establishment_year, University_Name. Write a C++ program with following member functions:
   i. To accept 'n' Collegedetails
   ii. To display College details of a specifiedUniversity
   iii. TodisplayCollegedetailsaccordingtoaspecifiedestablishmentyear (Use Array of Object and Functionoverloading)

3. Create a class Fraction containing data membersas Numerator and Denominator. Write a C++ program to overload operators ++, -- and * to increment, decrement a Fraction and multiply two Fraction respectively. (Use constructor to initialize values of an object).

4. Create a base class Conversion. Derive three different classes Weight (Gram, Kilogram), Volume(Milliliter, Liter), Currency(Rupees, Paise) from Conversion class. Write a C++ programtoperformread,convertanddisplayoperations.(UsePurevirtualfunction)

Set A:

1. Write a C++ program to calculate area of cone, sphere and circle by using function overloading.

2. Create a C++ class Employee with data members E_no, E_Name, Designation andSalary. Accept two employees information and display information of employee having maximum salary. (Use thispointer)

3. Write a C++ program to create a class Integer. Write a C++ program to implement necessarymemberfunctionstooverloadtheoperatorunarypreandpostdecrement'--'for an integernumber.

4. Create a C++ class Integer that contains one integer data member. Overload following binary operators (+,-,*,/).

5. ConsideraclassPointcontainingxandycoordinates.WriteaC++programtoimplement necessary functions to accept a point, to display a point and to find distance between two points using operator overloading (-). (Use friendfunction)

Set B:

1. Create class Person which contains data member as Passport_Id, Person_name, Nationality,Gender,Date_of_Birth,Date_of_Issue,Date_of_expiry.Writeac++program to perform following memberfunctions:
   i. Enter details of allpersons
   ii. Display passport details of oneperson
   iii. Display passport details of allpersons
   (Use Function overloading and Array of object).

2. Create a class Date with members as dd, mm, yyyy. Write a C++ program for overloading operators >> and <<to accept and display aDate.

3. Create a class MyString which contains a character pointer (using new operator). Write a C++ program to overload followingoperators:
   i. < To compare length of twostrings
   ii. != To check equality of twostrings
   iii. + To concatenate twostrings

4. Create a base class Shape. Derive three different classes Circle, Rectangle and Triangle from Shape class. Write a C++ program to calculate area of Circle, Rectangle and Triangle. (Use pure virtual function).

Set C:

1. Create a class MyString which contains a character pointer (Use new and delete operator).Write a C++ program to overload followingoperators:
   i. ! To change the case of each alphabet from givenstring
   ii. [] To print a character present at specified index

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No. 7: Managing Console I/O Operations

Stream is a sequence of bytes. It represents a device on which input and output operations are performed. C++ provides standard iostream library to operate with streams. The iostream is an object-oriented library which provides Input/Output functionality using streams. C++ stream classes are as follows:

I/O Stream	Meaning	Description
ios	General Input/Output Stream Class	It contains basic facilities that are used by all other input & output classes. Declares constants & functions for handling formatted input & output operations
istream	Input Stream	Inherits properties of ios. It reads and interprets input. Declares input functions get(), getline() and read().Contains overloaded extraction operator >>.
ostream	Output Stream	Inherits properties of ios. It can write sequences of characters and represents other kinds of data. Declares output functions put()and write(). Contains overloaded insertion operator<<.
iostream	Input / Output Stream	Inherits properties of ios istream and ostream & contains all input & output functions.
streambuf	File Stream Base	Provides an interface to physical devices through buffers & acts as a base for filebuf class used ios files

Unformatted I/O Operations:
We have used objects cin & cout which are predefined in iostream file for input & output of various types. cin is an object of type istream & cout is an object of type ostream.
We read data from keyboard using following format:
cin>>variable1>> variable2>>……>> variableN
We write data or display it on screen using following format:
cout<<variable1<<variable2<< .....<<variable

Example: Illustrate use of cin & cout statements.
```
int rno;
cin>>rno;
cout<<"Roll No:"<<rno<<endl;
```
In this we have to study different functions of istream class and of ostream class.

istream class functions:
int get()-Accepts a character from input screen & returns it.
istream &get(char &ch)- Accepts a character from input screen & assigns it to the character 'ch'.
istream &getline(char *buffer,int size,char del='\n')-It accepts a string from input stream until it enters a newline character.

ostream class functions:
ostream &put(char ch)- It inserts a character ch in an output screen.
ostream &write(const char *s, streamsize n)- It inserts first n characters of the character array pointed to by 's' into the output screen.

Examples: Program to illustrate use of get() & put() functions.

```
#include<iostream.h>
#include<conio.h>int
main()
{
        char c;
        clrscr();
        cin.get(c); //get a character from keyboard & assigns it to
        cout<<"Entered Character is:"<<c; //display an entered character on output screen.
        return 0;
}
```

In above program we can also display same character entered by user using **cout.put(c) method** instead of using cout statement.

Examples: Program to illustrate use of getline() & write() functions.

```
#include<iostream.h>
#include<conio.h>int
main()
{
        intsize;
        char name[20];
        char*city="Pune";
        clrscr();
        cout<<"Enter name of student:";//Accept name of student from user
        cin>>name;
        cout<<"Entered name is:"<<name;//Display entered name
        cout<<"City of student is:"<<endl;
        cout.write(city,10);//Display city using write function
        cout<<"Enter another name of student:"<<endl;
        cin.getline(name,size); //Use geline function to accept name
        cout<<"Another name of student is:"<<name; //Display another name of student accepted
        from user
        return 0;
}
```

Formatted Console I/O Operations:
C++ supports a number of features that could be used for formatting output.These features includes:

- ios class functions &flags
- Manipulators
- User-Defined Outputfunctions(Manipulators)

The ios class contains a large number of member functions that help us to format the output in a number of ways.

Manipulators are helping functions that can modify the input or output stream. These format manipulators are available by including the file "<iomanip.h>".

ios format functions & manipulators:

ios Functions	Task	Equivalent Manipulators
width()	Specify required field size for displaying an output value	setw()
precision()	Specify number of digits to be displayed after the decimal point of float value	setprecision()
fill()	Specify a character that is used to fill the unused portion of a field	setfill()
setf()	Specify format flags that can control the form of output display i. e left justification & right justification	setiosflags()
unsetf()	To clear the flags specified	resetiosflags()

```
#include<iostream.h>#include<c
onio.h> #include<math.h>
int main()
{
        clrscr();
        cout.width(5);                  //set width to 5
        cout<<123<<12<<endl;            //dispay output in width of box5
        cout.width(5);
        cout<<543;
        cout.width(5);
        cout<<19<<endl;
        cout.precision(3);              //display 3 digits after decimalpoint
        cout<<sqrt(2)<<"\n";
        cout.precision(4);              //display4digitsafterdecimalpoint
        cout<<sqrt(3)<<"\n";
        cout.fill('*');                 //Padding fill with'*'
        cout.width(10);
        cout<<"SYBBA"<<"\n";
        cout.fill('#');                 //Padding fill with'#'
        cout.setf(ios::right,ios::adjustfield);   //it display output to rightside
        cout.width(12);
        cout<<"CA"<<"\n";
        return 0;
}
Output:
  12312
```

```
 54319
 1.414
 1.7321
 *****SYBBA
 ##########CA
```

User-Defined Manipulators:
In addition to predefined functions C++ allows us to create our own manipulator functions to provide any special output formats.

```
ostream & manipulator(ostream & output)
{
        -------
        // code
        -------
        return output;
}
```
manipulator is the name of manipulator under creation.

Example: Program to illustrate how to create user defined manipulator.

```
#include<iostream.h>
ostream & unit(ostream & output)
{
        output<<"Kilograms";
        return output;
}
int main()
{
        cout<<"Weight:"<<40<<unit;
        return 0;
}
```
Output:

```
C:\TURBOC3\BIN>TC
Weight:40Kilograms
```

Practice programs:

1. Define a class Item that contains data member as Item_no, Item _Name, Item _Price. Derive a class Discount(discount_in_percentage) from class Item. A Customer buys 'n' items.Acceptquantityforeachitem,calculatetotaldiscountandaccordinglygenerateand display the bill using appropriateManipulators.

Set A:

1. Write a C++ program to create a class Employee which contains data members as Emp_Id, Emp_Name, Basic_Salary, HRA, DA, Gross_Salary. Write member functions to acceptEmployeeinformation.CalculateanddisplayGrosssalaryofanemployee.

(DA=25%ofBasicsalaryandHRA=40%ofBasic salary)(Useappropriatemanipulators todisplayemployeeinformationingiven format:-Emp_IdandEmp_Nameshouldbeleft justified and Basic_Salary, HRA, DA, Gross salary Right justified with a precision of threedigits)

2. Write a C++ program to create a class Teacher which contains data members as Teacher_Name, Teacher_City, Teacher_Contact_Number. Write member functions to accept and display five teachers information. Design User defined Manipulator to print Teacher_Contact_Number.(ForContactNumbersetrightjustification,maximumwidthto 10 and fill remaining spaces with'*')

Set B:

1. Create a C++ class Train with data members as Train_No,Train_Name,No_of Seats,Source_Station,Destination_Station. Write necessary member functions for the following:
   i. Accept details of ntrains.
   ii. Display all traindetails.
   iii. Display details of train from specified starting station and ending station byuser.
2. Create a C++ class Manager with data members Manager_Id, Manager_Name, Mobile_No., Salary. Write necessary member functions for thefollowing:
   i. Accept details of n managers
   ii. Display manager details in ascending order of theirsalary.
   iii. Display details of a particular manager. (Use Array of object and Use appropriate manipulators.)

Set C:

1. Create a C++ class Marksheet with data members Seat_No., Student_Name, Class, Subject_Name, Int_Marks, Ext_Marks, Total, Grand_Total, Percentage, Grade. Write member function to accept Student information for 4 subjects. Calculate Total, Grand_Total, Percentage, Grade and display Marksheet. (Use user defined manipulator)

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No. 8: Working with Files

File: It is collection of data or information. **Stream**: It is sequence of bytes.

To perform input and output operations on files, three classes included in the <fstream.h> library. It defines several classes including ifstream , ofstream and fstream.

Stream	**Description**
ofstream	Stream class to write on files
ifstream	Stream class to read from files
fstream	Stream class to both read and write from/to files.

Opening file:
File can be opened by using member function open() or by using constructor.

Stream	**Description**	**Examples By Using Constructor**	**Examples By Using Member Function**
ofstream	Stream class to write onfiles	ofstream outfile1("first.txt");	ofstream outfile2; outfile2.open("second.txt");
ifstream	Stream class to read fromfiles	ifstream infile1("first.txt");	ifstream infile2; infile2.open("second.txt");
fstream	Stream class toboth read and write to / fromfiles.	fstream file1("first.txt",ios::out);	fstream file2; file2.open("second.txt",ios::out);

Detecting End-Of-File:
It checks whether end of file occurs or not. eof() is member function of ios class. It returns nonzero value if end of file condition is encountered and zero otherwise.

Syntax:

```
ifstream fin;
if(fin.eof()!=0)
{
        exit(1);
}
```

This statement terminates the program on reaching end of file.

File Opening Modes:
There are different modes (flags) of a file which are listed below:

Parameter	**Meaning**
ios::in	Open for input operations.
ios::out	Open for output operations.

ios::binary	Open in binary mode.
ios::ate	Set the initial position at the end of the file. If this flag is not set, the initial position is the beginning of the file.
ios::app	All output operations are performed at the end of the file, appending the content to the current content of the file.
ios::trunk	If the file is opened for output operations and it already exists, its previous content is deleted and replaced by the new one.

Closing File:
A file which is opened while reading or writing in **file handling** must be closed after performing an action on it.

Syntax:
filename.close();

File Pointer and Their Manipulations:
Each file has two pointers associated with it known as file pointers.
Input pointer (get pointer)
Output pointer (put pointer)

Following member functions are used to move the file pointer at the desired position while reading or writing from the file.

Function	**Description**
seekg()	Moves get pointer(input) to specified location.
seekp()	Moves put pointer(output) to a specified location.
tellg()	Gives the current position of the get pointer.
tellp()	Gives the current position of the put pointer.

File Handling Functions:
C++ provides us with the following operations in File Handling:

open() Function-To create a file by using open function.
file.open("sample.txt",ios::in |ios::out);

get() & put() Functions- put() writes a single character in file and get() reads a single character from a file.
fstream file; //Input & output stream
file.put('h'); //put char to file
file.get(ch); //get character from file

read() & write() Functions- These functions are used to perform read & write operations on binary file.
infile.read((char *) & v, sizeof(v));
infile.write((char *) & v, sizeof(v));

These functions take two arguments. The first is address of the variable V and second is the length of that variable in bytes.

Text and Binary files:
The C++ language supports two types of files:

- Textfiles
- Binary files

Text Files:
These files are designed to store text. In such files various character translations are performed such as "\r+\f" is converted into "\n", whereas in binary files no such translations are performed. By default, C++ opens the files in text mode.

Example: Program to illustrate reading & writing to text file .

```
#include<fstream.h>i
nt main ()
{
  char sname[20]="SYBBA",line[20];
  ofstream outfile;
  outfile.open("example.txt");
  outfile<<sname;
  outfile.close();
  ifstream infile;
  infile.open("example.txt");
  infile.getline(line,20);
  cout<<line;
  infile.close();
  return 0;
}
```

OUTPUT:
SYBBA
By using above program we can write 'SYBBA' to text file 'example.txt'. Then we perform read operation on same file & display 'SYBBA' to output screen

Binary Files:
It is used to read & write a given number of bytes on the given stream. write() is a member function of ostream inherited by ofstream and read is a member function of istream inherited by ifstream. Objects of class fstream have both.

Example: Program to illustrate reading & writing to Binary File.

```
// writing on a text file
#include<fstream.h>
int main ()
{
  char sname[20]="SYBBA";
  ofstream outfile;
```

```
 outfile.open("example.bin");
 outfile.write((char *) & sname,sizeof(sname));
 outfile.close();
 ifstream infile;
 infile.open("example.txt");
 infile.read((char *)& sname,sizeof(sname));
 cout<<sname;
 infile.close();
 return 0;
}
```

OUTPUT

SYBBA

By using above program we can write 'SYBBA' to text file 'example.bin. Then we perform read operation on same file & display 'SYBBA' to output screen

Reading & Writing Class Objects:

How class objects can be written to & read from disk files.

