NAME: Shreyas Sawant
D7A, 55

SDS | Page No.
Date / /

AIM: Stack implementation using arrays:

THEORY:

- Stacks:

A stack is an ordered collection of items where the addition of new items and the removal of existing items always takes place at the same end. This end is commonly referred to as the "top". The end opposite the top is known as the "base".

The base of stack is significant since items stored in the stack that are closer to base represent those that have been in the stack the longest. The most recently added item is the one that is in position to be removed first. This ordering principle is known as LIFO, last-in-first-out. It provides an ordering based on length of time in the collection. Newer items are near the top, while older items near the base. Fig. 1 represents a stack of numbers

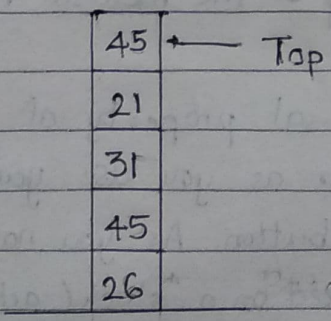



Fig 1 - Stack of numbers

One of the most useful ideas related to stacks comes from the simple observation of items as they are added and then removed. Assume you start out with a clean desk. Now place books one at a time on top of each other. You are constructing a stack. Consider what happens when you begin removing books. The order that they are removed is exactly the order in reverse that they were placed. Stacks are fundamentally important as they can used to reverse order of items. The order of insertion is the reverse of order of removal. Fig. 2 shows the removal property of stacks.

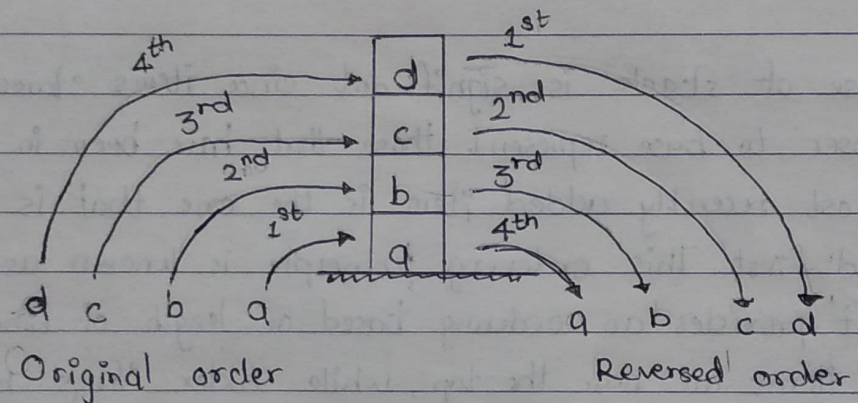
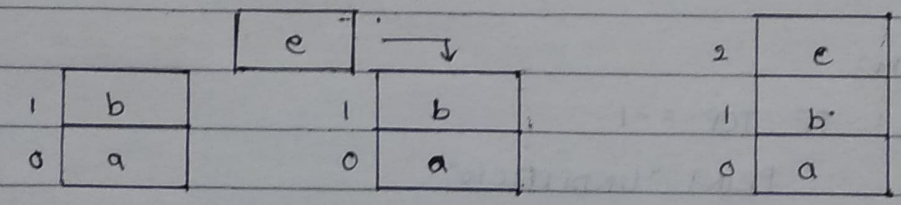



Fig 2. The Reversal Property of Stacks

Considering this reversal property of stacks, you can think of examples of stacks that occur as you use your computer. For example, every web page has a back button. As you navigate from web page to web page those pages are placed on a stack (actually it is the URLs in the stack).

- **Operations on Stack:**
A stack supports three basic operations: push, pop and peek.

i) **Push operation:**
The push operation is used to insert an element into stack. The new element is added at topmost position of stack.
Before inserting an element in stack, overflow conditions are checked if top == max-1, stack is full and no more insertions are allowed.




Intial Stack | Adding element to stack having value 'e' | Stack after insertion of element having value 'e'

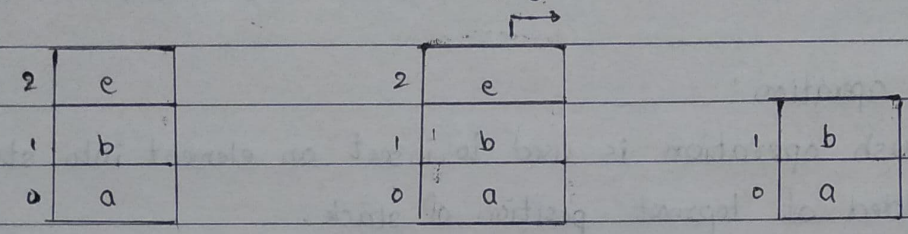
Algorithm:

```
STEP 1 : IF TOP == MAX - 1
             PRINT "OVERFLOW"
             GOTO STEP 4
         [END OF IF]
STEP 2 : SET TOP = TOP+1
STEP 3 : SET STACK[TOP] = VALUE
STEP 4 :   STOP
```

2) Pop operation:

The pop operation is used to delete the topmost element from stack. Before deletion, underflow condition is checked.

if top = = -1, then stak is empty and deletion can't be performed.




