**ASSIGNMENT – 17**

**1.PROBLEM STATEMENT**

Write a program in C to implement stack using array.

**2.ALGORITHMS**

Algorithm **Push\_Stack**

**Input:** The pointer to the stack ‘arr’, ‘top’ pointer,size of the stack ‘max’ and the value ‘item’ to be pushed in the stack.

**Output:** The element ‘item’ pushed into the stack at appropriate location.

**Remarks:** ‘top’ indicates the position of the topmost element in the stack and must be passed as a pointer.

**Steps:**

1. **If**(top==max) **then** // if the stack is full
2. **Print** “Stack Overflow”
3. **Exit** //terminate the program
4. **Else**
5. top=top+1 //increment the value of top by one
6. arr[top]=item //insert item at the position of top in stack
7. **EndIf**
8. **Return**
9. **Stop**

Algorithm **Pop\_Stack**

**Input:** The pointer to the stack ‘arr’, ‘top’ pointer.

**Output:** The item at ‘top’ removed from the scope of the stack.

**Remarks:** In array representation of stack, popped elements are not physically erased from the memory.

**Steps:**

1. **If**(top==0) then //if stack is empty
2. **Print** “Stack Underflow
3. **Exit** //terminate the program
4. **Else**
5. top=top-1 //decrement top by one
6. **EndIf**
7. **Stop**

**3.SOURCE CODE**

#include<stdio.h>

#include<stdlib.h>

//function to display stack

void display(int \*arr, int top)

{

int i;

if(top==-1)

{

printf("Stack Is Empty\n");

return;

}

printf("STACK: ");

for(i=0;i<=top;i++)

printf("%d ",arr[i]);

}

//function to push an item into the stack

void push(int \*arr,int\* top,int ele)

{

(\*top)++; // increment top by one

arr[\*top]=ele; // insert the item at the position of top in the stack

printf("PUSHED:%d\n",ele);

display(arr,\*top); //display stack

}

//function to pop an item from the stack

void pop(int\* arr,int \*top)

{

printf("POPPED: %d\n",arr[\*top]);

(\*top)--; //decrement top by one

display(arr,\*top); //display stack

}

int main(void)

{

int max,\*arr,top,ch,ele,i;

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

scanf("%d",&max);

arr=(int\*)calloc(max,sizeof(int)); //allocating stack in heap

top=-1;

while(1)

{

printf("\n1.PUSH\n2.POP\n3.DISPLAY\n4.Exit");

printf("\nEnter your choice: ");

scanf("%d",&ch);

switch(ch)

{

case 1:

if(top==max-1) //if stack is full

{

printf("Stack Overflow!");

return 0;

}

else

{

printf("Enter The Element To Be Pushed: ");

scanf("%d",&ele);

push(arr,&top,ele);

}

break;

case 2:

if(top==-1) //if stack is empty

{

printf("Stack Underflow");

return 0;

}

else

pop(arr,&top);

break;

case 3:

display(arr,top);

break;

case 4:

exit(0);

default:

printf("INVALID CHOICE\nPlease Try Again\n");