Example: Program to illustrate reading & writing class objects.

```
#include <iostream.h>
#include <fstream.h>
class student
{
        intrno;
        charname[20];
        public:
        void getdata();
        void putdata();
};
void student:: getdata()
{
        cout<<"Enter rno:\n";
        cin>>rno;
        cout<<"Entername:\n";
        cin>>name;
}
void student:: putdata()
{
          cout<<"Roll No:"<<rno<<endl;
          cout<<"Name:"<<name<<endl;
}
int main ()
 {
   student s[3];
   fstream file;
   file.open("student.txt",ios::in | ios::out);
```

```
   cout<< "Enter details of 3 students:\n";
   for(int i=0;i<3;i++)
   {
         s[i].getdata();
         file.write((char *) & s[i],sizeof(s[i]));
   }
   file.seekg(0); //reset to start
   cout<<"\nOUTPUT\n\n";
   for(i=0;i<3;i++)
   {
         file.read((char *) & s[i],sizeof(s[i]));
         s[i].putdata();
   }
   file.close();
   return 0;
}
```

OUTPUT:
Enter details of 3 students:
Enterrno:1
Entername:Avani
Enterrno:2
Entername:Ananya
Enterrno:3
Enter name:Kavya

OUTPUT
Roll No:1
Name:Avani
Roll No:2
Name:Ananya
Roll No:3
Name:Kavya

Updating A File:Random Access

Updating is the maintenance of any data file. The updating includes one or more of following tasks:

- Displaying contents of afile
- Modifying an existingitem
- Adding a newitem
- Deleting an existingitem

These action requires the file pointers to move to a particular location. File contains collection of items of equal lengths. Size of each item/object can be obtained using

int object_length=sizeof(object);

Location of object can be obtained using int location=m* object_length;
This location gives us byte number of the first byte of mth object.We can set file pointer to reach this byte with the help of seekg() & seekp() .

We also find total number of objects ina file using object_length as follows:
int n=file_size/ object_length;
The file_size can be obtained using function tellg() & tellp() when file pointer is located at the end of file.

Error Handling During File Operations:
There are several error handling functions supported by class ios that help you read and process the status recorded in a file stream. Following table lists these error handling functions and their meaning :

Function	Meaning
int eof()	Returns non-zero (true value) if end-of-file is encountered while reading; otherwise returns zero (false value).
int fail()	Returns non-zero (true) when an input or output operation has failed.
int bad()	Returns a non-zero value if an invalid operation is attempted or any unrecoverable error has occurred. However, if it is zero (false value), it may be possible to recover from any other error reported and continue operations.
int good()	Returns non-zero (true) if no error has occurred. This means, all the above functions are false. For example, if fin.good() is true, everything is okay with the stream named as fin and we can proceed to perform I/O operations. When it returns zero, no further operations can be carried out.

Command Line Arguments:
We supply arguments to main function at the time of invoking program by command line argument. They may be used to pass the names of data files.
Example: Program to illustrate use of command line arguments.

```
#include<iostream.h>#include<fstrea
m.h> #include<stdlib.h>
int main(int argc,char *argv[])
{
int number[9]={1,2,3,4,5,6,7,8,9};
if(argc!=2)
{
        cout<<"argc="<<argc<<"\n";
        cout<<"Error in arguments\n";
        exit(1);
}
ofstream fout1;
fout1.open(argv[1]);
if(fout1.fail())
```

```
{
        cout<<"Unable to open a file"<<argv[1]<<"\n";
        exit(1);
}
else
{
        for(int i=0;i<9;i++)
        {
        if(number[i]%2==0)
        fout1<<number[i]<<""; //write all even numbers from number array to file
        }
}
fout1.close();
ifstream fin;
int i;
char ch;
for(i=1;i<argc;i++)
{
        fin.open(argv[i]);
        cout<<"Contents of"<<argv[i]<<"\n";
        do
        {
        fin.get(ch);//reads an even numbers from file
        cout<<ch; //display it
}while(fin);
cout<<"\n\n";
fin.close();
}
return 0;
}
```

Output:

```
C:\TC\SOURCE>temp a.txt
Contents of a.txt
2468
C:\TC\SOURCE>exit
```

To run this program we first compile it. Then instead of using Ctrl+F9 we have to run it by dos shell. Click on File & then select DOS Shell.DOS Shell gets opened. Now give program name space a.txt. All even numbers will get added to a.txt file afterwards we print even numbers from 'a.txt' file to the output screen. Sometime program name is not found in BIN directory so change directory from BIN to SOURCE & then run your program.

Practice programs:

1. Write a C++ program to copy even numbers from the file "Numbers.txt" into the file "even.txt" and odd numbers into the file "odd.txt". Display the count of numbers in each file. Compute the median and average of numbers in bothfiles.
2. Write a C++ program that reads a "source.txt" file and creates another file named as "destination.txt" which is identical to source except that every sequence of consecutive blank spaces is replaced by a singlespace.
3. Write a C++ program to read the contents from the file "sample.txt". Store all the characters from "sample.txt" into the file "character.txt" & store all digits into the file "digit.txt".
4. Write a C++ program which will accept 'n' integers from user through command line argument.StorePrimenumbersinfile"Prime.txt"andremainingnumbersin"Others.txt".

Set A:

1. Write a C++ program to accept 'n' numbers from user through Command LineArgument. Store all positive and negative numbers in file "Positive.txt" and "Negative.txt" respectively.
2. Write a C++ program to read the contents of a text file. Count and display number of characters, words, lines and blank spaces from a file. Find the number of occurrences of a given word present in afile.
3. Create a C++ class Employee with data members Emp_No, Emp_Name, Emp_Marks. Write necessary member functions for thefollowing:
    i. Accept the details and store it into the file"Emp.dat"
    ii. Read the details from file and displayit.
    iii. Update a given record into thefile.

Set B:

1. Write a C++ program to create a class Newspaper with data members Name, publisher, cost. Write necessary member functions for thefollowing:
    i. Accept details for 'n' Newspapers from user and store it in a file "Newspaper.txt".
    ii. Display details of Newspapers from a file.
    iii. Count the number of objects stored in afile.
2. Write a C++ program that reads from a formatted file a list of 4 students and their marks for 3 tests, computes the average test score for each student and the grade and outputs them in anotherfile.
3. Create a C++ class 'city' with data members name and STD code. Accept 'n' cities with STD codes from user. Store this data in the file 'cities.txt'. Write a program that reads the data from file cities.txt display the list of city with STD codes from afile

Set C:

1. Create a C++ class MyFile containing:

fstream fp;
Char *fn;
Write necessary member Functions using operator overloading:

+	F3=F1+F2	Put contents of F1 and F2 inF3.
-	-F3	Changes the case of all upper and lower case characters inF3.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No. 9: Templates

Templates are powerful features of C++ which allows you to write generic programs. The simple idea is to pass data type as a parameter so that we don't need to write the same code for different data types. In simple terms, we can create a single function or a class to work with different data types using templates. Templates are often used in larger code base for the purpose of code reusability and flexibility of the programs. The concept of templates can be used in two different ways:

- FunctionTemplates
- ClassTemplates

Function Template:
It is used to define generic functions. A single function template can work with different data types at once. It works on different types of data.

Syntax of function template with single parameter:
A function template starts withthe keyword template followed by template parameter(s) inside <> which is followed by functiondeclaration.

```
template <class T>
returntype functionName(arguments of type T)
{
   // Body of function with type T
}
```

T is a generic name for a data type used by the function. This name can be used within the function definition.

Syntax of function template with Multiple Parameters:

```
template <class T1,class T2>
returntype functionName(arguments oftypesT1,T2,..... )
{
   // Body of function
}
```

Example: Program to illustrate use of Function Template with multiple parameters.

```
#include<iostream.h>template<c
lass T>
T add(T num1, T num2)
{
   return (num1 + num2);
}

int main()
{
   int result1;
```

```
    double result2;
    // calling with int parameters
    result1 = add(2, 3);
    cout << "2 + 3 = " << result1 << endl;

    // calling with double parameters
    result2 = add(2.2, 3.3);
    cout << "2.2 + 3.3 = " << result2 << endl;

    return 0;
}
```
Output:
2 + 3 = 5
2.2 + 3.3 = 5.5

Class Template:

Class templates are used for writing generic class operations. We would need to create a different class for each data type or create different member variables and functions within a single class using a class template.

Syntax of class template with single parameter:
```
template <class T>
class className
{
        public:
          ... .. ...
        //class member specification with anonymous type T
          ... .....

          ... .....

};
```
T is a generic name for a data type which will be specified when a class is instantiated. we can define more than one generic data type by using a comma-separated list.

Syntax of class template with multiple parameters:
```
template <class T1, class T2,…..>
class className
{
        public:
          ... .....
          ... .....
          ... .....

};
```

Example: Program to illustrate use of Class Template with multiple parameters.

```
#include<iostream.h>
#include<conio.h>
template<classT1,classT2>c
lassA
{
  T1 a;
  T2b;
   public:
  A(T1 x,T2y)
{
   a = x;
   b = y;
}
void display()
{
   cout<<"Values of a and b are :"<< a<<","<<b<<endl;
}
};
int main()
{
            clrscr();
            A<int,float> d(5,6.5);
            d.display();
            return 0;
}
```

Output:
Values of a and b are: 5, 6.5

Practice programs:

1. Write a C++ program to swap two integer values and two float values by using function template.

Set A:

1. Write a C++ template program to accept array elements of type integers & characters. Reverse an array of bothtypes.
2. Write a C++ program to find maximum & minimum of two integer numbers and twofloat numbers by using functiontemplate.
3. Write a C++ template program to sort the array elements of type integer, float and character.

Set B:

1. Write a C++ program to define class template for calculating the square of givennumbers with different datatypes.

2. Write C++ template program to find the area of circle & rectangle with different data types.
3. Write a template to represent a generic vector. Include member functions to perform the followingtasks:
    i. To create thevector.
    ii. To modify the value of a givenelement.
    iii. To multiply the vector by a scalarvalue.
    iv. To display the vector in the form (10,20, 30,…..)

Set C:

1. Write C++ template program to implement stack & its operations like push &pop.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Section-II

NODE JS

ASSIGNMENT NO. 1: NODE.JS WEB SERVER, MODULES & NPM

Introduction:

Node.js is an open-source server-side runtime environment that provides an event driven, non-blocking (asynchronous) I/O and cross-platform runtime environment for building highly scalable server-side application using JavaScript.

Node.js can be used to build different types of applications such as command line application, web application, real-time chat application, REST API server etc. However, it is mainly used to build network programs like web servers, similar to PHP, Java, or ASP.NET.

Downloads, Installation and setting up Environment for node.js

- The official Node.js website has installation instructions for Node.js: https://nodejs.org
- DownloadEditorvisualstudiocodefornodejsfrom: https://code.visualstudio.com/download
- OnceyouhavedownloadedandinstalledNode.js&VScodeeditoronyourcomputer, you can run "Hello World" node.js app and display Hello World! Message on a web browser.
- Create a Node.js file named "myfirst.js", and add the followingcode:

myfirst.js

```
var http = require('http');
http.createServer(function (req, res) {
   res.writeHead(200, {'Content-Type': 'text/html'});
res.end('Hello World!');
}).listen(8081);
console.log('Server running at http://127.0.0.1:8081/');
```

Now execute the myfirst.js to start the server as follows −

- **$ nodemyfirst.js**
- Verify the Output on browser. Server hasstarted.
- Server running athttp://127.0.0.1:8081/

A Node.js application consists of the following three important components −

Import required modules − We use the require directive to load Node.js modules.

Create server − A server which will listen to client's requests similar to Apache HTTP Server.

Read request and return response − The server created in an earlier step will read the HTTP request made by the client which can be a browser or a console and return the response.

Node.js has a built-in module called HTTP, which allows Node.js to transfer data over the Hyper Text Transfer Protocol (HTTP).

To include the HTTP module, use the require() method:

```
var http = require('http');
```

Node.js as a Web Server

The HTTP module can create an HTTP server that listens to server ports and gives a response back to the client.Use the createServer() method to create an HTTP server:

REPL Terminal:

- REPL stands for Read Eval Print Loop and it represents a computer environment like a Windows console or Unix/Linux shell where a command is entered and the system responds with an output in an interactivemode.
- Node.js or Node comes bundled with a REPL environment. It performs the following tasks-
- Read − Reads user's input, parses the input into JavaScript data-structure, and stores in memory.
- Eval − Takes and evaluates the datastructure.
- Print − Prints theresult.
- Loop − Loops the above command until the user presses ctrl-ctwice.
- The REPL feature of Node is very useful in experimenting with Node.js codes and to debug JavaScriptcodes.

  REPL can be started by simply running node on shell/console without any arguments as follows.

  $ node

  You will see the REPL Command prompt > where you can type any Node.js command −

  $ node > Simple Expression

  Let's try a simple mathematics at the Node.js REPL command prompt −

  $ node > 1 + 3

  4

  > 1 + (2 * 3) - 4

  3

You can make use variables to store values and print later like any conventional script.If var keyword is not used, then thevalue is stored in the variable and printed. Whereas ifvarkeywordisused,thenthevalueisstoredbutnotprinted.Youcanprintvariables using console.log().

```
$ node > x = 10 10
> var y = 10undefined
> x + y
  20
> console.log("HelloWorld")
Hello Worldundefined
```

Modules in Node.js

Node.js has a simple module loading system. A module in node.js is a simple or complex functionality organized in single or multiple javascript files which can be reusable again.

Node.js Module Types

In Node.js modules can be categorized in 3 types

1. CoreModules
2. LocalModules
3. Third PartyModules

Node.js Core Modules

Node.jshasseveralmodulescompiledintoitsbinarydistribution,andloadautomatically whentheNode.jsprocessstarts,thesearecalledthecoremodules.Thesecoremodulesof node.js are located within Node.js's source and are located inside**"lib"** folder.

Some of the core modules are listed below.

- ***http*** **-** This module is used to create httpserver.
- ***fs*** - This module is used to perform file operations like reading, writing, appending and deleting filesetc.
- ***Crypto*** - This module provides cryptographic functionalities like encryption, decryption, sign, verification, digestingetc.
- ***Querystring*** -This method includes methods to deal with querystring like unescapeBuffer, unescape, escape, encode, stringify, decode andparse.
- ***url*** - This module includes methods for url resolutions,resolve, parsing, formatetc.
- *path*-Thismoduleisusedtodealwithfilepathswhenworkingwithfilesystem.