Intial Stack | Removing element having value 'e' from stack | Stack after popping

Algorithm:

STEP 1: IF TOP = = -1

PRINT "UNDERFLOW"

GOTO STEP 4

[END OF IF]

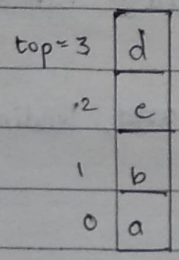
STEP 2: SET VAL = STACK[TOP]

STEP 3: SET TOP = TOP - 1

STEP 4: STOP

3. Peek Operation:

Peek operation is an operation that returns the value of topmost element of the stack without deleting it from stack. However the peek operation first checks if the stack is empty. If top == -1, underflow message is printed else value is returned.




The peek operator returns 'd', as it is value of topmost element of stack.

Algorithm:

STEP 1: IF TOP == -1

PRINT "STACK EMPTY"

GOTO STEP 3

STEP 2: RETURN STACK[TOP]

STEP 3: STOP

SDS | Page No.
Date / /

- **Limitations of Stack:**

i) Stack memory is very limited
ii) Creating too many objects on stack can increase risk of stack overflow.
iii) Random access is not possible
iv) Variable storage can be overwritten, which sometimes leads to undefined behaviour of function or program.
v) Stack might fall outside of memory area, leading to an abnormal behaviour.

SDS | Page No.
Date / /

## CONCLUSION:

Errors encountered:

1. Declared variable 'top' inside the main() function leading to limited access; unable to use in other functions.

```
void main()
{ int top;
}
```

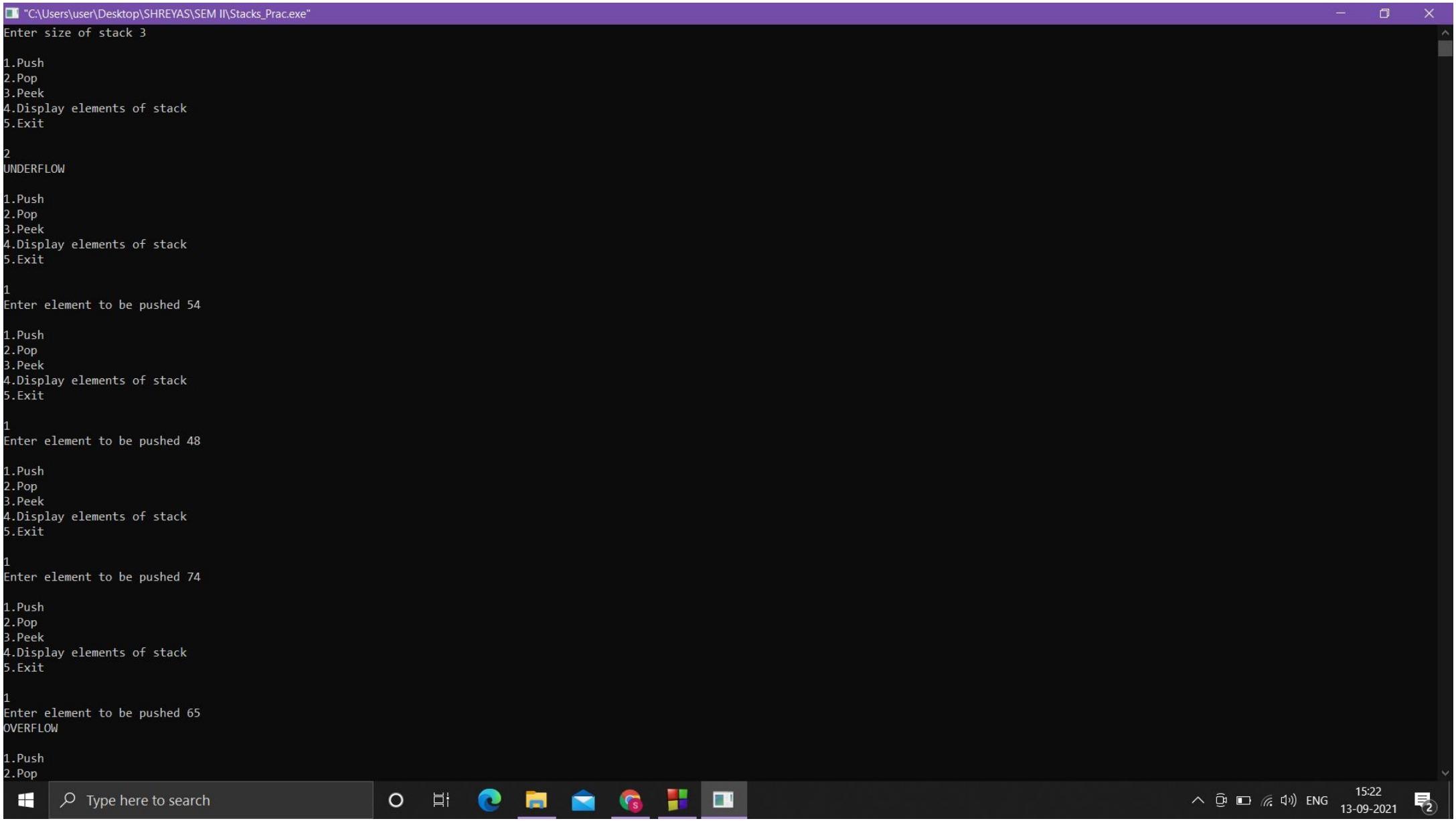
Solution: Declared variable 'top' outside of every function i.e. globally.

