**Date-**

**Assignment No. :**

**Problem Statement:**

Program in C to implement stack using array.

**Theory:**

Stack is an abstract data type with a bounded(predefined) capacity. It is a simple data structure that allows adding and removing elements in a particular order. Every time an element is added, it goes on the top of the stack and the only element that can be removed is the element that is at the top of the stack, just like a pile of objects.

## Basic features of Stack

1. Stack is an ordered list of similar data type.
2. Stack is a LIFO(Last in First out) structure or we can say FILO(First in Last out).
3. push() function is used to insert new elements into the Stack and pop() function is used to remove an element from the stack. Both insertion and removal are allowed at only one end of Stack called Top.
4. Stack is said to be in Overflow state when it is completely full and is said to be in Underflow state if it is completely empty.

## Applications of Stack

The simplest application of a stack is to reverse a word. You push a given word to stack - letter by letter - and then pop letters from the stack.

There are other uses also like:

1. Parsing
2. Expression Conversion(Infix to Postfix, Postfix to Prefix etc)

**Algorithm:**

**Input specification:** An array (Sorted for linear/Binary search or Unsorted for linear search) say **a[]**, where the search will be done,

The element which need to be searched, say **find** and the number of elements in the array a[], say **n**.

**Output specification:** Success message of the search with the position of the element or appropriate failure message.

**Steps:**

Algorithm for method linear\_search(a[], n, find):

1. For (c = 0 to n-1 )
2. If (a[c] == find) Then
3. Print search" is present at location " c+1
4. Exit
5. End If
6. Set c=c+1
7. End For
8. Print search" is not present in array."

Algorithm for method binary\_search(a[], n, find):

1. Set first=0
2. Set last = n-1
3. Set middle = (first+last)/2
4. While (first <= last) Then
5. If (a[middle] < find) Then
6. Set first = middle + 1
7. Else If (a[middle] == find) Then
8. Print search" is present at location "middle+1
9. Else
10. Set last = middle - 1
11. Set middle = (first + last)/2
12. If (first > last) Then
13. Print search" is not present in array."

**Source Code:**

#include<stdio.h>

#include<conio.h>

void ins(void);

void del(void);

void disp(void);

int a[30],s,top=-1;

int main()

{

int ch;

printf("Enter the size of the stack: ");

scanf("%d",&s);

while(1)

{

printf("\n1.Insertion\n2.Deletion\n3.Traverse\n4.Exit\nEnter your choice: ");

scanf("%d",&ch);

switch(ch)

{

case 1: ins();

break;

case 2: del();

break;

case 3: disp();

break;

case 4: return 0;

default: printf("Wrong choice");

}

}

return 0;

}

void ins()

{

if(top==s-1)

printf("Stack overflow");

else{

printf("Enter the element to insert: ");

scanf("%d",&a[++top]);

}

}

void del()

{

if(top<0)

printf("Stack underflow");

else{

printf("The deleted element is %d",a[top]);

top--;

}

}

void disp()

{

int i;

if(top<0)

printf("Stack underflow");

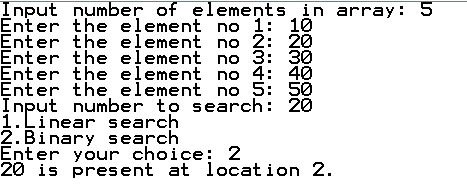
for(i=top;i>=0;i--)

printf("%d\n",a[i]);

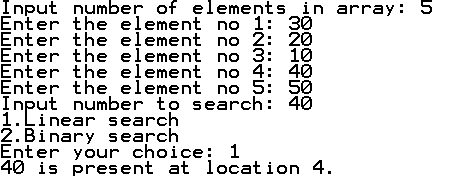
}

**Input & Output:**

Set 1:



Set 2:



**Discussion:**

1. Linear search is rarely used practically because other search algorithms such as the binary search algorithm and hash tables allow significantly faster searching comparison to linear search.
2. A linear search scans one item at a time, without jumping to any item .
   1. The worst case complexity is  O(n), sometimes known an O(n) search.
   2. Time taken to search elements keep increasing as the number of elements are increased.
3. A binary search however, cut down your search to half as soon as you find middle of a sorted list.
   1. The middle element is looked to check if it is greater than or less than the value to be searched.
   2. Accordingly, search is done to either half of the given list.
4. Input data needs to be sorted in Binary Search and not in Linear Search
5. Linear search does the sequential access whereas Binary search access data randomly.
6. Time complexity of linear search -O(n) , Binary search has time complexity O(log n).
7. Linear search performs equality comparisons and Binary search performs ordering comparisons.