Local Modules

Local modules are user defined modules which are mainly used for specific projects and locally available in separate files or folders within project folders. These types of modules contain application specific functionality.

Note

We can package locally created local modules and distribute them via NPM (Node Package Manager), which can be used by others and the node community.

Third party module

The third-party module can be downloaded by NPM (Node Package Manager). These types of modules are developed by others and we can use that in our project. Some of the best third party module examples are: express, gulp, lodash, async, socket.io, mongoose, underscore, pm2, bower, q, debug, react, mocha etc.

Third party modules can be install inside the project folder or globally.

How to load a module?

To load a module in your node application you can just use "require()" function. whose syntax is given below.

```
var module=require('module_name');
```

There are several ways to reference modules, this depends on what type of module we are going to load.

Loading core module

Core modules can be loaded as follows.

var http=require('http');

As I have already told you that code modules are loaded in "lib" directory, so in the above example http module will be loaded from lib folder.

How to create and load local module?

In Node.js files and modules are in a one-to-one correspondence. The following example will explain to you how to create a local Node.js module.

```
function Circle(radius) {
 return {
  area: function area() {
  return Math.PI * Math.PI * radius;
  }
       };
}
module.exports = Circle;
```

In the above example We have created a function which is used to find an area of the Circle. In this example in the last line I have written "module.exports=Circle" this is a very important line here. Here module is a variable that represents the modules in which we are currently in. We can export any type of object. Save above file using "Circle.js". To use this module inside another file, app.js, the code can be written as follows.

app.js	**Output**
var circle=require('./Circle.js'); var obj=circle(7); var output=obj.area(); console.log(output);	C:\Users\Dell-PC\Node-app> node app.js 69.0872308076255

In the above example We loaded a file whose name is Circle.js using the require function which exports Circle object.

Different ways for loading local node.jsmodule

Therearelotsofwaystoloadlocallycreatedmodules.

- Using absolutepath

  var module = require('/<folder_name1>/<folder_name2>/.../module');
- Using relativepath

  var module = require('./module');

Here We are not giving .js extension so there is no need to add ".js" Node finds .js files if we do not give .js as an extension, it means the following lines will be the same.

1. **var module =require('./module');**
2. **var module =require('./module.js');**

Using folder path

You can also use folder path to load modules as follows:

```
var module = require('./folder_name');
```

But a folder can contain lots of modules and javascript files so node finds index.js file and loads by default. Otherwise we can create package.json where we can define node module namewhich we want to load by default. We can write package.json like asfollows.

```
{
 "name" : "module_name",
 "main" : "./folder_path/module_name.js"
}
```

Loading third party modules

Third party Node.js module can be downloaded using NPM (node package manager) which you can download locally or globally. To download globally we use the following command.

npm install -g <module_name>

here we use -g to install package globally. If you want to install locally then use the following command.

npm install --save <module_name>

Above command will download node package inside node_modules folder and then you can directly use require function to load node module.

var module= require('module_name');

Caching Modules

In Node.js modules are cached when module is loaded the first time. It means if you load the same node module 2 times then node.js does not load that module again it will copy that module from cache. Example is shown below.

console.log("Module Loaded Successfully");

We have created a module that is My_Module and written the above code and saved it using My_Modules.js. Now I am creating app.js file where I am writing the following script.

var my_module1=require('./My_Module');

Above code will print the following output.

```
C:\Users\Dell-PC\Node-app> node app.js
   Module Loaded Successfully
```

NowWearemodifyingapp.jsandwritingthefollowingcode.

```
var my_module1=require('./My_Module');
var  my_module2=require('./My_Module');
```

In the above code We have created two objects for My_Module. But when you run it then you will get the following output.

C:\Users\Dell-PC\Node-app> node app.js
Module Loaded Successfully

This means that module initializes only once. This is very important to know if you are creating any module.

Practice Programs:

1) Create a Node.js Application that performs following operations on Bufferdata.
    1. Concat
    2. Compare
    3. Copy
2) Create node.js application that uses local module to find age ofperson after accepting date of birth.
3) Create node.js application that create navigation bar on your web page and on selecting option from navbar, goes to respective page such as contact us, about usetc.
4) Create node.js application that create unit conversion module such as mm to cm and using it performconversion.

SET A

1) Create a Node.js file that will convert the output "Hello World!" into upper-caseletters.
2) CreateaNode.jsApplicationthatusesuserdefinedModuletoreturntheSumofdigitsofaof given number.
3) Create a Node.js Application that uses user defined module circle.js which exports the functions area () and circumference () and display the details onconsole.

SET B

1) Create a Node.js Application that accepts first name, last name of a Person and define a Module that concatenate first name and lastname.
2) Create a Node.js Application that uses user defined Module to return the Factorial of a of given number.
3) CreateNodejsapplicationusinguserdefinedRectangle moduletofindarea ofrectangleand display the details onconsole.

SET C

1) Create Nodejs Module and Publish Over tonpm

Signature of the instructor: ___ Date:_

Assignment Evaluation:

0:NotDone 1:Incomplete 2:LateComplete 3: NeedsImprovement

4:Complete 5:Well-Done

ASSIGNMENT NO. 2: FILE SYSTEM

Node.js as a File Server

The Node.js file system module allows you to work with the file system on your computer.
To include the File System module, use the require() method:

```
var fs = require('fs');
```

Common use for the File System module:

- Readfiles
- Writefiles
- Createfiles
- Updatefiles
- Deletefiles
- Renamefiles

Major File I/O methods.

Read Files

The fs.readFile() method is used to read files on your computer.
Assume we have the following HTML & demo_readfile.js file (located in the same folder as Node.js):

demofile1.html

```
<html>
<body>
<h1>My Header</h1>
<p>My paragraph.</p>
</body>
</html>
```

demo_readfile.js

```
var http = require('http');
var fs = require('fs');
http.createServer(function (req, res) {
  fs.readFile('demofile1.html', function(err, data) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write(data);
    return res.end();
  });
}).listen(8081);
```

Initiate demo_readfile.js:
C:\Users*Your Name*>node demo_readfile.js
you will see the result on browser with url: http://localhost:8081

Synchronous vs Asynchronous

Every method in the fs module has synchronous as well as asynchronous forms. Asynchronous methodstakethelastparameterasthecompletionfunctioncallbackandthefirstparameterofthe callback function as error. It is better to use an asynchronous method instead of a synchronous method,astheformerneverblocksaprogramduringitsexecution,whereasthesecondonedoes.

Example

Create a text file named **input1.txt** with the following content −
We are students of SY BBA (CA)
learning node.js in simple and easy way!!!!!

Let us create a js file named **main.js** with the following code &**run the main.js** to see the result:

var fs = require("fs"); // Asynchronous read fs.readFile('input1.txt', function (err, data) { if (err) { return console.error(err); } console.log("Asynchronous read: " + data.toString()); }); // Synchronous read var data = fs.readFileSync('input1.txt'); console.log("Synchronous read: " + data.toString()); console.log("Program Ended");	**$ node main.js** **Output:** Synchronous read: We are students of SY BBA (CA) learning node.js in simple and easy way!!!!! Program Ended Asynchronous read: We are students of SY BBA (CA) learning node.js in simple and easy way!!!!!

Open a File

Syntax

Following is the syntax of the method to open a file in asynchronous mode −

fs.open(path, flags[, mode], callback)

Parameters

Here is the description of the parameters used −

- **path** − This is the string having file name includingpath.
- **flags** − Flags indicate the behavior of the file to be opened. All possible values have been mentionedbelow.
- **mode**− Itsetsthefilemode(permissionandstickybits),butonlyifthe filewascreated. It defaults to 0666, readable andwriteable.
- **callback**−Thisisthecallbackfunctionwhichgetstwoarguments(err,fd).

Flags

Flags for read/write operations are −

Sr.No.	Flag & Description
1	r: Open file for reading. An exception occurs if the file does notexist.
2	r+: Open file for reading and writing. An exception occurs if the file does not exist.
3	rs: Open file for reading in synchronous mode.
4	Open file for reading and writing, asking the OS to open it synchronously. See notes for 'rs' about using this with caution.
5	Open file for writing. The file is created (if it does not exist) or truncated (if it exists).
6	Wx: Like 'w' but fails if the path exists.
7	Open file for reading and writing. The file is created (if it does not exist) or truncated (if it exists).
8	wx+: Like 'w+' but fails if path exists.
9	a: Open file for appending. The file is created if it does not exist.
10	ax: Like 'a' but fails if the path exists.

11	a+: Open file for reading and appending. The file is created if it does not exist.
12	ax+: Like 'a+' but fails if the the path exists.

Example

Let us create a js file named main.js having the following code to open a file input.txt for reading and writing.

```
var fs = require("fs");
// Asynchronous - Opening File
console.log("Going to open file!");
fs.open('input.txt', 'r+', function(err, fd) {
  if (err) {
    return console.error(err);
  }
  console.log("File opened successfully!");
});
```

Now run the main.js to see the result −

$ node main.js

Output:

```
Going to open file!
File opened successfully!
```

Syntax

Following is the syntax of the method to get the information about a file −

fs.stat(path, callback)

Parameters:

Here is the description of the parameters used −

- **path** − This is the string having file name includingpath.
- **callback** − This is the callback function which gets two arguments (err, stats) where stats is an object of fs.Stats type which is printed below in theexample.

Apart from the important attributes which are printed below in the example, there are several useful methods available in fs.Stats class which can be used to check file type. These methods are given in the following table.

Sr.No.	Method	Description
1	stats.isFile():	Returns true if file type of a simple file.
	stats.isDirectory():	Returns true if file type of a directory.
3	stats.isBlockDevice():	Returns true if file type of a block device.
4	stats.isCharacterDevice():	Returns true if file type of a character device.
5	stats.isSymbolicLink():	Returns true if file type of a symbolic link.
6	stats.isFIFO():	Returns true if file type of a FIFO.
7	stats.isSocket():	Returns true if file type of asocket.

Example

Let us create a js file named main.js with the following code –

main.js	Now run the main.js to see the result − $ node main.js **Output**:
var fs = require("fs"); console.log("Going to get file info!"); fs.stat('input.txt', function (err, stats) { if (err) { return console.error(err); } console.log(stats); console.log("Got file info successfully!"); // Check file type console.log("isFile ? " + stats.isFile()); console.log("isDirectory ? " + stats.isDirectory()); });	Going to get file info! Stats { dev: 3666283250, mode: 33206, nlink: 1, uid:0, gid:0, rdev: 0, blksize: 4096, ino: 281474976790862, size: 83, blocks: 0, atimeMs: 1611035707736.8445, mtimeMs: 1611034982031.1924, ctimeMs: 1611034982031.1924, birthtimeMs: 1609511475673.6436, atime: 2021-01-19T05:55:07.737Z, mtime: 2021-01-19T05:43:02.031Z, ctime: 2021-01-19T05:43:02.031Z, birthtime: 2021-01-01T14:31:15.674Z } Got file info successfully! isFile ? true isDirectory ? false

Writing a File

Syntax : Following is the syntax of one of the methods to write into a file −

fs.writeFile(filename, data[, options], callback)

This method will over-write the file if the file already exists. If you want to write into an existing file then you should use another method available.

Parameters : Here is the description of the parameters used −

- **path** − This is the string having the file name includingpath.
- **data** − This is the String or Buffer to be written into thefile.
- **options** − The third parameter is an object which will hold {encoding, mode,flag}.By default. encoding is utf8, mode is octal value 0666. and flag is'w'
- **callback**−Thisisthecallbackfunction which getsasingleparametererrthatreturnsan errorin case of any writingerror.

Example : Let us create a js file named main.js having the following code –

var fs = require("fs"); console.log("Going to write into existing file"); fs.writeFile('input2.txt', 'Simply Easy Learning!', function(err) { if (err) { return console.error(err); } console.log("Data written successfully!"); console.log("Let's read newly written data"); fs.readFile('input2.txt', function (err, data) { if (err) { return console.error(err); } console.log("Asynchronous read: " + data.toString()); }); });	Now run the main.js to see the result − **$ node main.js** **Output:** Going to write into existing file Data written successfully! Let's read newly written data Asynchronous read: Simply Easy Learning!

Syntax

Following is the syntax of one of the methods to read from a file −

fs.read(fd, buffer, offset, length, position, callback)

This method will use file descriptor to read the file. If you want to read the file directly using the file name, then you should use another method available.

Parameters

Here is the description of the parameters used −

- **fd** − This is the file descriptor returned byfs.open().
- **buffer** − This is the buffer that the data will be writtento.
- **offset** − This is the offset in the buffer to start writingat.
- **length** − This is an integer specifying the number of bytes toread.
- **position** − This is an integer specifying where to begin reading from in the file. If position is null, data will be read from the current fileposition.
- **callback**−Thisisthecallbackfunctionwhichgetsthethreearguments,(err,bytesRead,buffer).

Example

Let us create a js file named main.js with the following code −

main.js	Now run the main.js to see the result − $ node main.js **Output:**
var fs = require("fs"); var buf = Buffer.alloc(1024); console.log("Going to open an existing file"); fs.open('input.txt', 'r+', function(err, fd) { if (err) { return console.error(err); } console.log("File opened successfully!"); console.log("Going to read the file"); fs.read(fd, buf, 0, buf.length, 0, function(err, bytes){ if (err){ console.log(err); } console.log(bytes + " bytes read"); // Print only read bytes to avoid junk. if(bytes > 0){ console.log(buf.slice(0, bytes).toString()); } }); });	Going to open an existing file File opened successfully! Going to read the file 83 bytes read Going to open an existing file File opened successfully!

Closing a File

Syntax
Following is the syntax to close an opened file –

fs.close(fd, callback)

Parameters

Here is the description of the parameters used −

- **fd** − This is the file descriptor returned by file fs.open()method.
- **callback**−ThisisthecallbackfunctionNoargumentsotherthanapossibleexceptionaregiven to the completioncallback.