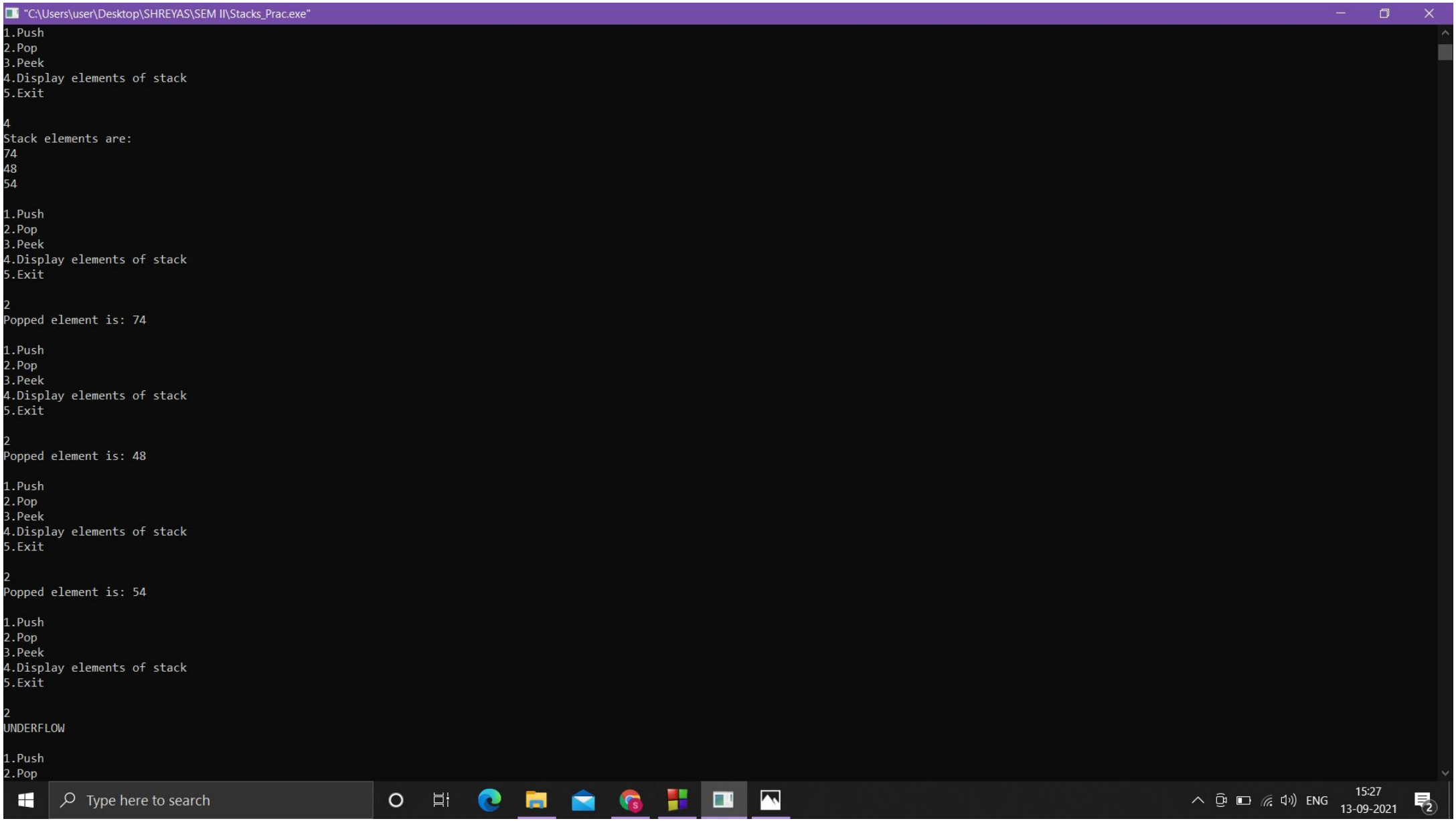
```
int top;
void push();
void pop();
:
Void main()
{ ... }
```

