



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

Editorial Board: Dr. D. Y. Patil ACS College, Pimpri, Pune

Section-I: Data Structure

Mrs. Madhuri Darekar.

Mrs. Ashwini Patil.

Section-II: Angular JS

Mr. Bhushan Nikam.

Mr. Satish Mulgi.

Section-III: PHP

Ms. Neeta Takawale.

Mrs. Sunayna Shivthare.

Section-IV: Big Data

Mrs. Sonali Nemade.

Mr. Yogesh Ingale.

Section-V: Block Chain

Mrs. Malati Tribhuwan.

Mr. Satyavan Kunjir.

Reviewed By:

Dr. Ranjit Patil.

Mrs. Sujata Patil.

Mrs. Sangeeta Nimbalkar.

Mrs. Leena Bhat.

Mr. Sudarshan Lakhdiv.

Mrs. Shakila Siddavatam.

Mr. Shivendu Bhushan.

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	String 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 $(ax^n + bx^{n-1} + \dots + jx + k)$, where $a, b, c, \dots, k$ 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 []

3: Needs Improvement []

1: Incomplete []

4: Complete []

2: Late 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: $O(n^2)$, Average Case: $O(n^2)$

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 step else 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 1st element with 0th
- In second iteration 2nd element is compared with the 0th and 1st 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: $O(n^2)$ Average Case: $O(n^2)$

Algorithm for Insertion Sort:

Step1: Start
 Step2: Accept 'n' numbers and store all in array 'A'
 Step3: set $i=1$
 Step4: if $i \leq n-1$ then goto next step else goto step 10
 Step5: set $Temp=A[i]$ and $j=i-1$
 Step6: if $Temp < A[j]$ && $j \geq 0$ 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 0th element is compared with all other elements, if the 0th 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 0th position. The Procedure is repeated for 1st 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: $O(n^2)$ Worst Case: $O(n^2)$ Average Case: $O(n^2)$

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 $A[j] < A[\text{min}]$ then $\text{min} = j$
 Step8: set $j = j + 1$ and goto step 7
 Step9: if (min not equal to i) then interchange $A[i]$ and $A[\text{min}]$
 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. $up \leq down$, 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 = $O(n \log n)$
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 atleast 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 "Required 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 $low \leq high$ 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

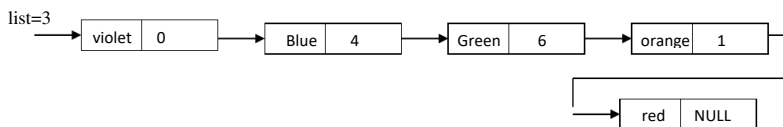
Link array	
0	4
1	-1
2	
3	0
4	6
5	
6	1

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:-

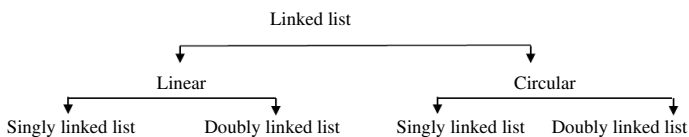
- The static representation uses arrays which is a static data structure and has its own limitations.
- A linked list is a dynamic data structure i.e. the size should increase and when elements are deleted, its size should decrease.
- This cannot be possible using an array which uses static memory allocation 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



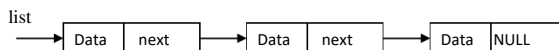
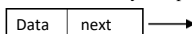
1) Linear linked list

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.



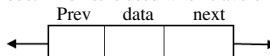
➤ Singly linked list-

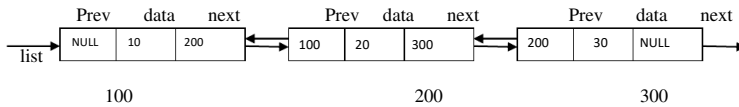
Each node in this list contains only one pointer which points to the next node.



➤ Doubly linked list

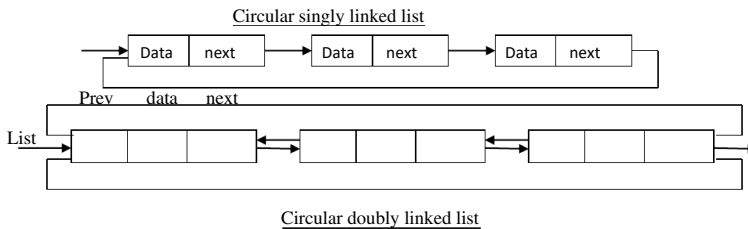
Each node in this contains two pointers, one pointing to the previous node and the other pointing to the next node. This list is used when traversing in both directions is required.





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.



• 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 be 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:**Algorithm**

- 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 []

3: Needs Improvement []

1: Incomplete []

4: Complete []

2: Late 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 is empty and false otherwise
- 5) isfull:
Check whether a queue is full. The operation return true if the queue is full 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 $n \geq 0$ disjoint sets $T_1, \dots, T_n$ where each of these sets is a tree. $T_1, \dots, T_n$ 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 – $\text{left_subtree}(\text{keys}) < \text{node}(\text{key}) \leq \text{right_subtree}(\text{keys})$

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 $n \times n$ 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 i th vertex

Outdegree(i) – returns the outdegree (the number of edges moving out) of the i th 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 is 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 outdegree 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 []

3: Needs Improvement []

1: Incomplete []

4: Complete []

2: Late Complete []

5: Well Done []

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.

```

<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>

```

<pre> </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> </pre>
<pre> 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> </pre>
<pre> 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> </pre>

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 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...

Example 1 : ng-click Event <pre> <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" />

 <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> </pre>	Example 2 : Mouse & Button Event <pre> <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> </pre>
---	---

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-binding 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:

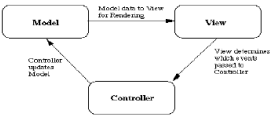
- 1) Write a HTML code using Angular JS to generate the following output
Undergraduate Courses (hint : use ng-repeat, ng-init directive)
 - i. BBA(CA)
 - ii. BCA(Science)
 - iii. B.Sc.(Computer Science)Post Graduate Courses
 - i. M.Sc.(Comp.Sci.)
 - ii. M.Sc.(CA)
 - iii. MCA
- 2) Write an AngularJS script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

Assignment Evaluation**0: Not Done** []**3: Need Improvement** []**1: Incomplete** []**4: Complete** []**2: Late 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.

