**Experiment Number: 1**

**Title:** Implementation of Stack Using C

**Problem Statement**:

Write a menu driven program in C to perform following operations on the stack.

1.Push

2.Pop

**Algorithm:**

1. Steps to create an empty Stack-

Step 1 **-** Include all the **header files** which are used in the program and define a constant **'SIZE'** with specific value.

Step 2 **-** Declare all the **functions** used in stack implementation.

Step 3 - Create a one dimensional array with fixed size (int stack[SIZE])

Step 4 - Define a integer variable 'top' and initialize with '-1'. (int top = -1)

Step 5 - In main method, display menu with list of operations and make suitable function calls to perform operation selected by the user on the stack.push(value) - Inserting value into the stack

2. In a stack, push() is a function used to insert an element into the stack. In a stack, the new element is always inserted at top position. Push function takes one integer value as parameter and inserts that value into the stack.

We can use the following steps to push an element on to the stack-

Step 1 - Check whether stack is FULL. (top == SIZE-1)

Step 2 - If it is FULL, then display "Stack is FULL!!! Insertion is not possible!!!" and terminate the function.

Step 3 - If it is NOT FULL, then increment top value by one (top++) and set stack[top] to value (stack[top] = value). pop() - Delete a value from the Stack

3. In a stack, pop() is a function used to delete an element from the stack. In a stack, the element is always deleted from top position. Pop funct ion does not take any value as parameter.

We can use the following steps to pop an element from the stack-

Step 1 - Check whether stack is EMPTY. (top == -1)

Step 2 - If it is EMPTY, then display "Stack is EMPTY!!! Deletion is not possible!!!" and terminate the function.

Step 3 - If it is NOT EMPTY, then delete stack[top] and decrement top value by one (top--).

display() - Displays the elements of a Stack

4. We can use the following steps to display the elements of a stack-

Step 1 - Check whether stack is EMPTY. (top == -1)

Step 2 - If it is EMPTY, then display "Stack is EMPTY!!!" and terminate the function.

Step 3 - If it is NOT EMPTY, then define a variable 'i' and initialize with top. Display stack[i] value and decrement i value by one (i--).

Step 3 - Repeat above step until i value becomes '0'. Code

#include<stdio.h>

#define MAX 5

int stack[MAX];

int top=-1;

void push();

void pop();

void display();

int main()

{

int ch;

do{

printf("\n\*\*\*Stack Operations\*\*\*");

printf("\n\n1. Push\t2. Pop\t3. Display\t4.Exit");

printf("\nEnter Your Choice:");

scanf("%d",&ch);

switch(ch)

{

case 1: push();

break;

case 2: pop();

break;

case 3: display();

break;

case 4: printf("\nBreak point");

break;

default: printf("\nWrong Choice!! Please enter the right choice 1\2\3\4");

break;

}

}while(ch!=4);

}

void push()

{

int val;

if(top==MAX-1)

printf("\nStack is Full !! Overflow!!");

else

{

printf("\nEnter the element to be pushed:");

scanf("%d",&val);

top++;

stack[top]=val;

}

}

void pop()

{

if(top==-1)

printf("\nStack is empty!! Underflow!!");

else

{

printf("\nThe popped element is %d\n",stack[top]);

top--;

}

}

void display()

{

int i;

if(top==-1)

printf("\nStack is Empty!! Underflow!!");

else

{

printf("\nThe Stack Status:");

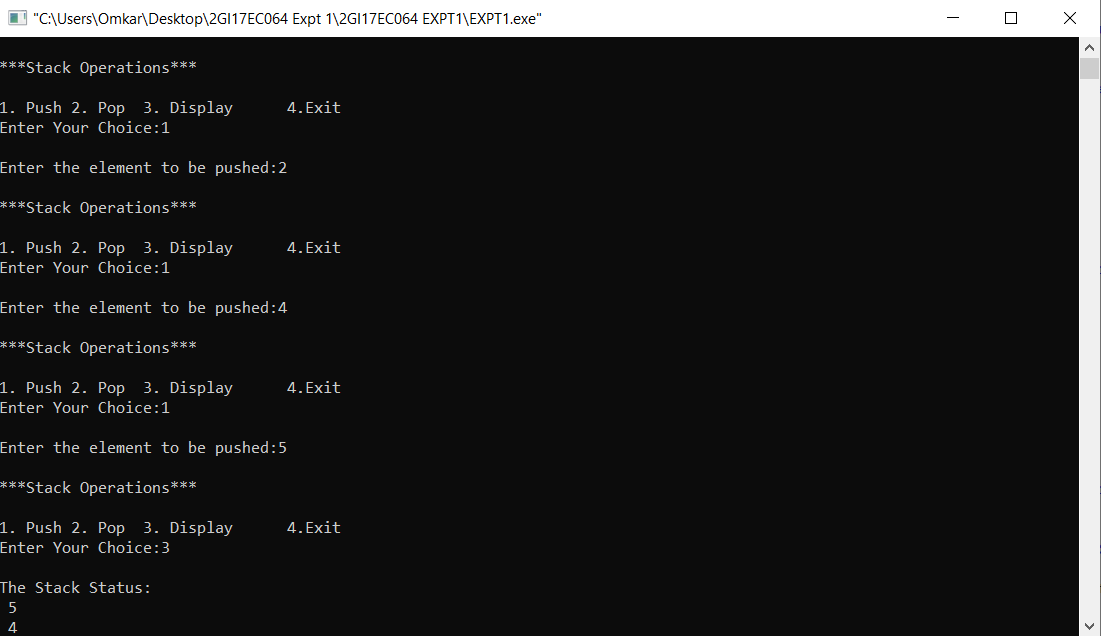
for(i=top;i>=0;i--)