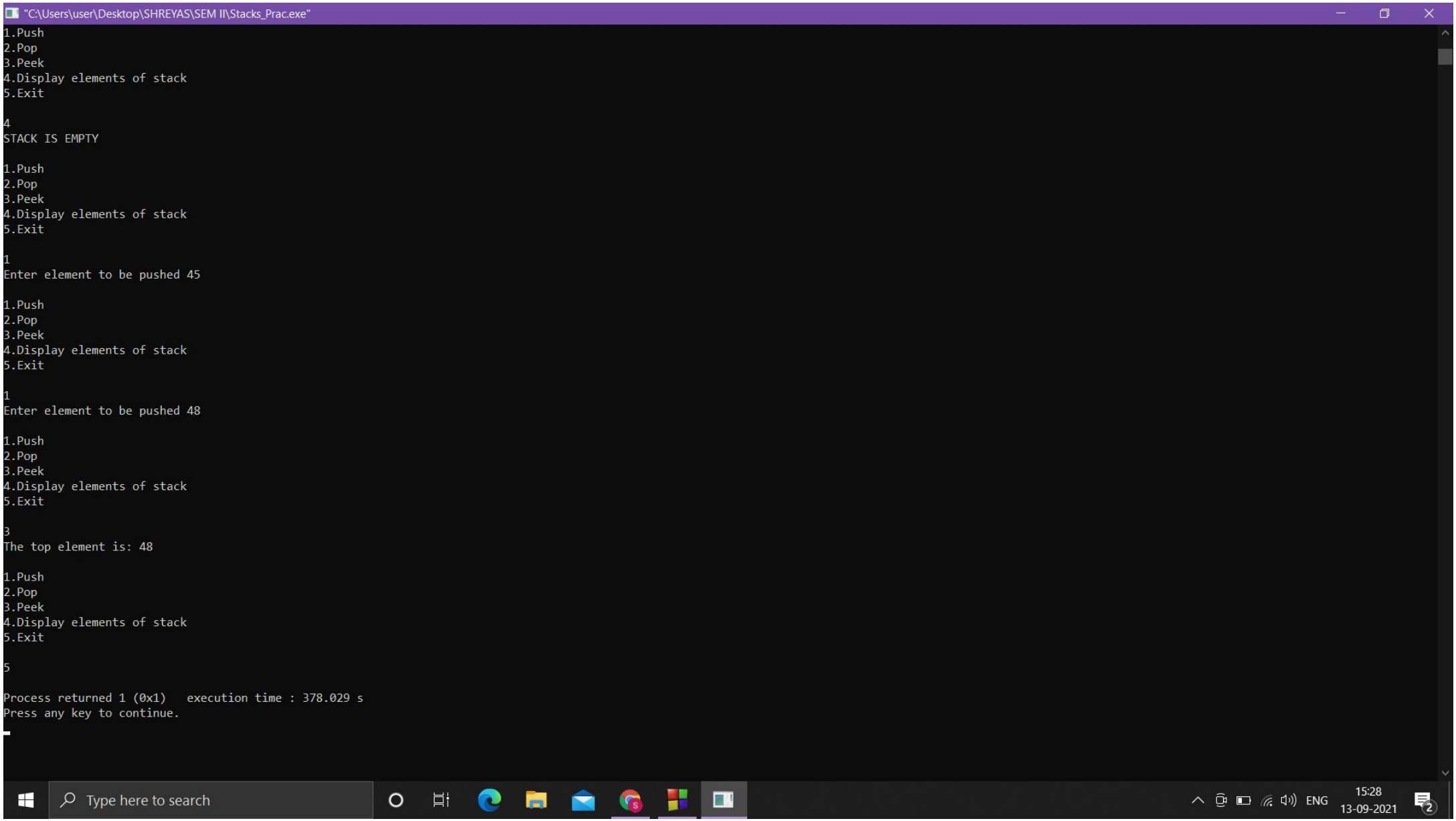
2. Missing break statement causing fall through and in the end, termination of program after reaching case 5.