Example

Let us create a js file named main.js having the following code –

main.js

```
var fs = require("fs");
var buf = new Buffer.alloc(1024);

console.log("Going to open an existing file");
fs.open('input2.txt', 'r+', function(err, fd) {
  if (err) {
    return console.error(err);
  }
  console.log("File opened successfully!");
  console.log("Going to read the file");

  fs.read(fd, buf, 0, buf.length, 0, function(err, bytes) {
    if (err) {
      console.log(err);
    }

    // Print only read bytes to avoid junk.
    if(bytes > 0) {
      console.log(buf.slice(0, bytes).toString());
    }

    // Close the opened file.
    fs.close(fd, function(err) {
      if (err) {
        console.log(err);
      }
      console.log("File closed successfully.");
    });
  });
});
```

Now run the main.js to see the result –

$ node main.js

Output:

```
Going to open an existing file
File opened successfully!
Going to read the file
Simply Easy Learning!
File closed successfully.
```

Update Files

The File System module has methods for updating files:

- fs.appendFile()
- fs.writeFile()

The fs.appendFile() method appends the specified content at the end of the specified file:

Example

Append "This is my text." to the end of the file "input2.txt":

main.js	Now run the main.js to see the result − **$ node main.js**
var fs = require('fs'); fs.appendFile('input2.txt', ' This is my text.', function (err) { if (err) throw err; console.log('Updated!'); });	**Output:** Updated!

Truncate a File

Syntax:

Following is the syntax of the method to truncate an opened file **–**

fs.ftruncate(fd, len, callback)

Parameters:

Here is the description of the parameters used −

- **fd** − This is the file descriptor returned byfs.open().
- **len**−Thisisthelengthofthefileafterwhichthefilewillbetruncated.
- **callback** − This is the callback function No arguments other than a possible exception are given to the completioncallback.

Example

Let us create a js file named main.js having the following code –

main.js	
var fs = require("fs"); var buf = Buffer.alloc(1024); console.log("Going to open an existing file"); fs.open('input.txt', 'r+', function(err, fd) { if (err) { return console.error(err); } console.log("File opened successfully!"); console.log("Going to truncate the file after 10 bytes"); // Truncate the opened file. fs.ftruncate(fd, 10, function(err) { if (err) { console.log(err); } console.log("File truncated successfully.");	Now run the main.js to see the result − **$ node main.js** **Output:** Going to open an existing file File opened successfully! Going to truncate the file after 10 bytes File truncatedsuccessfully. Going to read the samefile SimplyEas File closed successfully.

```
    console.log("Going to read the same file");

    fs.read(fd, buf, 0, buf.length, 0, function(err, bytes){
      if (err) {
        console.log(err);
      }

      // Print only read bytes to avoid junk.
      if(bytes > 0) {
        console.log(buf.slice(0, bytes).toString());
      }

      // Close the opened file.
      fs.close(fd, function(err) {
        if (err) {
          console.log(err);
        }
        console.log("File closed successfully.");
      });
    });
  });
});
```

Delete a File

Syntax: Following is the syntax of the method to delete a file –

fs.unlink(path, callback)

Parameters

Hereisthedescriptionoftheparametersused−

- path−Thisisthefilenameincludingpath.
- callback−ThisisthecallbackfunctionNoargumentsotherthanapossibleexceptionaregiven to the completioncallback.

Example : Let us create a js file named main.js having the following code –

Main.js

```
var fs = require("fs");
console.log("Going to delete an existing file");
fs.unlink('input.txt', function(err) {
  if (err) {
    return console.error(err);
  }
  console.log("File deleted successfully!");
});
```

Now run the main.js to see the result −

```
$ node main.js
```

Output:

```
Going to delete an existing file
File deleted successfully!
```

Create a Directory

Syntax:

Following is the syntax of the method to create a directory –

fs.mkdir(path[, mode], callback)

Parameters:

Here is the description of the parameters used −

- path − This is the directory name includingpath.
- mode − This is the directory permission to be set. Defaults to0777.
- callback − This is the callback function No arguments other than a possible exception are given to the completioncallback.

Example

Let us create a js file named main.js having the following code –

main.js	
var fs = require("fs"); console.log("Going to create directory /tmp/test"); fs.mkdir('/tmp/test',function(err) { if (err) { return console.error(err); } console.log("Directory created successfully!"); });	Now run the main.js to see the result − $ node main.js **Output:** Going to create directory /tmp/test Directory created successfully!

Read a Directory

Syntax: Following is the syntax of the method to read a directory –

fs.readdir(path, callback)

Parameters:

Here is the description of the parameters used −

path − This is the directory name including path.

callback − This is the callback function which gets two arguments (err, files) where files is an array of the names of the files in the directory excluding '.' and '..'.

Example

Let us create a js file named main.js having the following code –

main.js	
var fs = require("fs"); console.log("Going to read directory /tmp"); fs.readdir("/tmp/",function(err, files) { if (err) { return console.error(err); } files.forEach(function (file) { console.log(file); }); });	Now run the main.js to see the result − $ node main.js **Output:** Going to read directory /tmp ccmzx99o.out ccyCSbkF.out employee.ser hsperfdata_apache test test.txt

Remove a Directory

Syntax: Following is the syntax of the method to remove a directory –

fs.rmdir(path, callback)

Parameters

Here is the description of the parameters used −

- path − This is the directory name includingpath.
- callback−ThisisthecallbackfunctionNoargumentsotherthanapossibleexceptionaregiven to the completioncallback.

Example

Let us create a js file named main.js having the following code –

main.js	Now run the main.js to see the result −
var fs = require("fs"); console.log("Going to delete directory /tmp/test"); fs.rmdir("/tmp/test",function(err) { if (err) { return console.error(err); } console.log("Going to read directory /tmp"); fs.readdir("/tmp/",function(err, files) { if (err) { return console.error(err); } files.forEach(function (file) { console.log(file); }); }); });	$ node main.js **Output:** Going to read directory /tmp ccmzx99o.out ccyCSbkF.out employee.ser hsperfdata_apache test.txt

Rename Files

To rename a file with the File System module, use the fs.rename() method.

The fs.rename() method renames the specified file:

Example

Rename "input2.txt" to "input3.txt":

main.js	Now run the main.js to see the result −
var fs = require('fs'); fs.rename('input2.txt', 'input3.txt',function (err){ if (err) throw err; console.log('File Renamed!'); });	**$ node main.js** **Output:** File Renamed!

File Upload in node.js with express module:

- Installexpressmoduleusing**npminstallexpress &npminstallexpress-fileupload**onterminal of vscode.
- Create upload folder in your applicationfolder.
- Create index.html and app.js file in your application folder with followingcode:

Index.html

```
<h1>Hey there, Upload file here</h1>
<form method="post" enctype="multipart/form-data" action="/">
<input type="file" name="filename">
<input type="submit" value="Upload">
</form>
```

app.js

```
var express = require('express'),
   app = express(),
   http =
require("http").Server(app).listen(8081),
   upload = require("express-fileupload");
app.use(upload())

console.log("Server Started")
app.get("/",function(req,res){
  res.sendFile(__dirname+"/index.html");
})
app.post("/",function(req,res){
   if(req.files){
      var file = req.files.filename,
         filename = file.name;
      file.mv("./upload/"+filename, function
(err){
         if(err){
            console.log(err)
            res.send("error occured")
         }
         else{
            res.send("Done")
         }
      })
   }
})
```

Now run the app.js to see the result −

$ node app.js

It will upload selected file in upload folder.

Practice Programs:

1. Create a Node.js Application to count occurrence of given word in a file and display the counts onconsole.
2. Write Node.js application that transfer a file as an attachment in student admission form onweb.
3. Createnode.jsapplicationthatuploadimagefileanddisplayimageicon onbrowserasof your organizationlogo.
4. Createnode.jsapplicationtoacceptfeedbackenteredthroughafeedbackformintoafile.

SET A

1. Create a Node.js file that opens the requested file and returns the content of file on terminal.
2. UsingNode.jscreateawebpagetoreadtwofilenamesfromuserandappendcontentsof first file into secondfile.
3. Using Node.js create a web page to read two created files names from user and combine contents of both in to third one with all contents in Uppercase.

SET B

1. CreateaNode.jsfilethatwritesanHTMLform,thatwilluploadafileinparticularfolder
2. CreateaNode.jsApplicationtodownloadjpgimagefromtheServer.
3. Create a Node.js Application to check whether given name is directory or file, if it file, truncate the content after 7bytes.

SET C

1. Create a Node.js Application to count number of lines in a given file.

Signature of the instructor: _ _ _ Date: _

Assignment Evaluation:

0:NotDone 1:Incomplete 2: Late Complete 3: Needs Improvement

4:Complete 5:Well-Done

ASSIGNMENT NO. 3: EVENTS IN NODE.JS

NodeJS followsEventDrivenSingleThreadApproach.ManyobjectsinaNodeemitevents, for example, a net.Server emits an event each time a peer connects to it, an fs.readStream emits an event when the file is opened. All objects which emit events are the instances of events.EventEmitter.

EventEmitter Class

EventEmitter class lies in the events module. It is accessible via the following code −

```
// Import events module
var events = require('events');
// Create an eventEmitter object
var eventEmitter = new events.EventEmitter();
```

When an EventEmitter instance faces any error, it emits an 'error' event. When a new listener is added, 'newListener' event is fired and when a listener is removed, 'removeListener' event is fired. EventEmitter provides multiple properties like **on** and **emit**. **on** property is used to bind a function with the event and **emit** is used to fire an event.

Methods : The following table lists all the important methods of EventEmitter class.

EventEmitter Methods	Description
emitter.addListener(event, listener)	Adds a listener to the end of the listeners array for the specified event. No checks are made to see if the listener has already been added.
emitter.on(event, listener)	Adds a listener to the end of the listeners array for the specified event. No checks are made to see if the listener has already been added. It can also be called as an alias of emitter.addListener()
emitter.once(event, listener)	Adds a one time listener for the event. This listener is invoked only the next time the event is fired, after which it is removed.
emitter.removeListener(event,listener)	Removes a listener from the listener array for the specified event. Caution: changes array indices in the listener array behind the listener.
emitter.removeAllListeners([event])	Removes all listeners, or those of the specified event.
emitter.setMaxListeners(n)	By default EventEmitters will print a warning if more than 10 listeners are added for a particular event.
emitter.getMaxListeners()	Returns the current maximum listener value for the emitter which is either set by emitter.setMaxListeners(n) or defaults to EventEmitter.defaultMaxListeners.
emitter.listeners(event)	Returns a copy of the array of listeners for the specified event.
emitter.emit(event[, arg1][, arg2][, ...])	Raise the specified events with the supplied arguments.
emitter.listenerCount(type)	Returns the number of listeners listening to the type of event.

Events

Sr.No.	Events &Description
1	**newListener** • **event** − String: the eventname • **listener** − Function: the event handlerfunction This event is emitted any time a listener is added. When this event is triggered, the listener may not yet have been added to the array of listeners for the event.
2	**removeListener** • **event** − String The eventname • **listener** − Function The event handlerfunction This event is emitted any time someone removes a listener. When this event is triggered, the listener may not yet have been removed from the array of listeners for the event.

Example

Create a js file named main.js with the following Node.js code −

```
var events = require('events');
var eventEmitter = new events.EventEmitter();
// listener #1
var listner1 = function listner1() {
   console.log('listner1 executed.');
}
// listener #2
var listner2 = function listner2() {
   console.log('listner2 executed.');
}
// Bind the connection event with the listner1 function
eventEmitter.addListener('connection', listner1);
// Bind the connection event with the listner2 function
eventEmitter.on('connection', listner2);
var eventListeners = require('events').EventEmitter.listenerCount
   (eventEmitter,'connection');
console.log(eventListeners + " Listner(s) listening to connection event");
// Fire the connection event
eventEmitter.emit('connection');
// Remove the binding of listner1 function
eventEmitter.removeListener('connection', listner1);
console.log("Listner1 will not listen now.");
// Fire the connection event
eventEmitter.emit('connection');
eventListeners = require('events').EventEmitter.listenerCount(eventEmitter,'connection');
```

```
console.log(eventListeners + " Listner(s) listening to connection event");
console.log("Program Ended.");
```

Now run the main.js to see the result −

```
$ node main.js
```

Output:

```
2Listner(s)listeningtoconnectionevent
listner1executed.
listner2executed.
Listner1 will not listennow.
listner2executed.
1 Listner(s) listening to connection event
Program Ended.
```

EventEmitter “emit()” function

EventEmitter class has a “emit()” function, which is used to create an Event. It takes one parameter.

eventsEmitter.emit(NameOfEventToCreate);

Here, NameOfEventToCreate: we need to pass Event Name to emit() function call as String to create that Event.

Example:-

```
var events = require("events");
var eventsEmitter = new events.EventEmitter();
eventsEmitter.emit("mobileon");
```

EventEmitter “on()” function

EventEmitter class has a “on()” function, which is used to bind an Event with an Event Handler JavaScript Function. It takes two parameters.

eventsEmitter.on(NameOfEventToBind, EventHandlerFuction);

Here, NameOfEventToBind: We need to pass Event Name a to on() function call as String to bind that event to given Event Handler JavaScript Function.

and EventHandlerFuction: Given Event Handler JavaScript Function to handle that event. It may be an anonymous JavaScript function or Plain JavaScript function.

Example:

This example is using anonymous JavaScript function as Event Handler.

```
var events = require("events");
var eventsEmitter = new events.EventEmitter();
eventsEmitter.emit("mobileon",function(data){
        console.log(data);
});
eventsEmitter.emit("mobileon");
```

We can also use Plain JavaScript function as Event Handler as shown below:

```
var events = require("events");
var eventsEmitter = new events.EventEmitter();
eventsEmitter.emit("mobileon",mobileOnHadler);
eventsEmitter.emit("mobileon");

function mobileOnHadler(data){
   console.log(data);
}
```

With this knowledge about EventEmitter class, we will develop a real-time simple example to see how Node JS handles events.

Example:

event.js	**Output:**
var EventEmitter = require('events').EventEmitter; var myEmitter = new EventEmitter; var customer = function(name){ *// do something* console.log('Customer Name: ' + name); }; myEmitter.on('customer', customer); myEmitter.on('message', function(msg){ *// do something* console.log('message: ' + msg); }); *// Execute the Application* myEmitter.emit('customer', 'Ninad'); myEmitter.emit('customer', 'Viru'); myEmitter.emit('message', 'this is the first message'); myEmitter.emit('message', 'this is the second message'); myEmitter.emit('message', 'welcome to nodejs');	Customer Name: Ninad Customer Name: Viru message: this is the first message message: this is the second message message: welcome to nodejs

Common Patterns for EventEmitters:

There are two common patterns that can be used to raise and bind an event using EventEmitter class in Node.js.