<pre><html> <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/ angular.min.js"> </script> <body> <div ng-app="myApp" ng-controller="myCtrl"> First Name: <input type="text" ng- model="firstName">
 Last Name: <input type="text" ng- model="lastName">

 Full Name: {{fullName()}} <!--Function Calling -- > </div> <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;</pre>	Here, the AngularJS application runs inside the <div> 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 <pre> <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> </td> </tr> <tr> <td> Total </td> <td>{{ student.tot }}</td> </table> </div> </pre>	<pre> <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> </pre>
---	--

Example : To Print Simple ETicket

<pre> <html> <head> <script src = "http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"> </script> </head> <body> <h2>AngularJS eTicect Application</h2> <div ng-app = "mainApp" ng-controller = "eTicectController"> <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">Mumbai <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> </pre>	<pre> <table border = 2 align = center> <tr><td colspan = 3 align = center> Indian Railway Conter</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('eTicectContr oller', function(\$scope) { \$scope.name = "Satish" \$scope.address = "Pimpri" \$scope.contact = "9028813474" \$scope.source = "Pune" \$scope.destination = "Bidar" \$scope.jDate = "10/10/2020" }); </script> </pre>
---	---

	</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 University 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 <pre><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></pre>	Example 2 : The lowercase filter format strings to lower <pre><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></pre>
--	--

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: <pre><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> <li ng-repeat="x in names orderBy:'country'"> {{ x.name + ', ' + x.country }} </div> <script> angular.module('myApp', []) .controller('namesCtrl', function(\$scope)</pre>	Example 6: Date <pre><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>The following date format is the by default date format.</p> <p>Date = {{ today date }}</p> <p>You can write the date in many different formats.</p> <p>Date = {{ today date : "dd.MM.y" }}</p> <p>You can use predefined formats when displaying a date.</p> <p>Date = {{ today date : "fullDate" }}</p> <p>This is another format.</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></pre>
---	--

<pre>{ \$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></pre>	Example 4: The currency filter formats a number to a currency format <pre><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></pre>
Example 5 : Filter <pre><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:'912032322'}, { 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></pre>	<pre><<label>Any: <input ng- model="search.\$"></label>
 <label>Name only <input ng- model="search.name"></label>
 <label>Phone only <input ng- model="search.phone"></label>
 <label>Equality <input type="checkbox" ng- model="strict"></label>
 <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></pre>

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 <pre><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> The name is required. </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></pre>	

Form Validation <pre> <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> First Name is required. </td></tr> <tr><td>Enter last name: </td> <td><input name = "lastname" type = "text" ng- model = "lastName" required> Last Name is required. </td></tr> </pre>	<pre> <tr> <td>Email: </td><td><input name = "email" type = "email" ng-model = "email" length = "100" required> Email is required. Invalid email address. </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@dyppv.edu.in"; } \$scope.reset(); }); </script> </body> </html> </pre>
--	---

Practice Problems:

- 1) Create student registration form and validate all fields
- 2) Store 10 students information in an array and display students information in table form
orderBy Name (use orderBy 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** []**3: Need Improvement** []**1: Incomplete** []**4: Complete** []**2: Late 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: <pre><html> <script src="https://ajax.googleapis.com/ajax/libs/angular js/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></pre>	The \$timeout Service The \$timeout service is AngularJS' version of the window.setTimeout function. <pre><html><script src="https://ajax.googleapis.com/ajax/libs/angularj s/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></pre>
---	--

The \$interval Service The \$interval service is AngularJS' version of the window.setInterval function. <pre> <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> </pre>	Create Your Own Service To create your own service, connect your service to the module: <pre> <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> </pre>
The AngularJS \$http service makes a request to the server, and returns a response. <pre> <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> </pre>	

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 Date 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** []**3: Need Improvement** []**1: Incomplete** []**4: Complete** []**2: Late 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:** Students 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 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 []

3: Need Improvement []

1: Incomplete []

4: Complete []

2: Late 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.	<pre>if(Condition) { Statements; }</pre>	<pre><?php \$n=10; if (\$n%2==0) { echo "Number Is Even"; } ?></pre> 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.	<pre>if(Condition) { Statements; } else { Statements; }</pre>	<pre><?php \$n=10; if (\$n%2==0) { echo "Number Is Even";} else { echo "Number Is Odd";} ?></pre> Output: Number Is Even
elseif Statement: It checks more than one condition.	<pre>if (condition) { Statements;} elseif (condition) { Statements;} else { Statements;}</pre>	<pre><?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";} ?></pre> Output: a is less than b
switch Statement It checks the result of the expression with multiple conditions (cases). The case is	<pre>switch (expression) { case value1: Statements</pre>	<pre><?php \$num=2; switch(\$num) {</pre>

executed for which the match is found.	<pre> break; case value2: Statements break;..... default: Code to be executed if all cases are not matched; } </pre>	<pre> 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 </pre>
--	--	--

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

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

***Note:** Students 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** [☐]**3: Need Improvement** [☐]**1: Incomplete** [☐]**4: Complete** [☐]**2: Late 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';
$str="$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.	<pre><?php \$input = 'Sunayana'; echo strlen(\$input); ?></pre> Output: 8
trim()	It used to remove the whitespaces and other characters.	<pre><?php \$input = " Programming in PHP \n"; echo trim(\$input); ?></pre> Output: Programming in PHP
ltrim()	It used to strip whitespace or other characters from the beginning of a string.	<pre><?php \$input = " Programming in PHP \n"; echo trim(\$input); ?></pre> Output: Programming in PHP \n
rtrim()	It is used to remove the	<?php

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

	string.	<pre>echo strrev("sairamkrishna"); ?></pre> Output: anhsirkmaris
str_pad()	It pads a string to a new length.	<pre><?php //Pad to the right side of the string, to a new length of 20 characters: \$str = "Hello World"; echo str_pad(\$str,15,"="); ?></pre> Output: Hello World=====
explode()	It is used to split a string by string	<pre><?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); ?></pre> 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.	<pre><?php \$arr = array('Hello','World!','Beautiful','Day!'); echo implode(" ", \$arr); echo "

"; \$arr = array('Hello','World!','Beautiful','Day!'); echo implode(" ", \$arr); ?></pre> 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.	<pre><?php echo strpos("I love php, I love php too!", "php"). "
"; ?></pre> Output: 7

strpos()	It is used to find the first occurrence of a string and returns from that small string onwards.	<pre><?php echostrpos("Hello world!","world")."

"; echostrpos("w3resource.com","."); ?></pre> 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("Mercedes",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	<pre><?php \$a=array("10","20","30","40"); print_r(array_chunk(\$a,2); ?></pre> 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.	<pre><? php \$X=array("a","b","c"); \$Y=array("100","200","300"); \$Z=array_combine(\$X,\$Y); print_r(\$Z); ?></pre> Output : Array ([a] => 100, [b] => 200, [c] => 300)
array_diff ():	It is used to compare the values of two arrays and return the difference	<pre><?php echo ""; \$a=array(1,2,3,4,5); \$b=array(4,2,6); \$c=array_diff(\$a,\$b); print_r(\$c); ?></pre> Output: Array ([0] => 1 [2] => 3 [4] => 5)
array_intersect()	It returns the common elements of two arrays.	<pre><?php \$a=array(1,2,3,4); \$b=array(4,5,6,2); \$c=array_intersect(\$a,\$b);</pre>

