Practical No.

Date :-

Aim :- Write a C Program to Insert, Delete & Show an element in the stack.
(Perform Push & Pop operations)

Objective :- To study concept of Lineat Data structure i.e(stack).

Practical No.

Date :-

Aim :- Write a C Program to Insert, Delete & Show an element in stack.
(Perform Push & Pop Operations.

Objective :- To study concept of Linear Data structure i.e (stack).

Theory :- Linear Data Stuctures are data structures in which elements are arranged in a linear or sequential manner.

A stack is a fundamental linear data structure that follows the principle of last in first out (LIFO), meaning that the last element added is the first one to be removed.

There are various operations that are performed on stack i.e

i) Push (Insertion of an element)
ii) Pop (Deletion of an element)
iii) Top (Inspecting the element at Peek of the stack without removing it.
iv) Underflow (Is_Empty conduction)
v) Overflow (Is_full conduction)
vi) Size (Determining the number of elements in the stack.

v) Overflow (Is_full condition)

vi) Size (Determining the number of elements in the stack.

Program code :-

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 20
int queue_array[MAX];
int rear = -1;
int front = -1;
void insert()
{
 int add_item;
 if (rear == MAX-1)
   {
    printf("\n Overflow");
   }
 else
 {
  if (front == -1)
    {
     front = 0;
     printf("Enter element
      to be inserted\n");
     scanf("%d", &
      add_item);
     printf("%d is the element
     inserted in the queue",
     add_item);
     printf("----------");
     rear = rear + 1;
     queue_array[rear] =
     add_item;
    }
 }
}
void delete()
{
 if (front == -1 ||
     front > rear)
 {
  printf("\n Underflow");
  return;
 }
 else
 {
  printf("\n Element
  deleted from the Queue
  is %d\n", queue_array[front]);
```

Program Code :-

```
#include<stdio.h>
#include<stdlib.h>
#define SIZE 4
int top = -1 ;
int inp_array [SIZE];

void push();
void pop();
void show();

void main()
{
  while (1)
  {
  printf("\n Perform opera
  -tions on stack:");
  printf("\n1. Push the
  element\n 2. Pop the
  element \n 3. Show
  \n 4. End");
  printf ("\n\n Enter your
  Choice");
  scanf("%d", &choice);
```

```
switch (choice)
{
 case 1:
        push();
        break;
 case 2:
        pop();
        break;
case 3:
        show();
        break ;
case 4:
        exit (0);
default :
        printf("\n Invalid
          choice !!");
    }
}
```

```
void push ()
{
  int x;
  if (top == SIZE-1)
  {
    printf("\nOverflow");
  }
  else
  {
    printf("\nEnter the ele
    -ment to be added onto
    the stack:");
    scanf("%d",&x);
    top= top + 1;
    inp_array [top]=x;
  }
}
Void pop()
{
  if (top == -1)
  {
    printf ("Underflow\n");
  }
  else
  {
    printf("\nPopped Element :
      %d", inp_array [top]);
    top= top -1;
  }
}
void show ()
{
  if (top == -1)
  {
    printf ("\nUnderflow!");
  }
  else
  {
    printf("\nElements present
    in the stack are:\n");
    for(int i=top; i>=0;
    i--)
    {
      printf("%d\n", inp-
        array [i]);
    }
  }
```

Output :- Perform operations on stack :

1. Push the element
2. Pop the element
3. Show
4. EXIT

Enter the choice : 1

Enter the element to be added onto the stack:

11

:

Algorithm:- // PUSH OPERATION

```
procedure push (value):
 if top equals MAX-1;
 print "overflow"
 return
 top <- top + 1
 items [top] <- value
 print value "Pushed to
 stack"
```

// POP OPERATION

```
function pop () :
 if top equals -1;
 print "underflow"
 return -1;
 poppedItem <- item [top]
 top <- top - 1
 print PoppedItem, "Popped from
 stack"
 return PoppedItem.
```

// DISPLAY

```
procedure displayStack () :
  if is_Empty_ ();
   print "stack is empty"
   return
  print "Elements in the
        stack"
  for i item from top to
  0 step -1 : print items [i]
```

Conclusion :- Hence according to given theory, the code & it's corresponding output are valid.

Conclusion:- Hence according to given theory, the code & it's corresponding output are valid.