1. Return EventEmitter from afunction
2. Extend the EventEmitterclass

Return EventEmitter from a function

In this pattern, a constructor function returns an EventEmitter object, which was used to emit events inside a function. This EventEmitter object can be used to subscribe for the events. Consider the following example :

```
var emitter = require('events').EventEmitter;

function LoopProcessor(num) {
   var e = new emitter();

   setTimeout(function () {

      for (var i = 1; i <= num; i++) {
         e.emit('BeforeProcess', i);

         console.log('Processing number:' + i);

         e.emit('AfterProcess', i);
      }
   }
   , 2000)

   return e;
}
var lp = LoopProcessor(3);

lp.on('BeforeProcess', function (data) {
   console.log('About to start the process for '
+ data);
});

lp.on('AfterProcess', function (data) {
   console.log('Completed processing ' +
data);
});
```

Output:

```
About to start the process for 1
Processing number:1
Completed processing 1
About to start the process for 2
Processing number:2
Completed processing 2
About to start the process for 3
Processing number:3
Completed processing 3
```

In the above LoopProcessor() function, first we create an object of EventEmitter class and then use it to emit 'BeforeProcess' and 'AfterProcess' events. Finally, we return an object of EventEmitter from the function. So now, we can use the return value of LoopProcessor function to bind these events using on() or addListener() function.

Extend EventEmitter Class

In this pattern, we can extend the constructor function from EventEmitter class to emit the events.

Example:

```
var emitter = require('events').EventEmitter;
var util = require('util');
function LoopProcessor(num) {
   var me = this;

   setTimeout(function () {
      for (var i = 1; i <= num; i++) {
         me.emit('BeforeProcess', i);
         console.log('Processing number:' + i);
         me.emit('AfterProcess', i);
      }
   }
   , 2000)
   return this;
}

util.inherits(LoopProcessor, emitter)
var lp = new LoopProcessor(3);
lp.on('BeforeProcess', function (data) {
   console.log('About to start the process for ' + data);
});
lp.on('AfterProcess', function (data) {
   console.log('Completed processing ' + data);
});
```

Output:

```
About to start the process for 1
Processing number:1
Completed processing 1
About to start the process for 2
Processing number:2
Completed processing 2
About to start the process for 3
Processing number:3
Completed processing 3
```

In the above example, we have extended LoopProcessor constructor function with EventEmitter classusing`util.inherits()`methodofutilitymodule.So,youcanuseEventEmitter'smethods with LoopProcessor object to handle its ownevents.

In this way, you can use EventEmitter class to raise and handle custom events in Node.js.

Practice Programs:

1. Write Node.js application to create an EventEmitter which will emit an event that contains information about file upload at everysecond.
2. Write Node.js application to read 3 file contents and display message after reading each file using eventlooping.
3. Create Node.js application that display message on browser when email is received in your inbox.
4. Createnode.jsapplicationthatchangebackgroundcolour ofyour pageonbuttonclickevent

SET A

1. Create Node.js Application that binds multiple custom listeners to a singleevent.
2. Create a Node.js event-driven application that listens multiple events, and then triggers a callback function when one of those events isdetected.
3. Create Node.js application to bind custom event of receiving data from user and handles it with some listenerfunction.

SET B

1. Create node.js application that handles mouse clickevent.
2. Create node.js application that handles form submissionevent.
3. Create node.js application that change color oftext using event handling of button click.

SET C

1. Write Node.js application containing an event handler and handling event when it gets data from afile.

Signature of the instructor: _ _ _ Date: _

Assignment Evaluation:

0:NotDone	1:Incomplete	2: Late Complete	3: Needs Improvement
4:Complete	5:Well-Done		

ASSIGNMENT NO. 4: NODE.JS WITH DATABASE

Node.js can be used in database applications. One of the most popular databases is MySQL.
To be able to experiment with the code examples, you should have MySQL installed on your computer.

You can download a free MySQL database at https://dev.mysql.com/downloads/installer/

Install MySQL Driver

Once you have MySQL up and running on your computer, you can access it by using Node.js.
To access a MySQL database with Node.js, you need a MySQL driver. This tutorial will use the "mysql" module, downloaded from NPM.
To download and install the "mysql" module, open the Command Terminal and execute the following:

C:\Users*Your Name*>npm install mysql

Create Connection

Start by creating a connection to the database. Use the username and password from your MySQL database.

demo_db_connection.js

```
var mysql = require('mysql');
var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword"
});
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
});
```

Save the code in a file called "demo_db_connection.js" and run the file:
Run "demo_db_connection.js"
C:\Users*Your Name*>node demo_db_connection.js
Which will give you this result:

```
Connected!
```

Now you can start querying the database using SQL statements.

Query a Database

Use SQL statements to read from (or write to) a MySQL database. This is also called "to query" the database. The connection object created in the example above, has a method for querying the database:

```
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Result: " + result);
  });
});
```

The query method takes a sql statements as a parameter and returns the result.

Creating a Database

To create a database in MySQL, use the "CREATE DATABASE" statement:

Example

Create a database named "mydb":

```
var mysql = require('mysql');
var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword"
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  con.query("CREATE DATABASE mydb",
function (err, result) {
    if (err) throw err;
    console.log("Database created");
  });
});
```

Save the code in a file called "demo_create_db.js" and run the file:
Run "demo_create_db.js"
C:\Users*Your Name*> node demo_create_db.js
Which will give you this result:

```
Connected!
Database created
```

Creating a Table

To create a table in MySQL, use the "CREATE TABLE" statement. Make sure you define the name of the database when you create the connection:

Example : Create a table named "customers":

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  var sql = "CREATE TABLE customers
(name VARCHAR(255), address
VARCHAR(255))";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Table created");
  });
});
```

Save the code in a file called "demo_create_table.js" and run the file:
Run "demo_create_table.js"
C:\Users*Your Name*>node demo_create_table.js
Which will give you this result:

```
Connected!
Table created
```

Primary Key

When creating a table, you should also create a column with a unique key for each record.
This can be done by defining a column as "INT AUTO_INCREMENT PRIMARY KEY" which will insert a unique number for each record. Starting at 1, and increased by one for each record.
Example : Create primary key when creating the table:

```
var mysql = require('mysql');
var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  varsql="CREATETABLEcustomers(idINT
AUTO_INCREMENTPRIMARYKEY,name
VARCHAR(255), addressVARCHAR(255))";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Table created");
  });
});
```

Save the code in a file called "demo_create_table1.js" and run the file:
Run "demo_create_table.js"
C:\Users*Your Name*>node demo_create_table1.js
Which will give you this result:
Connected!
Table created

If the table already exists, use the ALTER TABLE keyword:
Example : Create primary key on an existing table:

```
var mysql = require('mysql');
var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  var sql = "ALTER TABLE customers ADD COLUMN id INT AUTO_INCREMENT
PRIMARY KEY";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Table altered");
  });
```

Insert into Table

To fill a table in MySQL, use the "INSERT INTO" statement.

Example : Insert a record in the "customers" table:

```
var mysql = require('mysql');
var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  var sql = "INSERT INTO customers (name, address) VALUES ('Company Inc', 'Highway 37')";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("1 record inserted");
  });
});
```

Save the code above in a file called "demo_db_insert.js", and run the file:

Run "demo_db_insert.js"

C:\Users*Your Name*>node demo_db_insert.js

Which will give you this result:

```
Connected!
1 record inserted
```

Insert Multiple Records

To insert more than one record, make an array containing the values, and insert a question mark in the sql, which will be replaced by the value array:

INSERT INTO customers (name, address) VALUES ?

Example

Fill the "customers" table with data:

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  var sql = "INSERT INTO customers (name, address) VALUES ?";
  var values = [
    ['John', 'Highway 71'],
```

Save the code above in a file called "demo_db_insert_multple.js", and run the file:

Run "demo_db_insert_multiple.js"

C:\Users*Your Name*>node demo_db_insert_multiple.js

Which will give you this result:

```
Connected!
Number of records inserted: 14
```

```
    ['Peter', 'Lowstreet 4'],
    ['Amy', 'Apple st 652'],
    ['Hannah', 'Mountain 21'],
    ['Michael', 'Valley 345'],
    ['Sandy', 'Ocean blvd2'],
    ['Betty', 'Green Grass1'],
    ['Richard', 'Sky st 331'],
    ['Susan', 'One way 98'],
    ['Vicky', 'Yellow Garden 2'],
    ['Ben', 'Park Lane 38'],
    ['William', 'Central st 954'],
    ['Chuck', 'Main Road 989'],
    ['Viola', 'Sideway 1633']
  ];
  con.query(sql, [values], function (err, result)
{
    if (err) throw err;
    console.log("Number of records inserted: "
+ result.affectedRows);
  });
});
```

The Result Object

When executing a query, a result object is returned. The result object contains information about how the query affected the table. The result object returned from the example above looks like this:

```
{
  fieldCount: 0,
  affectedRows: 14,
  insertId: 0,
  serverStatus: 2,
  warningCount: 0,
  message: '\'Records:14 Duplicated: 0 Warnings: 0',
  protocol41:true,
  changedRows:0
}
```

The values of the properties can be displayed like this:
Return the number of affected rows:
console.log(result.affectedRows)
Which will produce this result:
14

Get Inserted ID

For tables with an auto increment id field, you can get the id of the row you just inserted by asking the result object.

Note: To be able to get the inserted id, **only one row** can be inserted.

Example : Insert a record in the "customers" table, and return the ID:

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});
con.connect(function(err) {
  if (err) throw err;
  var sql = "INSERT INTO customers (name, address) VALUES ('Milind', 'Somatane Village 1')";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("1 record inserted, ID: " + result.insertId);
  });
});
```

Save the code in a file called "demo_db_insert_id.js", and run the file:
Run "demo_db_insert_id.js"
C:\Users*Your Name*>node demo_db_insert_id.js
Which will give you something like this in return:
1 record inserted, ID: 15

Selecting from a Table

To select data from a table in MySQL, use the "SELECT" statement.

Example : Select all records from the "customers" table, and display the result object:

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});
con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT *FROM customers", function (err,result, fields){
    if (err) throw err;
    console.log(result);
  });
});
```

Savethecodeaboveinafilecalled"demo_db_select.js" and run the file: Run "demo_db_select.js" C:\Users*Your Name*>nodedemo_db_select.js
Which will give you this result:

```
[ { id: 1, name: 'John', address: 'Highway 71'},
  { id: 2, name: 'Peter', address: 'Lowstreet 4'},
  { id: 3, name: 'Amy', address: 'Apple st 652'},
  { id: 4, name: 'Hannah', address: 'Mountain 21'},
  {id:5,name:'Michael',address:'Valley345'},
  {id:6,name: 'Sandy',address:'Oceanblvd2'},
  {id:7,name:'Betty',address:'GreenGrass1'},
  {id:8,name: 'Richard',address:'Skyst331'},
  { id: 9, name: 'Susan', address: 'One way98'},
  { id: 10, name: 'Vicky', address: 'Yellow Garden 2'},
  { id: 11, name: 'Ben', address: 'Park Lane 38'},
  {id:12,name:'William',address:'Centralst954'},
  {id:13,name:'Chuck',address:'MainRoad989'},
  { id: 14, name: 'Viola', address: 'Sideway 1633'}
]
```

Selecting Columns

To select only some of the columns in a table, use the "SELECT" statement followed by the column name.

Example : Select name and address from the "customers" table, and display the return object:

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT name, address
FROM customers", function (err, result,
fields) {
    if (err) throw err;
    console.log(result);
  });
});
```

Save the code above in a file called "demo_db_select2.js" and run the file:
Run "demo_db_select2.js"
C:\Users*Your Name*>node demo_db_select2.js
Which will give you this result:

```
[
  { name: 'John', address: 'Highway71'},
  { name: 'Peter', address: 'Lowstreet4'},
  { name: 'Amy', address: 'Apple st 652'},
  { name: 'Hannah', address: 'Mountain 21'},
  { name: 'Michael', address: 'Valley345'},
  {name:'Sandy',address:'Oceanblvd2'},
  {name: 'Betty',address:'GreenGrass1'},
  { name: 'Richard', address: 'Sky st 331'},
  { name: 'Susan', address: 'One way 98'},
  { name: 'Vicky', address: 'Yellow Garden
2'},
  { name: 'Ben', address: 'Park Lane 38'},
  { name: 'William', address: 'Central st954'},
  { name: 'Chuck', address: 'Main Road989'},
  { name: 'Viola', address: 'Sideway 1633'}
]
```

The Result Object

As you can see from the result of the example above, the result object is an array containing each row as an object.

To return e.g. the address of the third record, just refer to the third array object's address property:

Example : Return the address of the third record:

```
console.log(result[2].address);
```

Which will produce this result:

Apple st 652

The Fields Object

The third parameter of the callback function is an array containing information about each field in the result.

Example

Select all records from the "customers" table, and display the *fields* object:

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
```

Save the code above in a file called "demo_db_select_fields.js" and run the file:
Run "demo_db_select_fields.js"
C:\Users*Your Name*>node

```
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT name, address FROM
customers", function (err, result, fields) {
    if (err) throw err;
    console.log(fields);
  });
});
```

```
demo_db_select_fields.js
Which will give you this result:
[
  {
    catalog: 'def',
    db: 'mydb',
    table: 'customers',
    orgTable: 'customers',
    name: 'name',
    orgName: 'address',
    charsetNr: 33,
    length: 765,
    type: 253,
    flags: 0,
    decimals: 0,
    default: undefined,
    zeroFill: false,
    protocol41: true
  },
  {
    catalog: 'def',
    db: 'mydb',
    table: 'customers',
    orgTable: 'customers',
    name: 'address',
    orgName: 'address',
    charsetNr: 33,
    length: 765,
    type: 253,
    flags: 0,
    decimals: 0,
    default: undefined,
    zeroFill: false,
    protocol41: true
  }
]
```

As you can see from the result of the example above, the fields object is an array containing information about each field as an object.