		<pre>print_r(\$c); ?></pre> Output: Array([1]=>2, [2]=>4);
array_flip()	Exchanges all keys with their associated values in an array.	<pre><?php \$a = array("a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5); print_r(array_flip(\$a)); ?></pre> Output: Array ([1] => a [2] => b [3] => c [4] => d [5] => e)
array_splice()	It removes and replaces specified elements of an array	<pre><?php \$a=array(10,20,30,40,50,60); \$b=array_splice(\$a,2,3); print_r(\$a); print_r(\$b); ?></pre> 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 parameters.	<pre><?php \$a=array(1,2,3,4,5,6); \$b=array_slice(\$a,2,3); print_r(\$a); echo "
"; print_r(\$b); ?></pre> 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.	<pre><?php \$a=array(1,2,3); \$d=array_reverse(\$a); print_r(\$d) ?></pre> Output: Array ([0] => 3 [1] => 2 [2] => 1)
array_key_exists()	This function is used to check if an element exists in the array	<pre><?php \$a=array("a"=>"ABC","p"=>"PQR","x"=>"XY Z"); if(array_key_exists("p",\$a)) echo "Key Exists"; else echo "Key Does not Exists"; ?></pre> Output: Key Exists
array_push()	This function add the	<pre><?php</pre>

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

		Output: Array ([a] => green [0] => red [1] => blue)
extract()	It creates local variables from an array.	<pre>?php \$a = "Original"; \$my_array = array("a" => "Cat","b" => "Dog", "b" => "Horse"); extract(\$my_array); echo "\\$a = \$a; \\$b = \$b; \\$c = \$c"; ?></pre> Output: \$a = Cat; \$b = Dog; \$c = Horse
compact()	Create array containing variables and their values	<pre><?php \$city = "Pune"; \$state = "Mumbai"; \$result = compact("city", "state"); print_r(\$result); ?></pre> Output: Array ([city] => Pune [state] => Mumbai)
in_array()	Checks if a specified value exists in an array	<pre><?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"; ?></pre> Output: Element Available in array
count()	It gives number of elements in an array.	<pre><?php \$a=array(10,20,30,40,50,60); echo count(\$a); ?></pre> Output: 6

Array Sorting Functions

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

		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.	<pre><?php \$a=array("mh","ap","LM","za"); print_r(\$a); echo "
"; rsort(\$a); echo "
 After Sorting
"; print_r(\$a); ?></pre> 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	<pre><?php \$b= array("X"=>"XYZ","A"=>"ABC","L"=>"LMN"); print_r(\$b); asort(\$b); echo "
 After Sorting
"; print_r(\$b); ?></pre> 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.	<pre><?php \$b= array("X"=>"XYZ","A"=>"ABC","L"=>"LMN"); print_r(\$b); arsort(\$b); echo "
 After Sorting
"; print_r(\$b); ?></pre> 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.	<pre><?php \$b= array("X"=>"XYZ","A"=>"ABC","L"=>"LMN"); print_r(\$b); ksort(\$b);</pre>

		<pre>echo "
 After Sorting
"; print_r(\$b); ?></pre> 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.	<pre><?php \$b= array("X"=>"XYZ", "A"=>"ABC", "L"=>"LMN"); print_r(\$b); krsort(\$b); echo "
 After Sorting
"; print_r(\$b); ?></pre> Output: Array ([X] => XYZ [A] => ABC [L] => LMN) After Sorting Array ([X] => XYZ [L] => LMN [A] => ABC)

***Note:** Students 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** []**3: Need Improvement** []**1: Incomplete** []**4: Complete** []**2: Late 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.	<pre><?php define ("PI",3.14); echo PI ?></pre> 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	<pre><?php echo "Today is " . date("Y/m/d") . "
"; echo "Today is " . date("Y.m.d") . "
"; echo "Today is " . date("l"); ?></pre> 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 called default 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 functions as a single unit.

A class can be defined as:

Example:

```
<?php
class Car
{
    /* Member variables */
    var $price;
    /* Member functions */
    function setPrice($par)
    {
        $this->price = $par;
    }
    function getPrice()
    {
        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 to 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 to 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 be 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="<?php echo $_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** [☐]**3: Need Improvement** [☐]**1: Incomplete** [☐]**4: Complete** [☐]**2: Late 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:

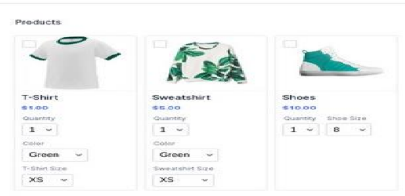
1. Write PHP program to store student information like Seat number, name, and class. On the second page, accept marks of the subject PHP, DS, CPP, and RDBMS. Display Result in table format on the third page (use cookies).
2. Write a PHP script to accept username and password. If in the first three chances, 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 the Patient 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(rno int, sname varchar(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 an insert/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 fetch data 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 details in 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.

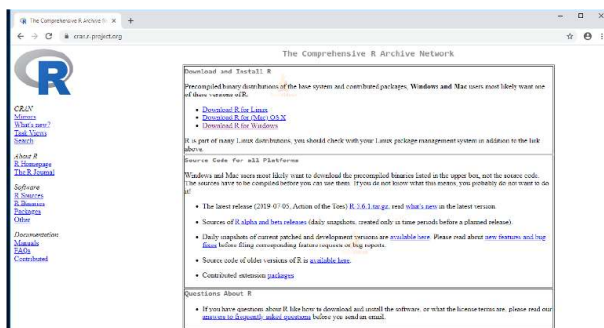
The core of R is an interpreted computer language which allows branching and looping as well as modular programming using functions. R allows integration with the procedures written in the C, C++, .Net, Python or FORTRAN languages for efficiency.

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](https://cran.r-project.org).



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

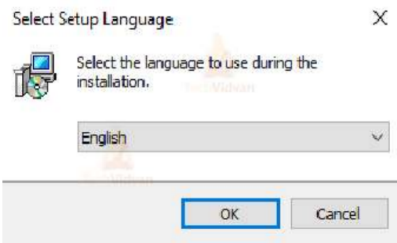
Step – 3: Click on the **base** subdirectory link or **install R for the first time** link.

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.

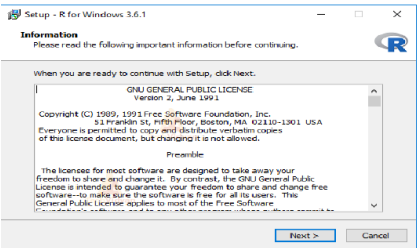


Step – 5: Run the .exe file and follow the installation instructions.

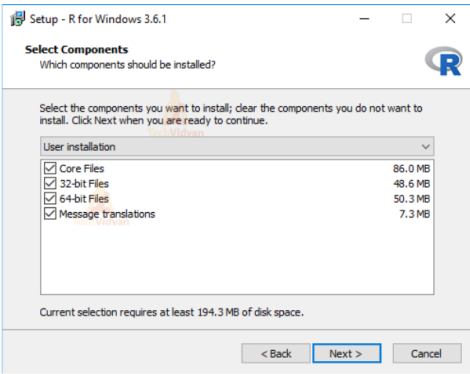
5.a. Select the desired language and then click **Next**.



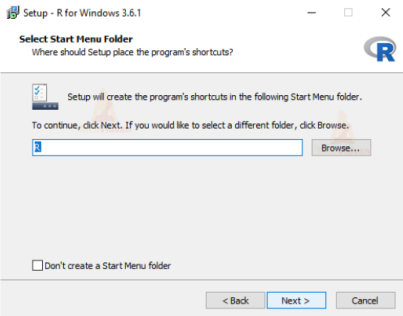
5.b. Read the license agreement and click **Next**.



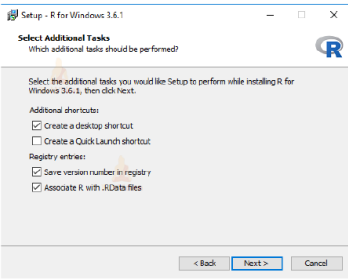
5.c. Select the components you wish to install (it is recommended to install all the components). Click **Next**.



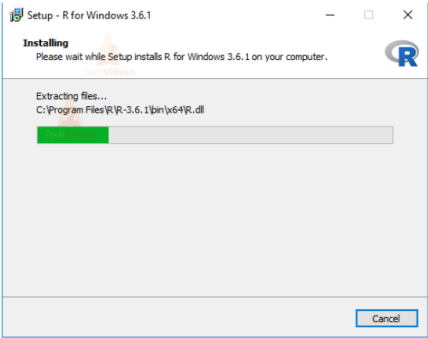
5.d. Enter/browse the folder/path you wish to install R into and then confirm by clicking **Next**.



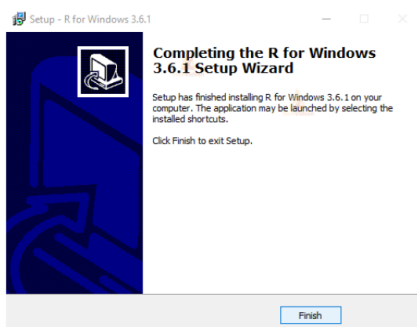
5.e. Select additional tasks like creating desktop shortcuts etc. then click **Next**.



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



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



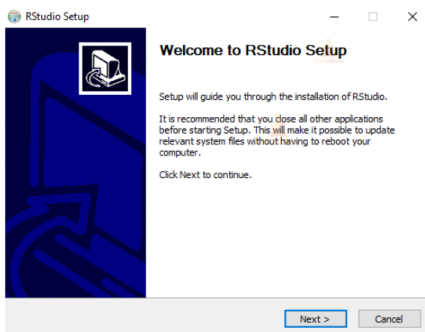
Install RStudio on Windows:

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**.

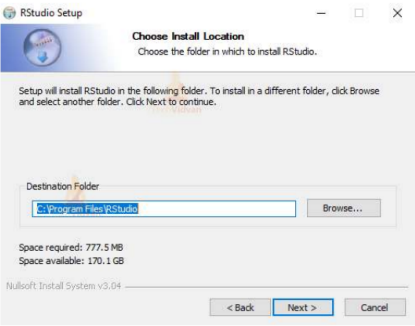
Step – 2: Click on the link for the windows version of RStudio and save the .exe file.

Step – 3: Run the .exe and follow the installation instructions.

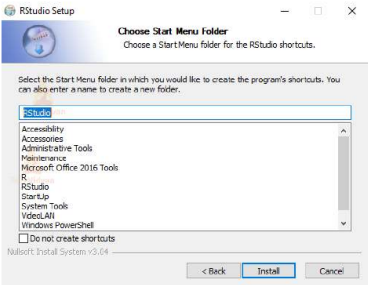
3.a. Click **Next** on the welcome window.



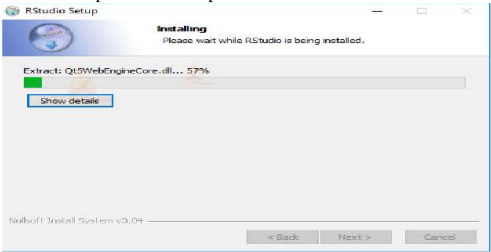
3.b. Enter/browse the path to the installation folder and click **Next** to proceed.



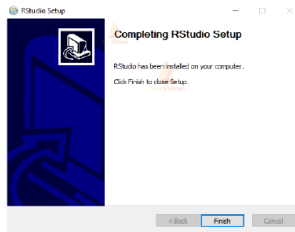
3.c. Select the folder for the start menu shortcut or click on do not create shortcuts and then click **Next**.



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 of as.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:

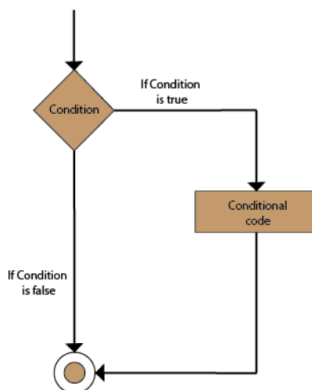
The if statement consists of the Boolean expressions followed by one or more statements. The if statement is the simplest decision-making statement which helps us to take a decision on the basis of the condition.

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.

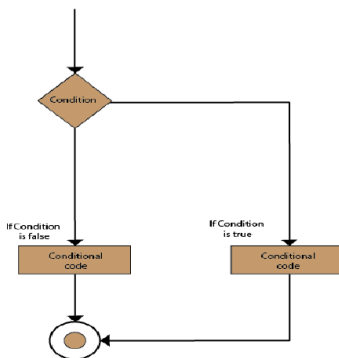
There is another type of decision-making statement known as the if-else statement. An if-else statement is the if statement followed by an else statement. An if-else statement, else statement will be executed when the boolean expression will false. In simple words, If a Boolean expression will have true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

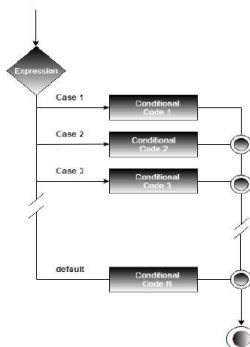
Switch Statement:

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....)
```

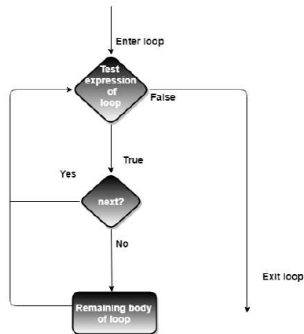
Flow Chart:

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
```

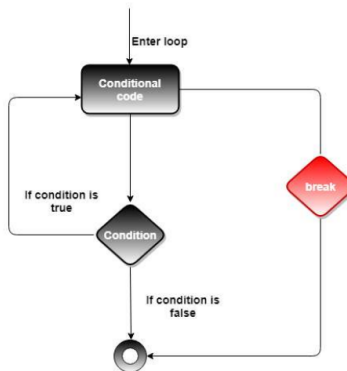
Break Statement:

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 or 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 substitute 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", "hl")))
```

7. Conversion to lower case

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

Example:

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

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

$$\text{Sum} = X + X^2/2! + X^3/3! + \dots$$

Assignment Evaluation**0: Not Done** []**3: Need Improvement** []**1: Incomplete** []**4: Complete** []**2: Late 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.integer(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 IndiaChina  UK   USA   Russia
> b=c(-3, -6)
> print(a[b])            #it prints IndiaChina  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 output

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 $Rollno 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 store 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 nth element of a given vector.
2. Write a R program to select second element of a given nested list.

