Section-I

Data Structure

Assignment No 1: Array (One Dimensional Array)

ARRAY

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

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

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

  Homogeneous: - All the elements are of same type.

WHAT IS POYNOMIAL

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

EXAMPLE:

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

Practice Program:

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

SET A:

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

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

SET B:

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

SET C:

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

Assignment Evaluation

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

Signature of Instructor

Assignment No 2: Sorting Techniques (Non Recursive)

SORTING

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

  'Or'

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

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

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

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

1) Internal sorting-

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

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

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

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

Algorithm for Bubble sort:

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

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

Algorithm for Insertion Sort:

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

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

Algorithm for Selection Sort

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

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

Practice Programs:

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

SET A:

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

SET B:

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

SET C:

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

Assignment Evaluation

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

Signature of Instructor

Assignment No 3: Sorting Techniques (Recursive)

QUICK SORT

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

Procedure –

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

Algorithm:

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

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

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

Algorithm for partitioning

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

Step 9: return up
Step 10: stop.

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

MERGE SORT.

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

Algorithm:-

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

Practice Programs:

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

SET A:

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

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

SET B:

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

SET C:

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

Assignment Evaluation

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

Signature of Instructor

Assignment No 4: Searching Techniques

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

LINEAR SEARCH

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

Algorithm for Linear Search:

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

BINARY SEARCH

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

Algorithm for Binary search:

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

Practice Programs:

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

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

SET A:

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

SET B:

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

SET C:

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

Assignment Evaluation

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

Signature of Instructor

Assignment No 5: Linked List

- **Linked list:-**

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

- **LINKED LIST IMPLEMENTATION-:**

A linked list may be implemented in two ways:

1) Static representation
2) Dynamic representation.

1) Static representation:-

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

Data array

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

Link array

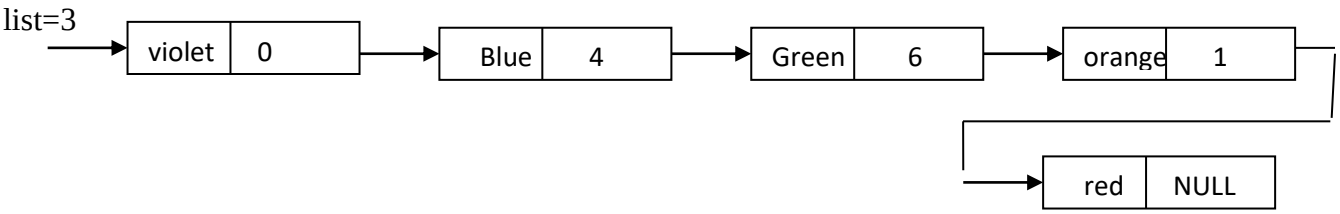
0	4
1	-1
2	
3	0
4	6
5	
6	1

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

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

2) **Dynamic Representation:-**

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



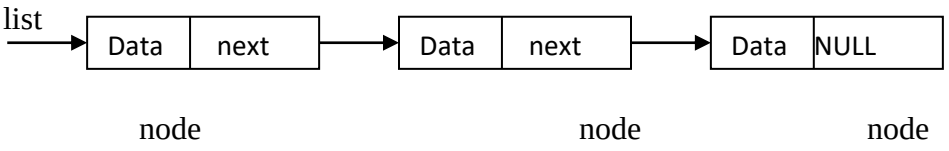
List is an external pointer which stores the address of the first node of the list.

- **TYPES OF LINKED LIST**

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

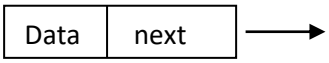
1) Linear linked list

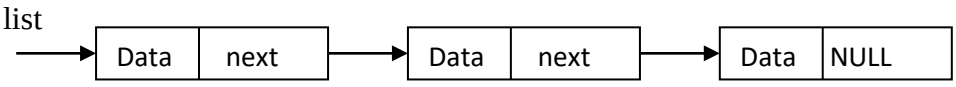
In this list the elements are organized in a linear fashion and list terminates at some point i.e. the last node contains a NULL pointer.



- **Singly linked list-**

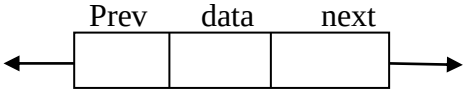
Each node in this list contains only one pointer which points to the next node.

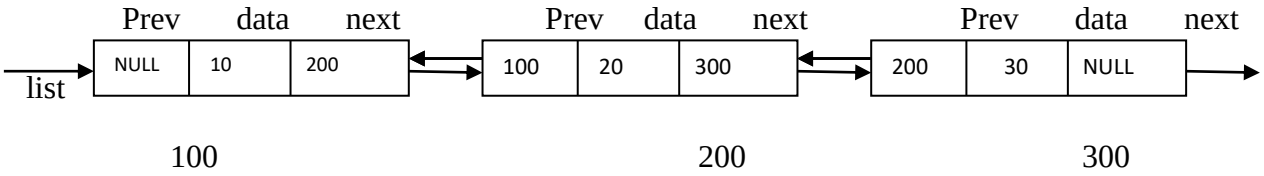




- **Doubly linked list**

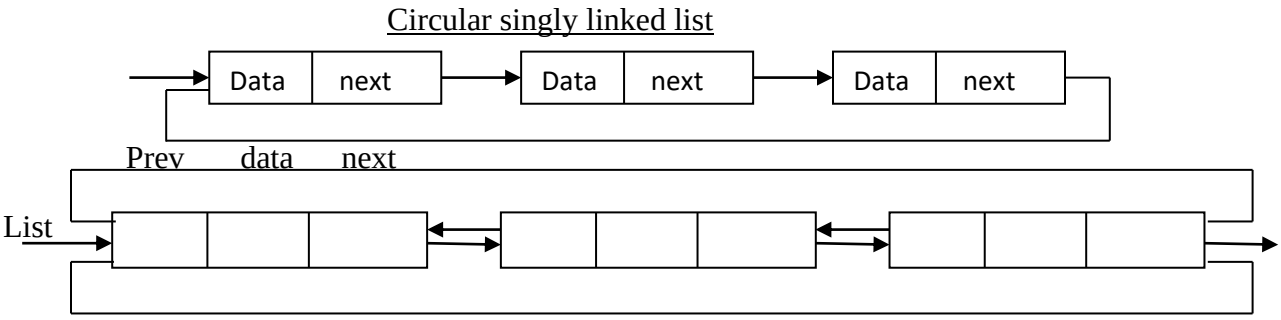
Each node in this contains two pointers, one pointing to the previous node and the other pointing to the next node. This list is used when traversing in both directions is required.





2) Circular list

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



Circular doubly linked list

- **OPERATIONS ON A LIST**

The following are some of the basic list operations:-

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

Practice Programs:

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

SET A:

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

SET B:

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

SET C:

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

Assignment Evaluation

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

Signature of Instructor

Assignment No 6: Stack

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

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

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

- **PRIMITIVE OPERATIONS ON A STACK.**

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

- **STACK IMPLEMENTATION:-**

The stack implementation can be done in two ways:-

1) Static implementation:-

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

1) Declaring of a stack :-

A stack to be implemented using an array will require.

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

We can use a structure for the above purpose.

2) Creating a stack:-

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

STACK s1;

3) initialize a stack:-

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

4) Checking whether stack is empty:-

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

5) Checking whether stack is full:-

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

6) The push operation:-