Solution: Inserting break statement in each case to prevent fall through and to make execution of program properly.

```c
#include <stdio.h>
#include <stdlib.h>
int top=-1,a;
int s[1000];
void push(int c)
{
    if(top==a-1)
    {
        printf("OVERFLOW\n");
    }
    else
    {
        top++;s[top]=c;
    }
}
void pop()
{
    if(top==-1)
    {
        printf("UNDERFLOW\n");
    }
    else
    {

        printf("Popped element is: %d\n",s[top]);top--;

    }
}
void peek()
{
    if(top==-1)
    {
        printf("NO ELEMENTS\n");
    }
    else
        printf("The top element is: %d\n",s[top]);
}

void display()
{   if(top==-1)
    {
        printf("STACK IS EMPTY\n");
    }
    else
    {printf("Stack elements are:\n");
    for(int i=top;i>-1;i--)
    {
        printf("%d\n",s[i]);
    }}
}
void main()
{
    printf("Enter size of stack ");

    scanf("%d",&a); int s[a];
    while(1){
    printf("\n1.Push\n2.Pop\n3.Peek\n4.Display elements of stack\n5.Exit\n\n");
    int n;
    scanf("%d",&n);


    switch(n)
     {


     case 1:
         {
            printf("Enter element to be pushed ");
            int t;
            scanf("%d",&t);
            push(t);break;
         }
     case 2: pop();break;
     case 3: peek();break;
     case 4: display();break;
     case 5: exit(1);
     default: printf("Incorrect choice\n");
     }
    }
 }
```


"C:\Users\user\Desktop\SHREYAS\SEM II\Stacks_Prac.exe"
Enter size of stack 3
1.Push
2.Pop
3.Peek
4.Display elements of stack
5.Exit
2
UNDERFLOW
1.Push
2.Pop
3.Peek
4.Display elements of stack
5.Exit
1
Enter element to be pushed 54
1.Push
2.Pop
3.Peek
4.Display elements of stack
5.Exit
1
Enter element to be pushed 48
1.Push
2.Pop
3.Peek
4.Display elements of stack
5.Exit
1
Enter element to be pushed 74
1.Push
2.Pop
3.Peek
4.Display elements of stack
5.Exit
1
Enter element to be pushed 65
OVERFLOW
1.Push
2.Pop
Type here to search
ENG
15:22
13-09-2021


"C:\Users\user\Desktop\SHREYAS\SEM II\Stacks_Prac.exe"
1.Push
2.Pop
3.Peek
4.Display elements of stack
5.Exit
4
Stack elements are:
74
48
54
1.Push
2.Pop
3.Peek
4.Display elements of stack
5.Exit
2
Popped element is: 74
1.Push
2.Pop
3.Peek
4.Display elements of stack
5.Exit
2
Popped element is: 48
1.Push
2.Pop
3.Peek
4.Display elements of stack
5.Exit
2
Popped element is: 54
1.Push
2.Pop
3.Peek
4.Display elements of stack
5.Exit
2
UNDERFLOW
1.Push
2.Pop
Type here to search
ENG
15:27
13-09-2021


"C:\Users\user\Desktop\SHREYAS\SEM II\Stacks_Prac.exe"
1.Push
2.Pop
3.Peek
4.Display elements of stack
5.Exit
4
STACK IS EMPTY
1.Push
2.Pop
3.Peek
4.Display elements of stack
5.Exit
1
Enter element to be pushed 45
1.Push
2.Pop
3.Peek
4.Display elements of stack
5.Exit
1
Enter element to be pushed 48
1.Push
2.Pop
3.Peek
4.Display elements of stack
5.Exit
3
The top element is: 48
1.Push
2.Pop
3.Peek
4.Display elements of stack
5.Exit
5
Process returned 1 (0x1) execution time : 378.029 s
Press any key to continue.
Type here to search
ENG
15:28
13-09-2021