Assignment Evaluation**0: Not Done** []**3: Need Improvement** []**1: Incomplete** []**4: Complete** []**2: Late 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 an 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 Column 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.

```

> a=c(10, 20, 30)
> b=c(40, 50, 60)
> m1=rbind(a,b)
> m1
      [,1] [,2] [,3]
a      10    20    30
b      40    50    60
> m2=cbind(b,a)
> m2
      b      a
[1,] 40    10
[2,] 50    20
[3,] 60    30

```

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, 2nd row and 3rd, 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 Column.  

```
>print(A[-2, -3])
```
  7. # Access All the element except 3rd and 4th Column.  

```
>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. # Adding 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 3x3 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** [   ]**3: Need Improvement** [   ]**1: Incomplete** [   ]**4: Complete** [   ]**2: Late 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 example

# 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 access elements of a vector, same way we access the elements of a factor. If gender is a factor then gender[i] would mean accessing 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 3<sup>rd</sup> and 5<sup>th</sup> rows with 1<sup>st</sup> and 3<sup>rd</sup> 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
2,Shubham,90
3,Rajesh,65
```

#student.csv file

### 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(displ), 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 mtcars 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 mtcars 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 weight 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.

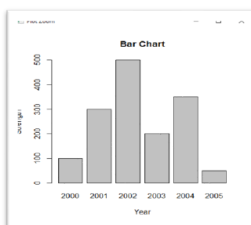
For this purpose, R provides the `barplot()` function, which has the following syntax:

**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))
```



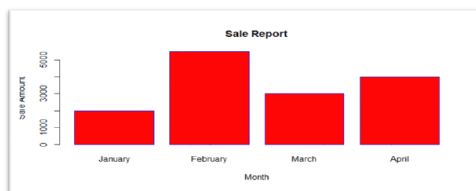
### Creating a barplot in R by reading data from CSV file:

```
>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           |
|------------|-------------|
| 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")
```



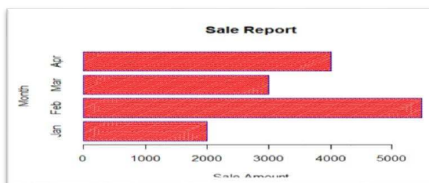
### Horizontal Bar Chart in R Programming:

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:

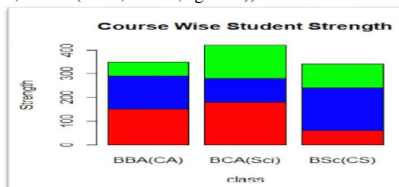
We can create bar charts with groups of bars and stacks using matrices as input values in each bar. One or more variables are represented as a matrix that is used to construct group bar charts and stacked bar charts.

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

| A1 |          | fx Class |     |     |  |
|----|----------|----------|-----|-----|--|
|    | 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.

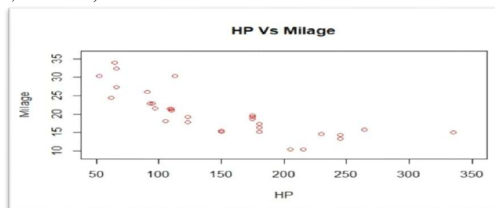
Scatterplot is created using plot() function. The syntax is as follows

**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:

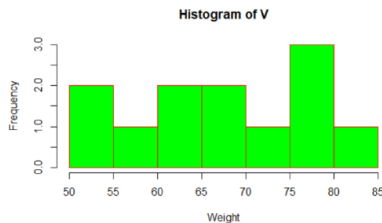
**Syntax:** hist(v, main, xlab, ylab, xlim, ylim, breaks, col, border) where

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,

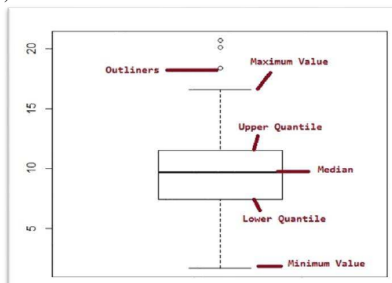
1. **data:** Dataframe, or List that contains the data to draw boxplot.
2. **Xlab:** It is used to describe the x-axis.
3. **Ylab:** It is used to describe the y-axis.
4. **Main:** It is used to give a title to the graph.
5. **Names:** It is the group of labels that will be printed under each boxplot.

### 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:**

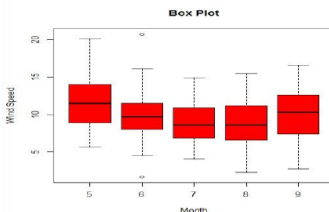
In this example, we create a Boxplot using the formula argument

**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 Temperature 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.

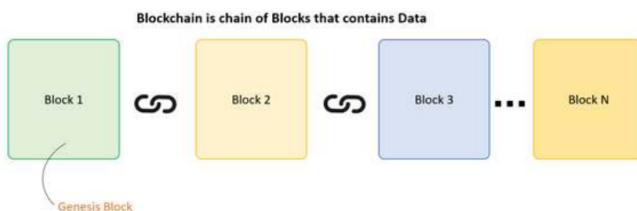
**A block, plus all the blocks before and after it, form a block chain.**

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 cryptography.

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
 {
 //Constructor
 //Hash function
 }
Step 3: Create a Blockchain 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.

```
constSHA256 = 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**

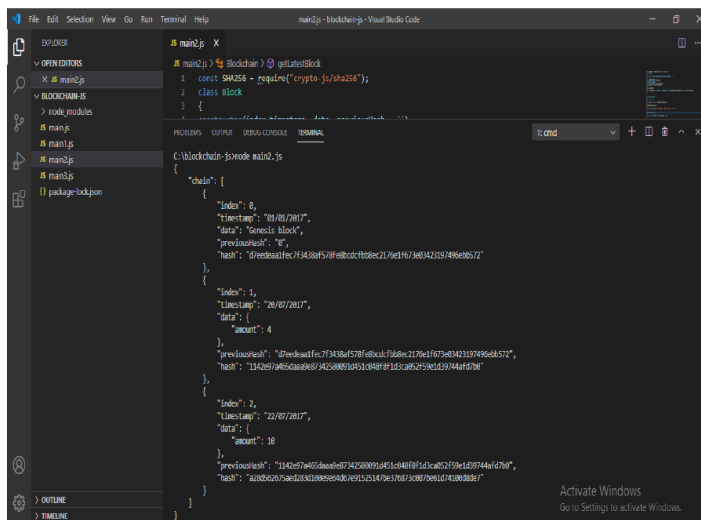
D:\Blockchain>npm install --save crypto-js