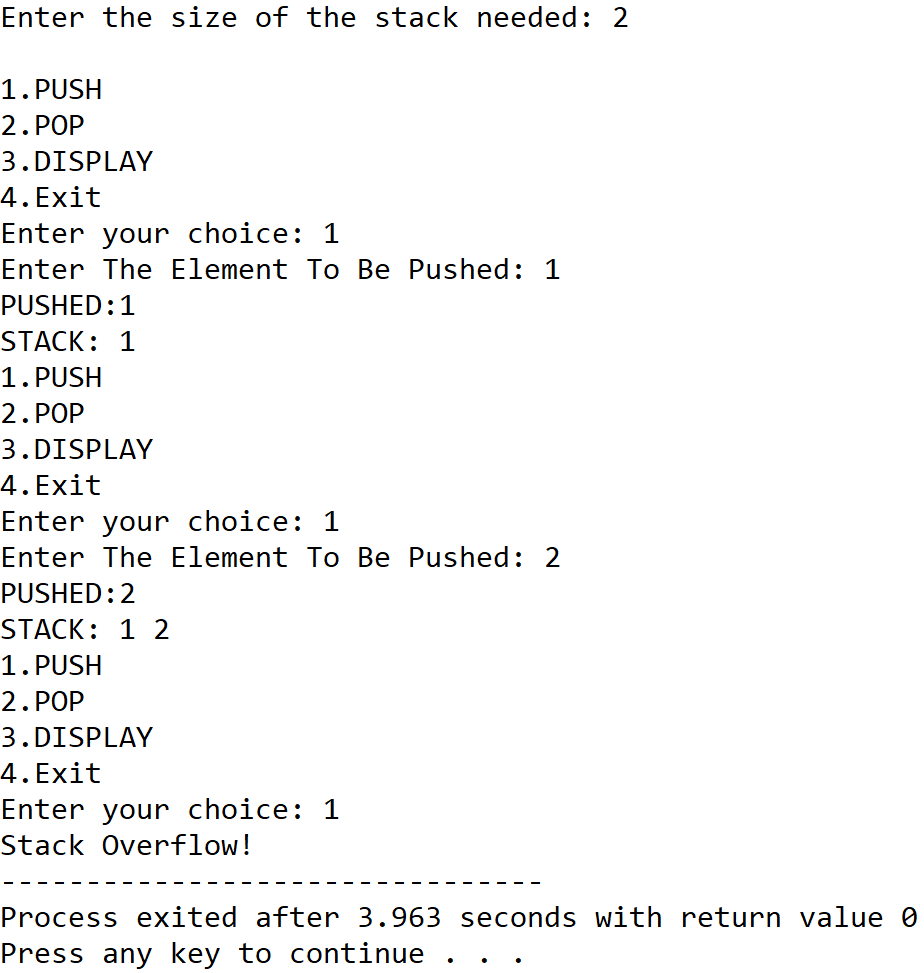
}

}

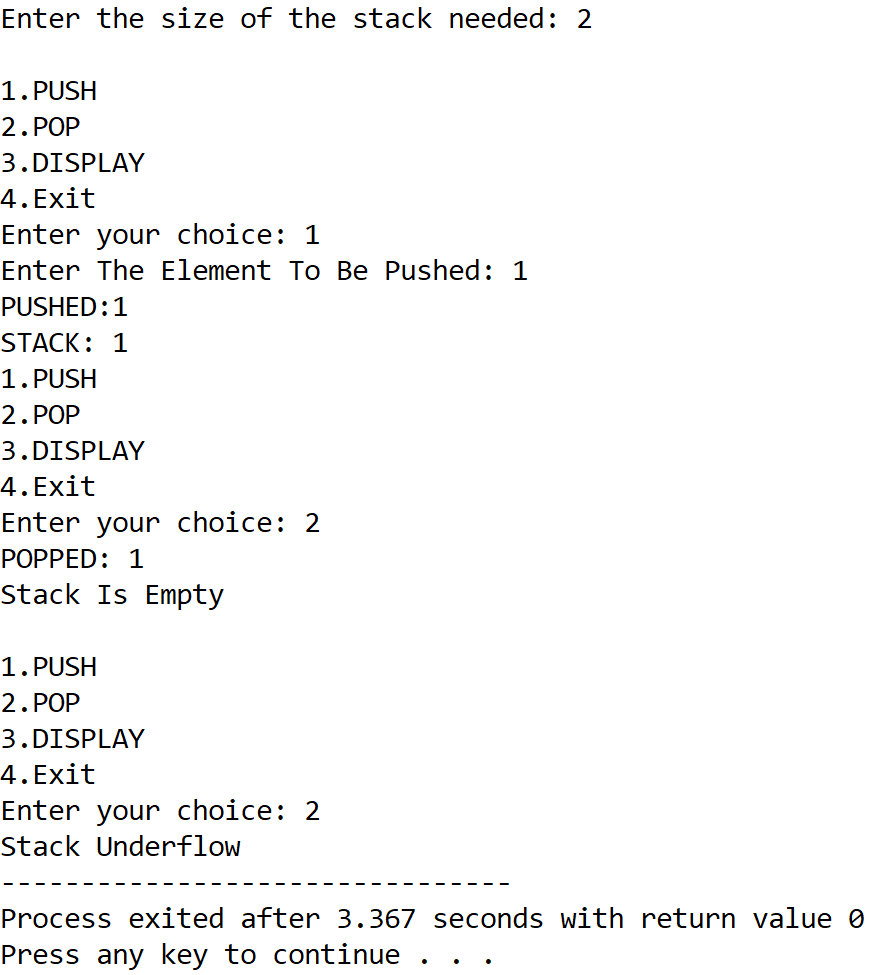
}

**4.OUTPUT**

**SET 1:** Push till Stack Overflow



**SET 2:** Push and Pop till Stack Underflow



**5.DISCUSSIONS**

**Variable Description**

* **\*arr:**pointer to an integer array to represent the stack.
* **max:**Number of elements of the stack.
* **top:** Indicates topmost element of stack.
* **ch:** variable to take user’s choice in switch-case-default.
* **ele:** variable to store variables to push into the stack.
* **i:** loop counter.

**Limitations**

* The program uses array representation of stack, in which stack overflow is very prominent.
* The popped elements are not physically removed from the memory and are just taken out of the scope of the stack, leading to wastage of memory locations.

**Uses**

* Stacks can be used to compute arithmetic expressions.
* Stacks are used by various programming languages to track function calls and store their stackframes.

**Future Scope:**

* The program can be made more memory efficient and the overflow state can be minimized by using a linked list to represent the stack.