To return e.g. the name of the second field, just refer to the second array item's name property:

Example : Return the name of the second field:

```
console.log(fields[1].name);
```

Which will produce this result:

```
address
```

Practice Programs:

1. Create a Node.js Application that Update date of birth of given employee in “employee" table and display theresult.
2. Using Node.js create Application that contains applicant details and check proper validation for (name, age, and nationality), as Name should be in upper case letters only, Age should not be less than 18 yrs and Nationality should be Indian and store the data in Licensedatabase.
3. Create Node.js application that display marksheet of student on web page after accepting his rollnumber.
4. Createnode.jsapplicationthatdisplaypurchasedetailofcustomer,after acceptingorders.

SET A

1. Create a Node.js application that demonstrate create database emp DB and employee table (eid, ename, Salary) inMySQL.
2. Create a Node.js file that Select all customers from the "customers" table who purchased only mobilephones.
3. Create a Node.js application that select all customers from the "customers" table who purchased only mobilephones.

SET B

1. Create a Node.js application that finds percentage of student whose seat number is entered through input form from resulttable.
2. Create two tables in MySQL DB product(pcode, pname, amount) andcustomer(cid, cname, pcode). Find customer names who purchasedtelevision.
3. Create node js application that accepts students details through html form such as name, address , percentage, class and store it in studenttable.

SET C

1. Create a Node.js application that create Emp, Dept &Dept-Emp tables with 1:M relationshipanddisplaythe min,max,avgsalaryofEmployeeforgivendepartment.

Signature of the instructor: _ _ _ Date: _

Assignment Evaluation:

0:NotDone	1:Incomplete	2: Late Complete	3: Needs Improvement
4:Complete	5:Well-Done		

Section-III

Advance PHP

Assignment 1: Introduction to Object Oriented Programming in PHP

Introduction:

Object-Oriented Programming (OOP) is a programming model that is based on the concept of classes and objects. As opposed to procedural programming where the focus is on writing procedures or functions that perform operations on the data, in object-oriented programming the focus is on the creations of objects which contain both data and functions together.

Object Oriented Concepts:

Before we go in detail, let's define important terms related to Object Oriented Programming.

- **Class:** Class is a programmer-defined data type, which includes local methods and local variables. A class may contain its own constants,variables (called "properties"), and functions (called "methods").
- **Object:** An individual instance of data structure defined by a class. You define a class once and then make many objects that belong to it. Objects are also known asinstance.
- **Constructor:** Constructor Functions are special type of functions which are called automatically whenever an object is created. It is a special function that initializes the properties of theclass.
- **Destructor:** Like a constructor function you can define a destructor function using function destruct (). You can release all the resources with-in adestructor.
- **Encapsulation:** Encapsulation means hiding or wrapping the code and data into a single unit to protect the data from outside world. It is used to protect class's internal data (properties and method) from code outside that class and hiding details of implementation. In PHP, encapsulation is provided by visibilityspecifiers.
- **Inheritance:** When a class is defined by inheriting existing function of a parent class then it is called inheritance. Here child class will inherit all or few member functions and variables of a parentclass.
- **Interface:** Interfaces allow you to create code which specifies which methods a class must implement, without having to define how these methods are handled. Interfaces are defined in the same way as a class, but with the *interface* keyword replacing the *class* keyword and without any of the methods having their contents defined. All methods declared in an interface must be public; this is the nature of an interface. Note that it is possible to declare a constructor in aninterface.
- **Introspection:** Introspection is the ability of a program to examine an object's characteristics, such as its name, parent class (if any), properties, and methods. With introspection, you can write code that operates on any class or object. You don't need to know which methods or properties are defined when you write your code; instead, you can discover that information at runtime, which makes it possible for you to write generic debuggers, serializes, profilers,etc.

Implementation of Object Oriented Concepts:

Function	Description	Example
class classname [extends baseclass]	Creates a class	Class student { [var $propery [= value];…] [function functionname (arguments) { //code } ….]}

$instance = new classname();	Create an object	<?php $instance1 = new myclass (); //This can also be done with a variable: $newname= 'hello'; $instance2 = new $newname(); ?>
class classname { function methodname() { Statements; } }	Add a Method	<?php class myclass { functionmymethod() { print " hello,myclass}} ?> To invoke the method on the object $instance1, we need to invoke the operator "->" to access the newly created function mymethod <?php $instance1=new myclass(); $instance1->mymethod(); ?>
void _construct ([mixed $args [, $. ..]])	**Constructor** is a function which is called right after a new object is created.	Method 1 <?php class student { public $name; public $marks; functionconstruct($nm,$mk) {
void _destruct (void)	**Destructor** is a function which is called right after you release an object.	<?php class Student { var $name; var $address; var $phone; //This is constructor function _construct() { this->name="abc"; this->address="pqr"; this->phone=1111; } functiondestruct() { echo "Student Object Released";} function printstudentinfo() {

		Echo this->name . ”\n”; echo this->address . ”\n”; echo this->phone . “\n”; }
		} $stud =new student(); $stud->printstudentinfo(); $stud=NULL; ?>
class extendedClass extends classname	**Inheritance** It is the ability of PHP to extend classes that inherit the characteristics of the parent class.	<?php class myclass { //property declaration public $var=’a default value’; //method declaration public function desplayVar() { echo $this->var; } } class extendedClass extends myclass
		{ //redefine the parent method function displayVar() { echo “Extending Class”; parent::displayVar(); } } $extend =new extendedClass(); $extend->displayVar(); ?> Output : Extending class a default value
class_exist()	**Introspection** We can use this function to determine whether a class exists.	$class = class_exists(classname);

get_declared_classes()	This function returns array of defined classes and checks if the class name is in returned array.	$classes = get_declared_classes();
get_class_methods()	We can use this function to get the methods and properties of class	$methods=get_class_methods(classna me);
get_class_vars()	This function returns only properties that have default values.	$properties=get_class_vars(classnam e);
get_parent_class()	This function is used to find the class's parent class.	$superclass=get_parent_class(classname);
is_object()	Is_object function is used to make sure that it is object.	$obj= is_obj(var);

get_class()	get_class() function is used to get the class to which an object belongs and to get class name	$classname= get_class(object);
method_exists()	This function is used to check if method on an object exists .	$method_exists=method_exists(object ,method);
get_object_vars()	This function returns an array of properties set in an object	$array=get_object_vars(object);
Interfaces	An i**nterface** is declared similar to a class but only include function prototypes (without implementation) and constants. When a class uses an interface the class must define all the methods / function of the interface otherwise the PHP engine will give you anerror. The interface's function /methods cannot have the details filled in. that is left to the class that uses the interface.	Example of an interface class duck { functionquack() { echo "quack,quack,qk, qk…"; } } Interface birds { function breath(); function eat(); } Class duck implements birds { functionquack() { echo "quack,quack,qk, qk…"; } } function breath() { echo "duck is breathing"; } function eat() { echo " duck is eating"; }

Practice Programs:

1) Write a PHP program to create class circle having radius data member and two member functions find_circumfernce () and find_area() . Display area and Circumference depending on user'spreference.
2) Create Class Collge and Class Department as base class and derived class respectively , Create one more class as Faculty to display it's detail information.(Use the concept ofinterface)
3) Write PHP script to demonstrate the concept of introspection for examiningobject.

Set A:

1) Write class declarations and member function definitions for an employee(code, name, designation). Derive emp_account(account_no, joining_date) from employee and emp_sal(basic_pay, earnings, deduction) from emp_account. Write a menu drivenprogram
   a) To build a mastertable
   b) To sort allentries
   c) To search anentry
   d) Displaysalary
2) Define an interface which has methods area( ), volume( ). Define constant PI. Create a class cylinder which implements this interface and calculate area and volume. (Hint: Use define())
3) Write a Calculator class that can accept two values, then add them, subtract them, multiply them together, or divide them onrequest.

   For example:

   $calc = new Calculator(
   3, 4); echo $calc-
   >add(); // Displays "7"
   echo $calc- >multiply(); // Displays "12"

SetB:

1) Create a class named DISTANCE with feet and inches as data members. The classhas the following member functions: convert_feet_to_inch() , convert_inch_to_feet() . Display options using radio button and display conversion on nextpage.
2) Write a PHP program to create a class Employee that contains data members as Emp_Name, Dept_name , Basic_sal,DA, HRA,TA , IT,PF,PT , GROSS, DEDUCTION ,NET . It hasmember functions calculate_gross , calculate_deductions , Calculate_net_salary . Display pay slip of employee. Create and Initialize members Emp_Name, Dept_name , Basic_sal of Employee object by using parameterized constructor.
3) Write a PHP program to create a class temperature which contains data members as Celsius and Fahrenheit . Create and Initialize all values of temperature object by using parameterized constructor . Convert Celsius to Fahrenheit and Convert Fahrenheit to Celsius using member functions. Display conversion on nextpage.

Set C:

1) Write a PHP program to create a class article having articleid, name, articleqty, price. Write menu driven program to perform following functions :(Use array ofobjects)
   i) Display details of all articlespurchased.
   ii) Display details of articles whose price exceeds500
   iii) Display details of articles whose quantity exceeds50
2) Write a PHP program to create a class Worker that has data members as Worker_Name, No_of_Days_worked, Pay_Rate. Create and initialize the object using default constructor, Parameterized constructor. Also write necessary member function to calculate and display the salary ofworker.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete []
3: Needs Improvement [ ]	4: Complete [ ]	5: well done [ ]

Signature of the Instructor

Assignment 2: To study Web Techniques

Sticky Forms:

Sticky form remembers the values you entered in the input fields. Good example of sticky form is Google search box. Sticky form helps user to type the same form again supplying the values in inputs. Sticky form is form in which the results of a query are accompanied by a search form whose default values are those of the previous query. To make sticky form , You just include the attribute value for text fields, and selected/checked for other elements:

Example :

```
<html>
<body>
<form action="<?php $_SERVER['PHP_SELF']; ?>" method="POST">
<b>Your Name : </b><input type="text" name="name" value="<?php if(isset($_POST['name'])) echo
$_POST['name'];?>">

<p><input type="submit" name="submit" value="Submit" /></p>

</form>
<?php
   echo "Your Name is =". $_POST['name']."<br>";
?>
</body>
</html>
```

Multi – Valued Parameters:

HTML selection lists, created with the select tag, can allow multiple selections. To ensure that PHP recognizes the multiple values that the browser passes to a form-processing script, you need to make the name of the field in the HTML form end with [].When PHP engine sees a submitted form field name with square brackets at the end, it creates a nested array of values within the $_GET or $_POST and $_REQUEST superglobal array, rather than a single value.

For example:

```
<select name="languages[]">
  <input name="c">C</input>
  <inputname="c++">C++</input>
  <inputname="php">PHP</input>
  <inputname="perl">Perl</input>
</select>
```

Now, when the user submits the form , $_GET['languages'] contains an array instead of a simple string. This array contains the values that were selected by the user.

Example

```
<html>
<head><title>LANGAUGES</title></head>
<body>
<form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="GET">
Select your Language :<br>
<select name="languages[]" multiple>
<option value="c"> C </option>
<option value ="c++"> C++</option>
<option value ="php"> PHP</option>
<option value ="perl"> Perl </option>
</select>
<br>
<input type="submit" name="s" value="My Languages!" />
</form>
<?php
if (array_key_exists('s', $_GET))
 {
$lang = join (" ,", $_GET['languages']);
echo "You know $lang languages.";
}
?>
</body>
</html>
```

Sticky Multi – Valued Parameters:

You can make multiple selection form elements sticky. You'll need to check to see whether each possible value in the form was one of the submitted value.
For example :

RED: <input type="checkbox" name="attributes[]" value="red" <?= if (is_array($_GET['attributes']) and in_array('red', $_GET['attributes'])) { "checked"; } ?>>

Consider following example to implement sticky multi-value parameters

```
<html>
<head><title>LANGAUGES</title></head>
<body>

<?php
$c1 = $_GET['c1'];
?>
<form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="GET">
Qualification : <br>
<input type="checkbox" name="c1[]" value="ssc" <?php if(in_array('ssc', $_GET['c1'])) {echo
"checked"; }?>> SSC <br>

<input type="checkbox" name="c1[]" value="hsc"
<?php if(is_array($_GET['c1']) and in_array('hsc', $_GET['c1'])) { echo "checked"; }?>> HSC <br>

<input type="checkbox" name="c1[]" value="bca"
<?php if(is_array($_GET['c1']) and in_array('bca', $_GET['c1'])) { echo "checked";}?>> BCA <br>

<input type="checkbox" name="c1[]" value="mca"
<?php if(is_array($_GET['c1']) and in_array('mca', $_GET['c1'])) {echo "checked";}?>> MCA <br>

<input type="submit" name="s" value="My Qualification" />
</form>
<?php
if (array_key_exists('s', $_GET))
 {
$a = join (" ,", $_GET['c1']);
echo "You Qualification : $a";
}
?>
</body>
</html>
```

Self Processing Page:

- Self processing page means one PHP page can be used to both generate a form and process it.You can use PHP_SELF variable for generating self processing page. PHP_SELF is a variable that returns the current script being executed. This variable returns the name andpath of the current file (from the root folder). You can use this variable in the action field of the FORM.
- <form name="form1" method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>">

Example A self-processing page

```
<html>
<head><title>Temperature Conversion</title></head>
<body>

<?php
 if ($_SERVER['REQUEST_METHOD'] == 'GET') {
?>

<form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="POST">
Fahrenheit temperature:
<input type="text" name="fahrenheit" /><br />
<input type="submit" name="Convert to Celsius!" />
</form>
<?php
 }
 elseif ($_SERVER['REQUEST_METHOD'] == 'POST')
 {
   $fahr = $_POST['fahrenheit'];
   $celsius = ($fahr - 32) * 5/9;
   printf("%.2fF is %.2fC", $fahr, $celsius);
 } else
{
   die("This script only works with GET and POST requests.");
 }
?>
</body>
</html>
```

Server information :

$_SERVER is a PHP super global array which holds information about the items like Server information, Header information, Details on PHP page request, File name or path information, Remote user information, HTTP Authentication Details.