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:**



```
main2.js X
1 const SHA256 = require('crypto-js/sha256');
2 class Block
3 {
4 constructor(index, timestamp, data, previousHash) {
5 this.index = index;
6 this.timestamp = timestamp;
7 this.data = data;
8 this.previousHash = previousHash;
9 }
10 static createBlock(index, timestamp, data, previousHash) {
11 return new Block(index, timestamp, data, previousHash);
12 }
13 }
14 const block1 = Block.createBlock(0, '01/01/2017', 'Genesis block', '0');
15 const block2 = Block.createBlock(1, '20/07/2017', '4', block1.hash);
16 const block3 = Block.createBlock(2, '22/07/2017', '10', block2.hash);
17 console.log(block1);
18 console.log(block2);
19 console.log(block3);
```

```
{
 "chain": [
 {
 "index": 0,
 "timestamp": "01/01/2017",
 "data": "Genesis block",
 "previousHash": "0",
 "hash": "d7eed0a1fec7f483bf578feb0dcfbb8ec217aef167a0342333749e0b572"
 },
 {
 "index": 1,
 "timestamp": "20/07/2017",
 "data": {
 "amount": 4
 },
 "previousHash": "d7eed0a1fec7f483bf578feb0dcfbb8ec217aef167a0342333749e0b572",
 "hash": "1342657a0650a08087342c80810451048f8f1f53a052f9e1d39740f4d70f"
 },
 {
 "index": 2,
 "timestamp": "22/07/2017",
 "data": {
 "amount": 10
 },
 "previousHash": "1342657a0650a08087342c80810451048f8f1f53a052f9e1d39740f4d70f",
 "hash": "a20265c075a0c033c108e9e94067ef3525347be3703730870e01d7410030de7"
 }
]
}
```

**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();
 }
}

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(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()
 {
 return this.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();
 }
}

class Blockchain
{
 constructor()
 {
 this.chain = [this.createGenesisBlock()];
 }
}

```

```

createGenesisBlock()
{
 return new Block(0,"01/06/2020", "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);
} }
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 [ ]

3: Need Improvement [ ]

1: Incomplete [ ]

4: Complete [ ]

2: Late 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

Syntax:     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 mutability> returns (<return types>) {

```
// statement that write in
}
```

#### State mutability:

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.
- ✓ **recover(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.

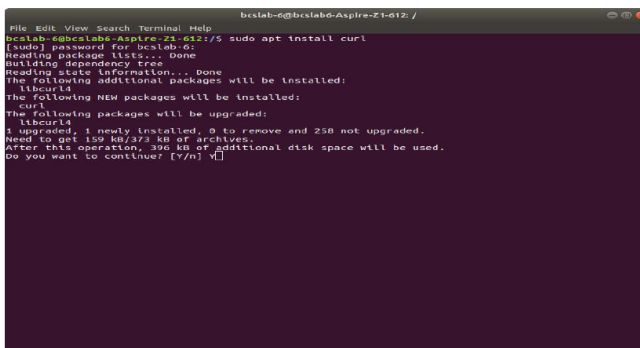
Fabric is the first distributed ledger platform to support smart contracts authored in general-purpose programming languages such as Java, Go and Node.js, rather than constrained domain-specific languages (DSL). This means that most enterprises already have the skill set needed to develop smart contracts, 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`



```

bcs1ab-0@bcs1ab0-Aspire-Z1-G12: /
bcs1ab-0@bcs1ab0-Aspire-Z1-G12:/$ sudo apt install curl
[sudo] password for bcs1ab-0:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
 libcurl4
The following NEW packages will be installed:
 curl
The following packages will be upgraded:
 libcurl4
1 upgraded, 1 newly installed, 0 to remove and 250 not upgraded.
Need to get 159 kB/372 kB of archives.
After this operation, 396 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y

```

**Step 2:** Installing Docker type on terminal : `sudo apt install docker`

```

bcs1ab-@bcs1ab-Aspire-Z1-612: /
File Edit View Search Terminal Help
bcs1ab-@bcs1ab-Aspire-Z1-612:/$ sudo apt install docker
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
 docker
0 upgraded, 1 newly installed, 0 to remove and 258 not upgraded.
Need to get 12.9 kB of archives.
After this operation, 45.1 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 docker amd64 1.5-1build1 [12.9 kB]
Fetched 12.9 kB in 0s (59.8 kB/s)
Selecting previously unselected package docker.
(Reading database ... 10951 files and directories currently installed.)
Preparing to unpack .../docker_1.5-1build1_amd64.deb ...
Unpacking docker (1.5-1build1) ...
Setting up docker (1.5-1build1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...

```

After installation you check the version type on terminal : **docker--version**

**Step 3: Installing Docker Compose type on terminal: sudo apt install docker-compose**

```

bcs1ab-@bcs1ab-Aspire-Z1-612: /
File Edit View Search Terminal Help
bcs1ab-@bcs1ab-Aspire-Z1-612:/$ sudo apt install docker-compose
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
 bridge-utils cgroupfs-mount containerd docker.io polang-docker-credential-helpers pigz
 python-ascicrypto python-backports.ssl-match-hostname python-cached-property python-certifi
 python-cffi-backend python-chardet python-cryptography python-docker python-dockerpty
 python-dockerpycreds python-docopt python-enum34 python-funcsigs python-functools32
 python-ldna python-lpaddress python-iso8583 python-mock python-openssl python-pbr
 python-pkg-resources python-requests python-six python-texttable python-urllib3
 python-websocket python-yaml runc ubuntu-fan
Suggested packages:
 aufs-tools btrfs-progs debootstrap docker-doc rinse zfs-fuse | zfsutils
 python-cryptography-doc python-cryptography-vectors python-enum34-doc python-funcsigs-doc
 python-mock-doc python-openssl-doc python-openssl-dbg python-setuputils python-socks
 python-ntlm
The following new packages will be installed:
 bridge-utils cgroupfs-mount containerd docker-compose docker.io
 polang-docker-credential-helpers pigz python-ascicrypto python-backports.ssl-match-hostname
 python-cached-property python-certifi python-cffi-backend python-chardet python-cryptography
 python-docker python-dockerpty python-dockerpycreds python-docopt python-enum34
 python-funcsigs python-functools32 python-ldna python-lpaddress python-iso8583 python-mock
 python-openssl python-pbr python-pkg-resources python-requests python-six python-texttable
 python-urllib3 python-websocket python-yaml runc ubuntu-fan
0 upgraded, 36 newly installed, 0 to remove and 258 not upgraded.
Need to get 65.7 MB of archives.
After this operation, 329 MB of additional disk space will be used.
Do you want to continue? [Y/n]

```

After installation you check the version type on terminal: **docker-compose --version**