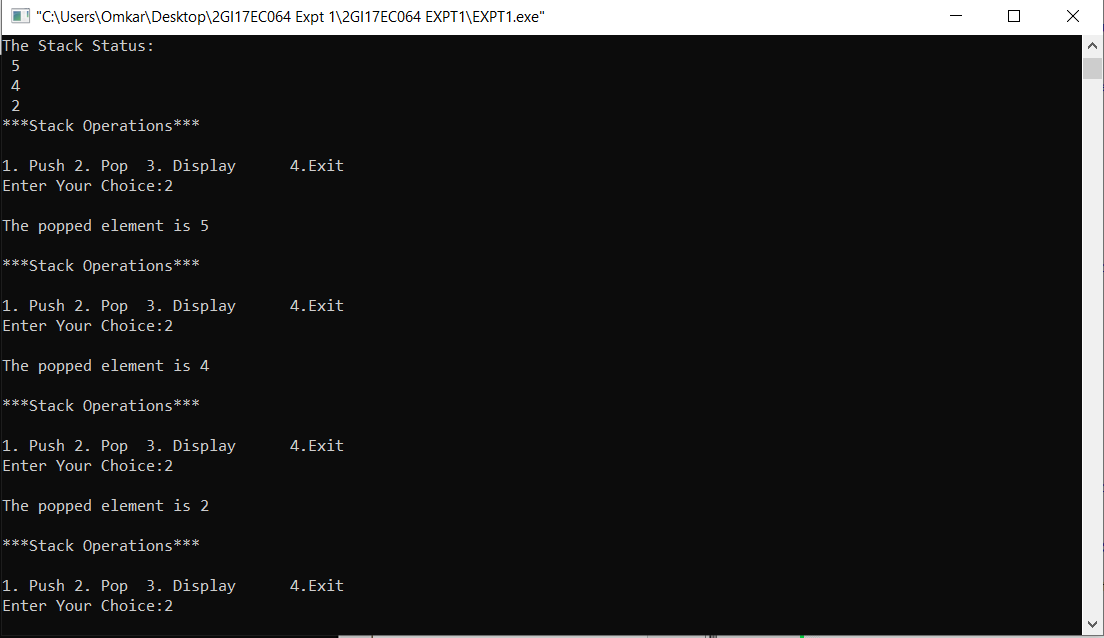
printf("\n %d",stack[i]);

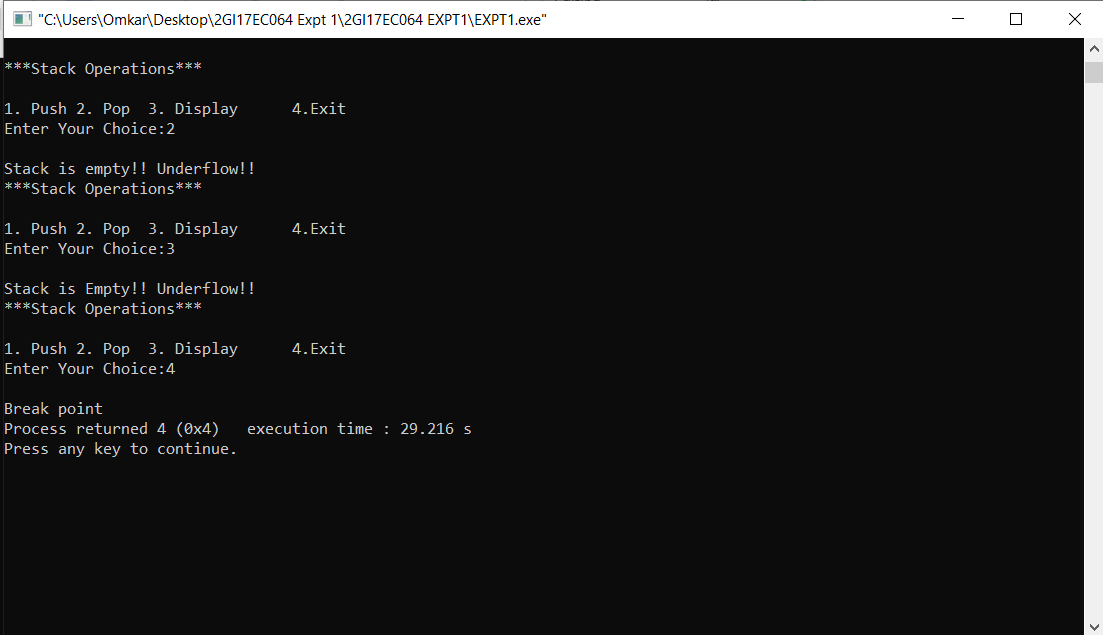
}

}

**Output:**







**ANALYSIS(LIMITATIONS):**

1. While using array to implement stack is easy, it has limitation that the stack cannot grow beyond a fixed size (of array).

2. The time complexity increase in insertion and depletion operation. - Wastage of memory because arrays are fixed in size. - If there is enough space present in the memory but not in continuous form, in this case we cannot initialise the array.