Element/Code	**Description**
$_SERVER['PHP_SELF']	Returns the filename of the currently executing script
$_SERVER['GATEWAY_INTERFACE']	Returns the version of the Common Gateway Interface (CGI) the server is using
$_SERVER['SERVER_ADDR']	Returns the IP address of the host server
$_SERVER['SERVER_NAME']	Returns the name of the host server (such as www.w3schools.com)
$_SERVER['SERVER_SOFTWARE']	Returns the server identification string (such as Apache/2.2.24)
$_SERVER['SERVER_PROTOCOL']	Returns the name and revision of the information protocol (such as HTTP/1.1)
$_SERVER['REQUEST_METHOD']	Returns the request method used to access the page (such as POST)

$_SERVER['QUERY_STRING']	Returns the query string if the page is accessed via a query string
$_SERVER['HTTP_ACCEPT']	Returns the Accept header from the current request
$_SERVER['HTTP_HOST']	Returns the Host header from the current request
$_SERVER['HTTPS']	Is the script queried through a secure HTTP protocol
$_SERVER['REMOTE_ADDR']	Returns the IP address from where the user is viewing the current page
$_SERVER['REMOTE_HOST']	Returns the Host name from where the user is viewing the current page
$_SERVER['REMOTE_PORT']	Returns the port being used on the user's machine to communicate with the web serve
$_SERVER['SERVER_PORT']	Returns the port on the server machine being used by the web server for communication (such as 80
$_SERVER['SCRIPT_NAME']	Returns the path of the current script
$_SERVER['SCRIPT_URI']	Returns the URI of the current page

Example: to display server information like name , script name , user agent etc.

```
<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>
```

Output :

```
/php/demo_global_server.php
www.w3schools.comwww.w
3schools.com
https://www.w3schools.com/php/showphp.asp?filename=demo_global_server
Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.112
Safari/537.36
/php/demo_global_server.php
```

Practice Programs:

1) Write a PHP script to Design a form to accept a number from the user to check whether number is palindrome or not. (Use the concept of self processingpage)
2) Write PHP program to accept user details such as user-id, name, Address, email, and mobile no. Display same information on nextpage.
3) Write PHP program to create student registration form and display student information. (Use sticky formconcept).

Set A:

1) Write PHP program accept name, select your cities you would like to visit and display selected information on page. (Use multi-valuedparameter),.
2) Write PHP program to create student registration form and display student information. (Use sticky formconcept).
3) Write a PHP script for the following: Design a form to accept a number from theuser. Perform the operations and show theresults.
    - Check whether number is palindrome ornot.
    - Reverse the number using recursions.

   (Use the concept of self processing page.)
4) Write PHP program to select list of subjects from list box and display selected subject on information. (Use sticky multi-valuedparameter)

Set B:

1) Write a PHP Script to display Server information in table format (Use$_SERVER).
2) Write a PHP program to accept two strings from user and check whether entered strings are matching or not. (Use sticky formconcept).
3) Write a PHP script to accept an Indian currency and then convert it in dollar orpounds (radio buttons) according to user's preference. (use concept of self processingform).
4) Write PHP program to accept client name, property details (Flat, Bunglow, Plot), Display selected information same page. (Use multi- valueparameter).

Set C:

1) Write PHP program to accept name of student , Gender(male ,female) using radio buttons ,Qualification(SSC, HSC, BCA, MCA) using check boxes . Display information of student. (Use sticky multi-valued parameter).

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete []
3: Needs Improvement [ ]	4: Complete [ ]	5: well done [ ]

Signature of the Instructor

Assignment 3 – XML

Introduction to XML:

XML stands for eXtensible Markup Language. It is a text-based markup language derived from Standard Generalized Markup Language (SGML). XML was designed to store and transport data. XML was designed to be both human- and machine-readable. XML is a markup language much like HTML. XML was designed to describe data. XML tags are not predefined . You must define your own tags.XML is self describing.

XML document are well – formed and valid. A well - formed XML document follows the basic XML syntax rules.A valid document also follows the rules imposed by a DTD or an XSD.

A simple document is shown in the following example −

```
<?xml version = "1.0"?>
<contact-info>
   <name>Tanmay Patil</name>
   <company>TutorialsPoint</company>
   <phone>(011) 123-4567</phone>
</contact-info>
```

The following image depicts the parts of XML document.

Document Prolog Section :
Document Prolog comes at the top of the document, before the root element. This section contains −

- XMLdeclaration
- Document typedeclaration

Document Elements Section:
Document Elements are the building blocks of XML. These divide the document into a hierarchy of sections, each serving a specific purpose.

XML declaration :
It contains details that prepare an XML processor to parse the XML document. It is optional, but when used, it must appear in the first line of the XML document.

<?xml version="version_number" encoding="encoding_declaration" standalone="standalone_status" ?>

An XML declaration should abide with the following rules:

- The XML declaration is case sensitive and must begin with "<?xml>" where "xml" iswritten in lower-case. If the XML declaration is included, it must contain version numberattribute.
- The Parameter names and values are case-sensitive.The names are always in lowercase.
- The order of placing the parameters is important. The correct order is:*version, encoding and standalone.* Either single or double quotes may beused.
- The XML declaration has no closing tag i.e.</?xml>

Example of XML declaration:

- <?xml>
- <?xmlversion="1.0">
- <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <?xml version='1.0' encoding='iso-8859-1' standalone='no'?>

DTD :Document Type Declaration :

- The XML Document Type Declaration, commonly known as DTD, is a way to describe XML languageprecisely.
- DTDs check vocabulary and validity of the structure of XML documents against grammatical rules of appropriate XMLlanguage.
- An XML DTD can be either specified inside the document, or it can be kept ina separate document and then likedseparately.
- Basic syntax of a DTD is asfollows:

```
<!DOCTYPE element DTD identifier
 [
   declaration1
   declaration2
  ........
]>
```

XML Tags :

XML tags are case sensitive. The tag <Letter> is different from the tag <letter>. Opening and closing tags must be written with the same case.

For example,

<Message>This is incorrect</message>

<message>This is correct</message>

XML Elements :

- An XML file is structured by several XML-elements, also called XML-nodes or XML-tags. XML-elements' names are enclosed by triangular brackets <>.
- EachXML-elementneedstobeclosedeitherwithstartorwithendelementsasshownbelow: <element>....</element>.
- An XML document can have only one rootelement
- An XML-element can contain multiple XML-elements as its children, but the children elements must notoverlap.
- In XML, all elements must be properly nested within eachother.

XML attributes:

- An XML-element can have one or more attributes.
- Attribute names in XML (unlike HTML) are case sensitive. That is, *HREF* and*href*are considered two different XMLattributes.
- Same attribute **cannot have two values in asyntax**

So XML follows tree structure

```
<root>
  <child>
```

```
            <subchild> .... </subchild>
          </child>
        </root>
<?xml version = “1.0” ?>
<BookStore>
    <Books>
            <PHP>
                    <title>Programming PHP</title>
                    <publication>O’RELLY</publication>
            </PHP
            >
            <PHP><title>Beginners PHP</title>
                    <publication>WROX</publication>

            </PHP
            >
    </Books>
</BookStore>
```

SimpleXML :

- SimpleXML is an extension that allows us to easily manipulate and get XMLdata.
- The SimpleXML extension is the tool of choice for parsing an XMLdocument.
- SimpleXML turns an XML document into a data structure you can iterate through likea collection of arrays andobjects.
- The SimpleXML extension includes interoperability with the DOM for writing XML filesand built-in XPathsupport.
- SimpleXML is easier to code than the DOM, as its nameimplies.

SimpleXMLElement class represents an element in an XML document.

- To create root element of xml document, first create object of SimpleXMLElement class and initialize with rootelement.
- For example :
- $bk=newSimpleXMLElement(“<bookstore/>”);

Methods or functions of simpleXMLElement class

Function name	description	syntax	example
addChild()	The addChild() function adds a child element to the SimpleXML elemen	addChild(name,value);	$book = $bk->addchild(“book”);
addAttribute()	adds an attribute to the SimpleXML element.	addAttribute(name, value);	$book->addAttribute(“Category” , “Technical”);
getName()	Returns the name of the XML tag referenced by the SimpleXML element	getName();	$bk->getName();
asXML()	Returns a well-formed XML string (XML version 1.0) from a SimpleXML object	asXML([filename]);	echo $bk->asXML();
children()	Returns the children of a specified node as an array	children()	foreach ($book->children() as $child) { echo "Child node: " . $child . " "; }
attributes();	Returns the attributes/values of an element	attributes();	foreach ($book->attributes () as $k=>$v) { echo $k : $v . " "; }
count();	The count() function counts the children of a specified node.	count();	$cnt=$book->count();

simplexml_load_file()	Converts an XML file into a SimpleXMLEle me nt object	simplexml_load_file(file)	$xml=simplexml_load_file("note.xml");
simplexml_load_string()	The simplexml_load_str ing() function converts a well-formed XML string into a SimpleXMLElem e nt object.		<?php $note=<<<XML <note> <to>Tove</to> </note> XML; $xml=simplexml_load_string($note);

Reading XML document

```
<?php
$bk = simplexml_load_file("book.xml");
echo htmlspecialchars($bk->asXML());
?>
```

- With SimpleXML, all the elements in XML document are represented as tree of SimpleXMLElementobjects.Anygivenelement'schildrenareavailableaspropertiesof elements SimpleXMLElement object.

- For example ,We can access element name as properties $book->title , $book->publisheretc.

Consider an application that reads "Book.xml" file into simple XML object. Display attributes and elements.

```
        //book .xml
<?xml version='1.0' encoding='UTF-8'?>
<bookstore>
<book category="Technical">
<title> LET US C </title>
<author> YASHWANT KANETKAR </author>
<year> 1980 </year>
</book>
<book category="Cooking">
<title> COOKING EVERYDAY </title>
<author> TARALA DALAL </author>
<year> 2000 </year>
</book>
<book category="YOGA">
<title> LIGHT ON YOGA </title>
<author> B.K.IYENGAR </author>
<year> 1990 </year>
</book>
</bookstore>

// book.php
<?php
$xml = simplexml_load_file("book.xml");
echo $xml->getName() . "<br />";
foreach($xml->children() as $child)
  {
   echo $child->getName() . "<br>";
```

```
   foreach($child->attributes() as $k=>$v)
   {
    echo $k . "=". $v . "<br>";
    foreach($child->children() as $i=>$j)
    {
          echo $i .":". $j."<br>";

    }
  }
}
?>
```

Practice Programs:

1) Write a XML program which shows how you can easily read and display the contents of an XML document using SimpleXML.
2) Write a script to create "Company.xml" file with multiple elements as shownbelow:

```
<CemployeeTeam>
   <Team Name="Red">
        <Ename>____</ Ename>
         <Eexperience>_____</ Eexperience >
         <Emobno>___</ Emobno>
         <Eaddress>_____</Eaddress>
   </Team>
</CemployeeTeam>
```

3) Write a PHP Script to read book.XML and print book details in tabular format using simple XML.(Content of book.XML are (bookcode , bookname , author , year ,price).

Set A:

1) Write a PHP script to create XML file named"Course.xml"

```
<Course>
 <SYBBA CA>
  <Studentname> .....</Studentname>
  <Classname>..... </Class name>
  <percentage>.... </percentage>
   </SYBBA CA>
  </Course>
```

Store the details of 5 students who are in SYBBACA.

2) Write PHP script to generate an XML code in the followingformat

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<CATALOG>
<CD>
   <TITLE>Empire Burlesque</TITLE>
   <ARTIST>Bob Dylan</ARTIST>
   <COUNTRY>USA</COUNTRY>
   <COMPANY>Columbia</COMPANY>
   <PRICE>10.90</PRICE>
   <YEAR>1985</YEAR>
</CD>
</CATALOG>
```

Save the file with name "CD**.xml".**

3) Write PHP script to generate an XML code in the followingformat

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<CATALOG>
```

<PLANT>
<BOTANICAL>Sanguinaria canadensis</BOTANICAL>
<ZONE>4</ZONE>
<LIGHT>Mostly Shady</LIGHT>
<PRICE>$2.44</PRICE>
<AVAILABILITY>031599</AVAILABILITY></PLANT></CATAOG>

Save the file with name "plant**.xml".**

Set B:

1) Write a script to create "cricket.xml" file with multiple elements as shownbelow:

<CricketTeam>
<Team country="India">
<player>_____</player>
<runs>_____</runs>
<wicket>____</wicket>
</Team>
</CricketTeam>

Write a script to add multiple elements in "cricket.xml" file of category, country="Australia".

2) Write a script to create "breakfast.xml" file with multiple elements as shownbelow:

<breakfast_menu>
<food>
<name>French Fries</name>
<price>Rs45</price>
<description>Young youths are very much intrested to eat it </description>
<calories>650</calories>
</food>
</breakfast_menu>

Write a script to add multiple elements in "breakfast.xml" file of category, Juice.

3) Create a XML file which gives details of movies available in "Mayanagari CD Store" from followingcategories
   a) Classical
   b) Action
   c) Horror

Elements in each category are in the following format

<Category>
<MovieName> ----</MovieName>
<ReleaseYear> ---- </ReleaseYear>
</Category>

Save the file with name "movies.xml".

Set C:

1) Create an application that reads "book.xml" file into simple XML object. Display attributes and elements (Hint:simple_xml_load_file() function).

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete []
3: Needs Improvement [ ]	4: Complete [ ]	5: well done [ ]

Signature of the Instructor

Assignment 4: PHP with AJAX

AJAX stands for **Asynchronous JavaScript and XML**. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script. When a user wants more content, they click a link. With AJAX, a user can click something and content can be loaded into the page, using JavaScript, **without reloading the entirepage.**

Conventional web applications transmit information to and from the sever using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server. With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and Update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.

AJAX cannot work independently. It is used in combination with other technologies to create interactive WebPages.

1) **JavaScript**
   - ✓ Loosely typed scriptinglanguage.
   - ✓ JavaScript function is called when an event occurs in apage.
   - ✓ Glue for the whole AJAXoperation.
2) **DOM**
   - ✓ API for accessing and manipulating structureddocuments.
   - ✓ Represents the structure of XML and HTMLdocuments.
3) **CSS**
   - ✓ Allows for a clear separation of the presentation style from the content andmay be changed programmatically byJavaScript.
4) **XMLHttpRequest**
   - ✓ JavaScript object that performs asynchronous interaction with theserver.

XMLHttpRequest is a JavaScript object capable of calling the server and capturing its response. It is used to send HTTP or HTTPS requests to a web server and load the server response data backinto thescript.

Creating an XMLHttpRequest Object :
All modern browsers (IE7+, Firefox, Chrome, Safari, and Opera) have a builtin XMLHttpRequest object.

Syntax for creating an XMLHttpRequest object:
xmlhttp=newXMLHttpRequest();
When a request to a server is sent, we want to perform some actions based on the response.
The onreadystatechange event is triggered every time the readyState changes. The readyState property holds the status of the XMLHttpRequest.