**Step 4: Installing Go Language type on terminal : sudo apt install golang-go**

```

bcs1ab-6@bcs1ab6-Aspire-Z1-G12: /
File Edit View Search Terminal Help
bcs1ab-6@bcs1ab6-Aspire-Z1-G12: /$ sudo apt install golang-go
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
 golang-1.10-go golang-1.10-race-detector-runtime golang-1.10-src golang-race-detector-runtime
 golang-src pkg-config
Suggested packages:
 bzr mercurial subversion
The following NEW packages will be installed:
 golang-1.10-go golang-1.10-race-detector-runtime golang-1.10-src golang-go
 golang-race-detector-runtime golang-src pkg-config
0 upgraded, 7 newly installed, 0 to remove and 258 not upgraded.
Need to get 40.3 MB of archives.
After this operation, 255 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu blonic-updates/main amd64 golang-1.10-src amd64 1.10.4-2ubuntu1~18.04.1 [11.1 MB]
Get:2 http://in.archive.ubuntu.com/ubuntu blonic-updates/main amd64 golang-1.10-go amd64 1.10.4-2ubuntu1~18.04.1 [28.6 MB]
25% [2 golang-1.10-go 25.6 KB/28.6 MB 0%]

```

**Step 5: Installing nodejs type on terminal : `sudo apt install nodejs`**

```

bcs1ab-6@bcs1ab6-Aspire-Z1-G12: /
File Edit View Search Terminal Help
bcs1ab-6@bcs1ab6-Aspire-Z1-G12: /$ sudo apt install nodejs
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
 libc-ares2 libhttp-parser2.7-1 libuv1 nodejs-doc
The following NEW packages will be installed:
 libc-ares2 libhttp-parser2.7-1 libuv1 nodejs nodejs-doc
0 upgraded, 5 newly installed, 0 to remove and 258 not upgraded.
Need to get 2,076 kb of archives.
After this operation, 24.8 MB of additional disk space will be used.
Do you want to continue? [Y/n]

```

**Step 6: Installing npm type on terminal : `sudo apt install npm`**

```

bcslab-6@bcslab6-Aspire-Z1-G12: /
File Edit View Search Terminal Help
bcslab-6@bcslab6-Aspire-Z1-G12:/ $ sudo apt install npm
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
 gyp javascript-common libjs-async libjs-inherits libjs-jquery libjs-node-uuid
 libjs-underscore libssl1.0-dev libuv1-dev node-abbrev node-ansi node-ansi-color-table
 node-archy node-async node-balanced-match node-block-stream node-brace-expansion
 node-builtin-modules node-combined-stream node-concat-map node-cookie-jar node-delayed-stream
 node-forever-agent node-form-data node-fs.realpath node-fstream node-fstream-ignore
 node-github-url-from-git node-glob node-graceful-fs node-gyp node-hosted-git-info
 node-inflight node-inherits node-int node-is-builtin-module node-isexe
 node-json-stringify-safe node-lockfile node-lru-cache node-mime node-minimatch node-mkdirp
 node-mute-stream node-node-uuid node-nopt node-normalize-package-data node-npmlog node-once
 node-osenv node-path-is-absolute node-pseudomap node-qs node-read node-read-package-json
 node-request node-retry node-rimraf node-semver node-sha node-slide node-spx-correct
 node-spx-expression-parse node-spx-license-ids node-tar node-tunnel-agent node-underscore
 node-validate-npm-package-license node-which node-wrapappy node-yallist nodejs-dev
Suggested packages:
 node-hawk node-aws-sign node-oauth-sign node-http-signature debhelper
The following NEW packages will be installed:
 gyp javascript-common libjs-async libjs-inherits libjs-jquery libjs-node-uuid
 libjs-underscore libssl1.0-dev libuv1-dev node-abbrev node-ansi node-ansi-color-table
 node-archy node-async node-balanced-match node-block-stream node-brace-expansion
 node-builtin-modules node-combined-stream node-concat-map node-cookie-jar node-delayed-stream
 node-forever-agent node-form-data node-fs.realpath node-fstream node-fstream-ignore
 node-github-url-from-git node-glob node-graceful-fs node-gyp node-hosted-git-info
 node-inflight node-inherits node-int node-is-builtin-module node-isexe
 node-json-stringify-safe node-lockfile node-lru-cache node-mime node-minimatch node-mkdirp
 node-mute-stream node-node-uuid node-nopt node-normalize-package-data node-npmlog node-once
 node-osenv node-path-is-absolute node-pseudomap node-qs node-read node-read-package-json
 node-request node-retry node-rimraf node-semver node-sha node-slide node-spx-correct
 node-spx-expression-parse node-spx-license-ids node-tar node-tunnel-agent node-underscore
 node-validate-npm-package-license node-which node-wrapappy node-yallist nodejs-dev npm
0 upgraded, 73 newly installed, 0 to remove and 258 not upgraded.
Need to get 4,335 kB of archives.
After this operation, 24.2 MB of additional disk space will be used.
Do you want to continue? (Y/n)

```

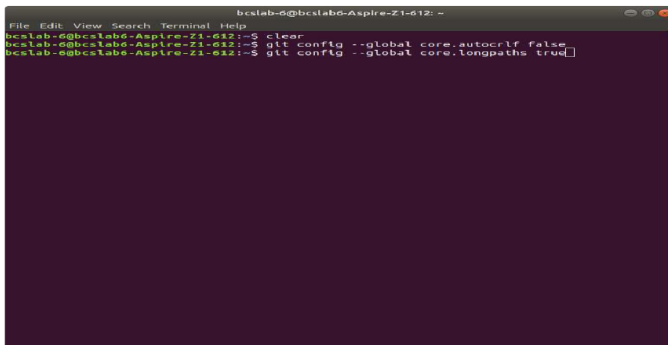
### Step 7: Installing python type on terminal: `sudo apt install python`

```

bcslab-6@bcslab6-Aspire-Z1-G12: /
File Edit View Search Terminal Help
bcslab-6@bcslab6-Aspire-Z1-G12:/ $ sudo apt install python
Reading package lists... Done
Building dependency tree
Reading state information... Done
python is already the newest version (2.7.13-r1-1).
python set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 258 not upgraded.
bcslab-6@bcslab6-Aspire-Z1-G12:/ $

```

### Step 8: After Installation these two command type on `gitconfig --global core.autocrlf false` `gitconfig --global core.longpaths true`



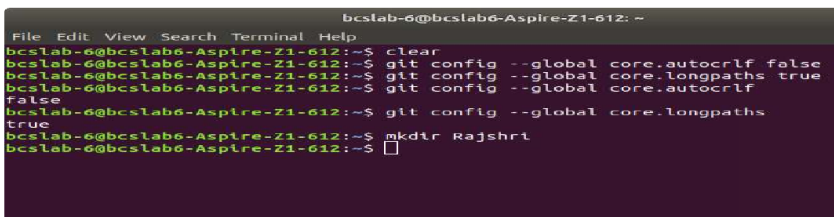
```

bcslab-6@bcslab6-Aspire-Z1-612: ~
File Edit View Search Terminal Help
bcslab-6@bcslab6-Aspire-Z1-612:~$ clear
bcslab-6@bcslab6-Aspire-Z1-612:~$ git config --global core.autocrlf false
bcslab-6@bcslab6-Aspire-Z1-612:~$ git config --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**