Three important properties of the XMLHttpRequest object:

- **readyState** :The readyState property defines the current state of the XMLHttpRequestobject.

The following table provides a list of the possible values for the readyState property:

State	Description
0	The request is notinitialized.
1	The request has been setup.
2	The request has beensent.
3	The request is in process
4	The request iscompleted.

- **OnReadyStateChange :** Determine the function called when the objects readyState changes.

```
xmlobj.onreadystatechange=function()
 {
}
```

- **responseText:**Returns the response as astring.
- **responseXML:**Returns the response as XML. This property returns an XML documentobject, which can be examined and parsed using the W3C DOM node tree methods andproperties.
- **Status:**Returns the status as a number (e.g., 404 for "Not Found" and 200 for"OK").
- **statusText:**Returns the status as a string (e.g., "Not Found" or"OK").
- **Methods of XMLHttpRequest object:**

To send a request to a server, we use the open() and send() methods of the XMLHttpRequest objec

- **open(method, URL, async)**

Specifies the method, URL, and other optional attributes of a request. The method parameter can have a value of "GET", "POST", or "HEAD". The "async" parameter specifies whether the request should be handled asynchronously or not. "true" means that the script processing carries on after the send() method without waiting for a response, and "false" means that the script waits for a response before continuing script processing.

- **send(content):** Sends therequest.
- **abort()**Cancels the currentrequest.

Practice Programs:

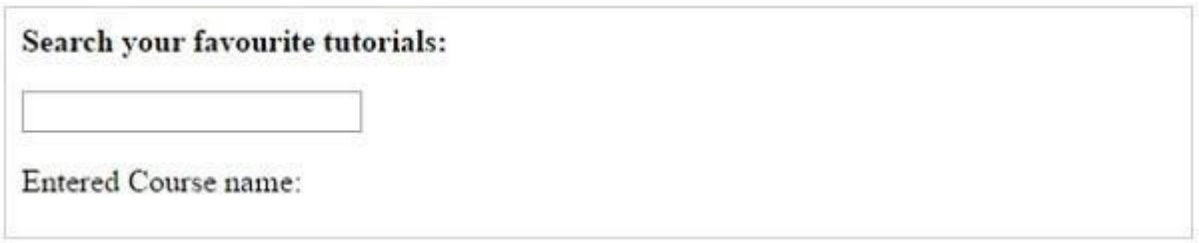
1) Write a simple PHP program which implements Ajax for addition of twonumbers.
2) Write an Ajax program to display list of games stored in an array on clicking OKbutton.
3) Write an Ajax program to read a text file and print the contents of the file when user click on the print button.(consider "a.txt" file to create text & write text as "Ajax Example" init.)

Set A:

1) Write a PHP script using AJAX concept, to check user name and password are valid or Invalid (use database to store user name andpassword).
2) Write Ajax program to carry out validation for a username entered in textbox. If the textbox is blank, print 'Enter username'. If the number of characters is less than three,print' Username is too short'. If value entered is appropriate the print 'Validusername'.
3) Write Ajax program to get book details from XML file when user select a book name. Create XML file for storing details of book(title, author, year,price).

Set B:

1) Write Ajax program to fetch suggestions when is user is typing in atextbox. (eg like google suggestions.Hint create array of suggestions and matching string will be displayed)
2) Write Ajax program to get player details from XML file when user select a player name. Create XML file for storing details of player (Country, player name, wickets, runs).
3) Write a AJAX program to display the following output to search your favourite tutorial from "tutorial.php"file.



Set C:

1) Write a AJAX program to display the selected course information from the list given in XML file and show the following output.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete []
3: Needs Improvement [ ]	4: Complete [ ]	5: well done [ ]

Signature of the Instructor

Assignment 5: Connecting Database using PHP & AJAX

Fetch Data from MySQL table using PHP

To fetch data from the MySQL database, configure the following steps –

1) First, Include database connection filedatabase.php
2) Assign connection variable $conn to a new variable$db
3) Create a custom function fetch_data(). This function will return data to fetch from the database. Then call fetch_data() and assign it to a new variable$fetchData.
4) Also, Create another custom function show_data($fetchData). This function returns data with an HTMLtable.
5) Callshow_data($fetchData).
6) This function accepts a parameter to get fetched data.
   File Name –backe

Function name	description	syntax	example
mysql_connect	Open aconnection to aMySQL Server	mysqli_connect([$host, $username, $passwd, $dname, $port, $socket])	$conn = new mysqli($servername, $username, $password);
mysql_create_db	Create a MySQLdatabase	mysql_create_db (string $database_name ,resource$link_identifier = NULL) : bool	$sql = 'CREATE DATABASE my_db';
mysql_error	Returns the text ofthe error messagefrom previousMySQL operation	mysql_error (resource $link_identifier = NULL) : string	echo mysql_errno($link) . ": " . mysql_error($link). "\n";
mysql_fetch_row	Get a result rowas an enumeratedarray	mysql_fetch_row (resource $result) : array	$row = mysql_fetch_row($result);
mysql_db_query	Selects a databaseand executes aquery on it	mysql_db_query (string $database , string $query,resource$link_identifier=NULL) : resource|bool	$result = mysql_query($sql, $link);
select_db()	used to change the default database for the connection.	$mysqli -> select_db(*name*)	mysqli_select_db($con, "test");
mysql_close	Close MySQLconnection	mysql_close (resource $link_identifier = NULL) : bool	mysql_close($link);

Practice Programs:

1) Write an Ajax program to display list of book stored in an array on clicking ok button. (ConsiderBook_List.php)
2) Write an Ajax program to search the book name according the character typed & display same list using array. (UseNew.php)
3) Write an Ajax program to display list of games stored in an array on clicking okbutton.

Set A:

1) Write Ajax program to print Movie details by selecting an Actor's name. Create table MOVIE and ACTOR as follows with 1 : M cardinality MOVIE (mno, mname, release_yr) and ACTOR(ano, aname).
2) Create Trip table as follows

   Trip (tno, tname, Source, Destination, cost). Write Ajax program to select the trip name and print the selected trip details.
3) Create student table as follows Student(sno, sname, per).

   Write Ajax program to select the student name and print the selected student's details.

Set B:

1) Write Ajax program to get player details from player table by inserting a player name at run time display it's details in tabular form .Consider ,

   player (Country, player_name, wickets, runs).
2) Write Ajax program to calculate maximum runs scored for a particular country (Use Above Playertable).

Set C:

1) Write Ajax program to get details of voters whose vage is greater than 40 year from Voter table Create voter table as Voter (vid, vname, vage,vaddress).

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete []
3: Needs Improvement [ ]	4: Complete [ ]	5: well done [ ]

Signature of the Instructor

Assignment 5: PHP Framework - Druple

Drupal is open source software that allows publishing, managing and organizing a wide variety of content on a website easier. Drupal is used to easily manage, update and publish the content in the website. Many individuals and organizations are using Drupal to create professional websites to suit their custom requirements. Because of easy creating sites, application and management, Drupal is used by many organizations. We can enhance the functionality of Drupal by adding available add-on modules.

Creating Contents

You can add two types of contents in your website: Article and Basic Page.
To create content click the link "Add content." From the short cut menu.
Choose between Article and Basic page.

Creating Articles

Content type Article has the following features:

- Summary posted to the front page of the Website.
- Commentsenabled.
- An image can be displayed with thearticle.
- User name of the article author as well as the time it was originallypublished.
- Tags enabled, allowing you to categorize articles.

To create an article, do the followingsteps:

- From the shortcut menu, click the link "Add content." An overlay will appear prompting you to choose between Article and Basicpage.
- Click"Article".
- Enter a title and body for yourpage.
- Scroll to the bottom and clickSave.

Creating Basic Page

Content type Basic Page has the following features:

- Are not published to the front page of your Website.
- Do not allow visitors to postcomments.
- Do not have taggingenabled.
- Do not have an image uploadwidget.
- Are notdate-stamped.

To create a Basic Page, do the following steps:

- From the shortcut menu, click the link "Add content." An overlay will appear prompting you to choose between Article and Basicpage.
- Click "Basicpage".
- Enter a title and body for yourpage.
- Scroll to the bottom and clickSave.

Customizing the Display

Use the following steps to change the theme and logo image of yourwebsite:

- Using the administrative dashboard, click the tabAppearance.
- Scroll down to the bottom of the screen (where all the disabled themes live), and beneath your theme's screen shot, click the link Enable and setdefault."

After the screen refreshes, click the settings link for your theme.

- Scroll to the fieldset "Logo image settings." Unselect the check box "Use the default logo." A new set of settings will berevealed.
- Click Browse and find your logo image for this theme on your harddrive.
- Scroll to the bottom of the screen and click "Saveconfiguration."

Blocks

- Blocks can be placed into any region in your theme.

To create Block, do the followingsteps:

- Using the administrative dashboard, navigate to Structure >Blocks.
- Click the link “Addblock”.
- Enter description and the text.
- Scroll to the bottom and click “Saveblock.”

Modules

The modules are used to create, edit, and delete content; convert URLs into specific database requests to retrieve content; and create the menus you use to navigate your Web site.

Modules are little programs that allow you to do more things with your Website. Modules are set of files contained in a Drupal folder. These files may include the following:

- An information file that describes the module to Drupal. This file lists the version, files within the module directory, configuration screen shots, and a short description of the module. This file isrequired.
- Installation instructions for Drupal that create the necessary database tables for the module. This file isrequired.
- PHP scripts that hook into Drupal and allow you to perform specifictasks.
- Template files responsible for the output of the module. These template files can be altered by your theme. These files areoptional.
- CSS files, JavaScript files, and images. These files areoptional.

Practice Programs:

1) Create a Page in Drupal titles “Game”. Add the details of different games (football, hockey, and cricket) with player list on the page. The page should contain announcement about upcoming match.
2) Create a module in Drupal To design a form with the following components:
   Item - Ino, IName, andRate
   One submit button.

   After submitting the form insert a Item record into a table named Item. Also display a message when the record is inserted successfully, and fetch the Item from the table and display “<Iname>=<Rate>. Also add Navigation on the Home Page called “Item Rate”.
3) Using Drupal create a module containing details of your college. On the home page add Navigation which contains your college name and also add logoimage.

Set A:

1) Create a Basic Page in Drupal titled “About Me”. Add the details about yourself in the page. Also place this page link in the Main Menu. Display this menu link before all the menu items. Show text “This is <your name>” when move the mouse pointer at this menulink.
2) Develop a module in Drupal to create a page showing your contact details (name, roll_no, address, phone). Also add Navigation on the Home Page called “ContactDetails”.
3) Using Drupal create a page showing the teacher details (name, contactno, subjecttaught). Add Navigation on the home page called “TeacherDetails”

Set B:

1) Create a Block in Drupal titled “Event”. The block should be displayed in the left side of each page. The block should contain announcement about an upcoming events. Also change the theme of your website by followingproperties:
   a) Change the backgroundcolour.
   b) Change the logoimage.
2) Create a front-page article in Drupal titled “My Article”. Write an article about PHP programming Language and add to the article page. Display an Image appropriate to the Article at the bottom of the Article. Also place this page link in the Main Menu. Display this menu link before all the menu items. Show text “This is <your name>” when move the mouse pointer at this menu link. Also post a commentabout.
3) Create a module in Drupal To design a form with the followingcomponents:

Text Fields - Roll No, Name, and Address
One submit button.

After submitting the form insert a student record into a table named student. Also display a message when the record is inserted successfully, and fetch the name of student from the table and display "Hello: <student name>". Also add Navigation on the Home Page called "Student Form".

Set C:

1) Develop a module in Drupal to design a registration form with the following fields:
Text Field – First Name, Last Name, email, city List Boxes – Select Country, Date of Birth (Separate Select Boxes for month, day, and year)
Radio Buttons – Gender - Male/Female
Check Boxes – Technology Known – Java, PHP
One Browse button to upload picture.
Perform validation to check if the First Name and Last Name are not empty and the email is valid. If that is not the case display error message and the form will not be submitted.
Display message "Form has been submitted successfully" after clicking on the Submit button.
Also add a Navigation on the Home Page called "Registration".

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete []
3: Needs Improvement [ ]	4: Complete [ ]	5: well done [ ]

Signature of the Instructor

Add-On
jQuery

Savitribai Phule Pune University
Syllabus for B.B.A (CA) (CBCS 2019 Pattern) Semester IV
Subject Code: - 407
Subject Name -: jQuery
Practical Assignments

1. Write a jQuery code to check whether jQuery is loaded or not.
2. Write a jQuery code to scroll web page from top to bottom and vice versa.
3. Write a jQuery code to disable right click menu in html page.
4. Write a jQuery code to disable the submit button until the visitor has clicked a check box.
5. Write a jQuery code to fix broken images automatically.
6. Write a jQuery code to blink text continuously.
7. Write a jQuery code to create a zebra stripes table effect.
8. Write a jQuery code to print a page.
9. Write a jQuery code to allow the user to enter only 15 characters into the textbox.
10. Write a jQuery code to make first word of each statement to bold.
11. Write a jQuery code to create a division (div tag) using jQuery with style tag.
12. Write a jQuery code to select values from a JSON object.
13. Write a jQuery code to add list elements within an unordered list element.
14. Write a jQuery code to remove all the options of a select box and then add one option and select it.
15. Write a jQuery code to underline all the words of a text.
16. Write a jQuery code to demonstrate how to get the value of a textbox.
17. Write a jQuery code to remove all CSS classes from an application.
18. Write a jQuery code to distinguish between left and right mouse click.
19. Write a jQuery code to check if an object is a jQuery object or not.
20. Write a jQuery code to detect whether the user has pressed 'Enter key' or not.
21. Write a jQuery code to count number of rows and columns in a table.
22. Write a jQuery code to display form data onto the browser.
23. Write a jQuery code to find absolute position of an element.
24. Write a jQuery code to remove a specific value from an array.
25. Write a jQuery code to change button text.
26. Write a jQuery code to add options to a drop-down list.
27. Write a jQuery code to set background-image to the page.
28. Write a jQuery code to get the selected value and currently selected text of a dropdown box.
29. Write a jQuery code to disable a link.
30. Write a jQuery code to Restrict "number"-only input for textboxes including decimal points.
31. Write a jQuery code to set value in input text.