```

bcslab-6@bcslab6-Aspire-Z1-612: ~
File Edit View Search Terminal Help
bcslab-6@bcslab6-Aspire-Z1-612:~$ clear
bcslab-6@bcslab6-Aspire-Z1-612:~$ git config --global core.autocrlf false
bcslab-6@bcslab6-Aspire-Z1-612:~$ git config --global core.longpaths true
bcslab-6@bcslab6-Aspire-Z1-612:~$ git config --global core.autocrlf
false
bcslab-6@bcslab6-Aspire-Z1-612:~$ git config --global core.longpaths
true
bcslab-6@bcslab6-Aspire-Z1-612:~$ mkdir Rajshri
bcslab-6@bcslab6-Aspire-Z1-612:~$

```

**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 -sSL<https://goo.gl/6wtTN5> | bash -s 1.1.0 1.1.0 0.4.6**

Download for latest version files of fabric samples  
**curl -sSL<http://bit.ly/2ysbOFE> | bash -s**

```
File Edit View Search Terminal Help
bcslab-6@bcslab-Aspire-Z1-G12: ~/Rajshri

bcslab-6@bcslab-Aspire-Z1-G12:~/Rajshri$ curl -sSL http://bit.ly/2ysb0FE | bash -s
Clone hyperledger/fabric-samples repo

==== Cloning hyperledger/fabric-samples repo and checkout v2.1.1
Cloning into 'fabric-samples'...
remote: Enumerating objects: 5586, done.
remote: Total 5586 (delta 0), reused 0 (delta 0), pack.reused 5586
Receiving objects: 100% (5586/5586), 3.36 MiB | 2.72 MiB/s, done.
Resolving deltas: 100% (2883/2883), done.
error: pathspec 'v2.1.1' did not match any file(s) known to git.

Pull hyperledger fabric binaries

==== Downloading version 2.1.1 platform specific fabric binaries
==== Downloading: https://github.com/hyperledger/fabric/releases/download/v2.1.1/hyperledger-fabric-linux-and64-2.1.1.tar.gz
% Total % Received % Xferd Average Speed Time Time Time Current
 Dload Upload Total Spent Left Speed
100 654 100 654 0 0 1149 0 0:01:40 0:01:40 0:00:00 1147
2 72.7M 2 1682k 0 0 25584 0 0:01:07 0:01:33 0:00:00 26560
```

**Step 12:** Go to your downloaded directory. Type on terminal  
**cd fabric-samples/first-network**

```
root@DESKTOP-1DFAG01: ~/Rajshri/fabric-samples/first-network

root@DESKTOP-1DFAG01:~/Rajshri# cd fabric-samples/first-network
root@DESKTOP-1DFAG01:~/Rajshri/fabric-samples/first-network#
```

**Step 13:** check the files on that path Type on terminal: **ls -l**

```

root@DESKTOP-1DFA601: ~/Rajshri/fabric-samples/first-network
root@DESKTOP-1DFA601:~/Rajshri/fabric-samples/first-network# ls -l
total 72
-rw-rw-rw- 1 root root 335 Jul 10 19:31 README.md
drwxrwxrwx 1 root root 512 Jul 10 19:31 bin
-rwxrwxrwx 1 root root 20114 Jul 10 19:31 byfn.sh
drwxrwxrwx 1 root root 512 Jul 10 19:31 crypto-config
-rw-rw-rw- 1 root root 7912 Jul 10 19:31 configtx.yaml
-rw-rw-rw- 1 root root 3996 Jul 10 19:31 crypto-config.yaml
-rw-rw-rw- 1 root root 2971 Jul 10 19:31 docker-compose-cli.yaml
-rw-rw-rw- 1 root root 2345 Jul 10 19:31 docker-compose-couch.yaml
-rw-rw-rw- 1 root root 4560 Jul 10 19:31 docker-compose-couch.yaml
-rw-rw-rw- 1 root root 2883 Jul 10 19:31 docker-compose-eze-template.yaml
-rw-rw-rw- 1 root root 3493 Jul 10 19:31 docker-compose-org3.yaml
-rwxrwxrwx 1 root root 10627 Jul 10 19:31 byfn.sh
drwxrwxrwx 1 root root 512 Jul 10 19:31 crypto-config
drwxrwxrwx 1 root root 512 Jul 10 19:31 crypto-config
root@DESKTOP-1DFA601:~/Rajshri/fabric-samples/first-network#

```

**Step 14:** Then generate a network using to run file type on terminal  
`./byfn.sh generate`

```

root@DESKTOP-1DFA601:~/Rajshri/fabric-samples/first-network ./byfn.sh generate
Generating certs and genesis block for with channel 'mychannel' and CLI timeout of '10' seconds and CLI delay of '3' seconds
Continue? [Y/n] Y
Proceeding ...
/root/Rajshri/fabric-samples/first-network/./bin/cryptogen
Generating certificates using cryptogen tool
crypto-gen generate --config=./crypto-config.yaml
org1.example.com
org2.example.com
org3.example.com
+ set -x

/root/Rajshri/fabric-samples/first-network/./bin/configtxgen
Generating Orderer Genesis Block
configtxgen profile TwoOrgsOrdererGenesis outputblock ./channel-artifacts/genesis.block
2020-07-10 19:38:20.420 IST [common/tools/configtxgen] main -> INFO 001 Loading configuration
2020-07-10 19:38:20.482 IST [msp] getMapConfig -> INFO 002 Loading NodeOUs
2020-07-10 19:38:20.586 IST [msp] getMapConfig -> INFO 003 Loading NodeOUs
2020-07-10 19:38:20.586 IST [common/tools/configtxgen] doOutputBlock -> INFO 004 Generating genesis block
2020-07-10 19:38:20.586 IST [common/tools/configtxgen] doOutputBlock -> INFO 005 Writing genesis block
+ set -x

Generating channel configuration transaction 'channel.tx'
configtxgen profile TwoOrgsChannel outputCreateChannelTx ./channel-artifacts/channel.tx -channelID mychannel
2020-07-10 19:38:20.567 IST [common/tools/configtxgen] main -> INFO 006 Loading configuration
2020-07-10 19:38:20.581 IST [common/tools/configtxgen] doOutputChannelCreateTx -> INFO 007 Generating new channel c
nfigtx
2020-07-10 19:38:20.582 IST [msp] getMapConfig -> INFO 008 Loading NodeOUs

```

**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 SmartContract1.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\SmartContract1.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\SmartContract1.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\_SmartContract1.js**

```
=====
```

```
Replacing 'SmartContract1'

>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 variables (local, state, global ).
2. Write a solidity program to perform setter and getter operations on State variable.
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** [ ☐ ]**3: Need Improvement** [ ☐ ]**1: Incomplete** [ ☐ ]**4: Complete** [ ☐ ]**2: Late Complete** [ ☐ ]**5: Well Done** [ ☐ ]**Signature of Instructor**