Experiment :- 1

Aim : Implementation of stack using Array for real word application.

Objective :

1) To introduce the concepts of data structures and analysis procedure.
2) To conceptualize linear data structures and its implementation for various real word applications.

Theory :

1) Introduction to data structure :

Data structures are a way of organizing and storing data in a computer such that it can be retrieved and used effectively. Data structure considers not only the elements stored but also their relationship to each other.

- classification of Data structure

i) Primitive data structure
ii) Non - primitive data structure

- non-primitive data structure classify as :

1) Linear list and non - linear list

Examples of ~~non~~ linear list are Array, stack, Queue, Lists & examples of non-linear list are Graph, trees.

2) Introduction to stack.

- A list means a linear collection of element, for eg. array
- A linear list that follows insertion and deletion at one end only is called stack.
- Elements in stack have the same data type and are ordered by when they were added.

- The only accessible element : Top.
- Also known as last In First Out (LIFO) as the elements are removed in the opposite order of which they were added.

3) Various operations (PUSH, POP, CHANGE, PEEP, DISPLAY)

- PUSH :
PUSH operation refers to inserting an element in the stack, since there's only one position at which the new element can be inserted - ToP of stack, the new element is inserted at the top of stack.

- POP :
POP operation refers to remove an element from the top of the stack (newst element in the stack). The element is removed to the stack container and the size of the stack is decresed by 1.

- PEEP :
PEEP operation is stack function that returns the value of the ~~top~~ ~~most~~ specific element of the stack without deleting that element from stack.

- CHANGE :
CHANGE operation in stack function that user can change or update the contents of the specific element.

- Display :
Display function displays all the elements in the stack.

4) Algorithm :

- PUSH :

Procedure PUSH (S, TOP, X) . This procedure inserts an element x to the top of a stack which is represented by a vector S containing N elements with a pointer TOP denoting the top element in the stack

1. [Check for stack overflow]
   If TOP ≥ N
   then write ('Stack Overflow')
   Return

2. [Increment TOP]
   TOP ← TOP + 1

3. [Insert element]
   S[TOP] ← X

4. [Finished]
   Return

- POP

Function POP (S, TOP) . This function removes the top element from a stack which is represent by vector S & returns this element, TOP is a pointer to the top element of the stack.

1 [Check for underflow on stack]
   If TOP = 0
   then write ('Stack Underflow on Pop')
   take action in response to underflow.
   Exit

2. [Decriment Pointer]
   TOP ← TOP - 1

3. [Return former top element of stack]
Return (S[TOP + 1])

- PEEP

Function PEEP (S, TOP, I) Given a vector S constisting of N element representing a sequentially alloceted stak, and a pointer TOP denoting the top element of the stack, this function returns the value of the Ith element from the top of the stack. The element is not deleted by this function.

1. [Check for stack underflow]
If TOP - I + 1 ≤ 0
then Write ('Stack underflow on PeeP')
take action in response to underflow
EXIT

2. [Return Ith element from top of stack]
Return (S[TOP - I + 1])

- CHANGE

Procedure CHANGE (S, TOP, X, I). As before a vector S consisting N elements represents a sequenctally allocated stack & a pointer TOP denotes the top element of the stack. This procedure changes the value of the Ith element from the top of the stack to the value contained in x.

1. [check for stack underflow]
If TOP - I + 1 ≤ 0
then write ('Stack Underflow on change')
Return

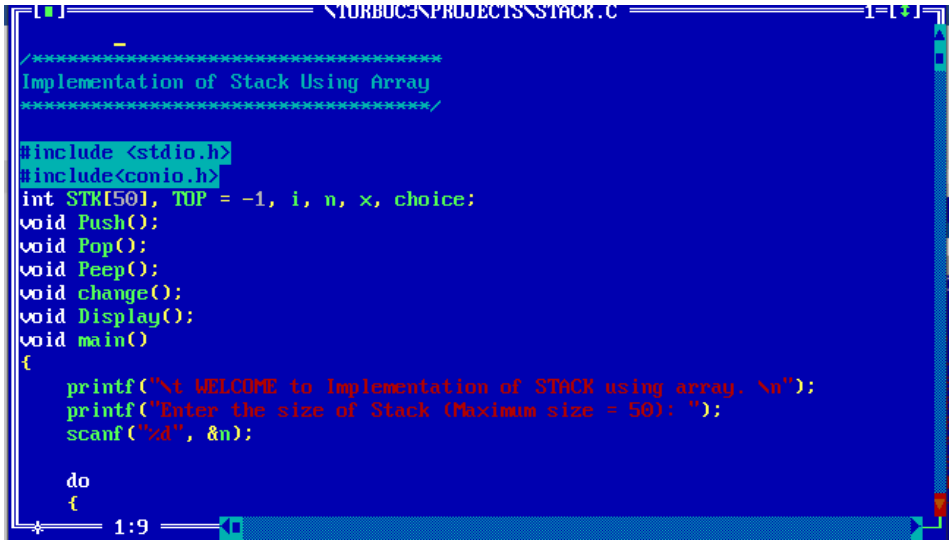
2. [Chang Ith element from top of stack]
S[TOP-I+1] ← x
3. [Finished]
Return.

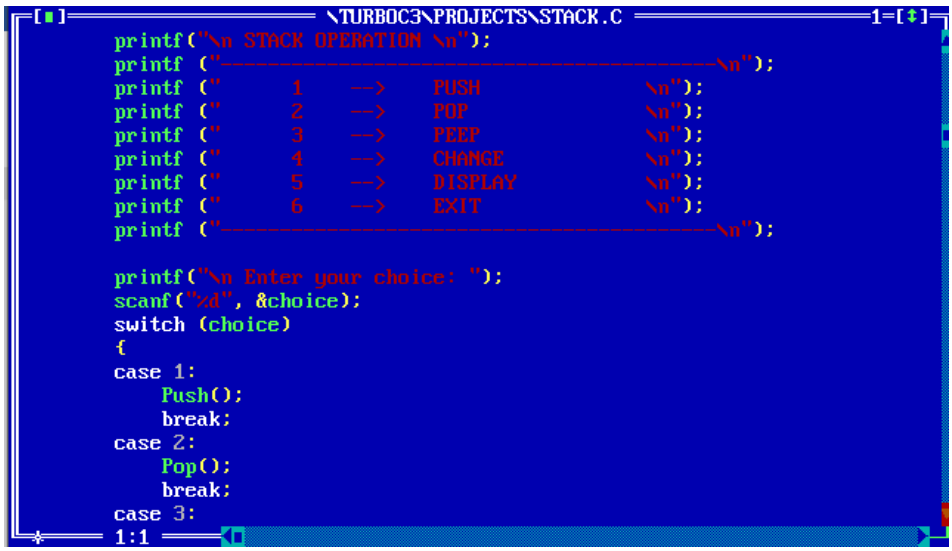
-Display
1. Check whether stack is EMPTY (TOP == -1)
2. If it is EMPTY, then display "stack is EMPTY" and terminate the function.
3. If it is NOT EMPTY, then define a variable 'i'. And initialize with top. Display stack[i] value and decrement i value by one (i--).
4. Repeat above step until i value becomes '0'.

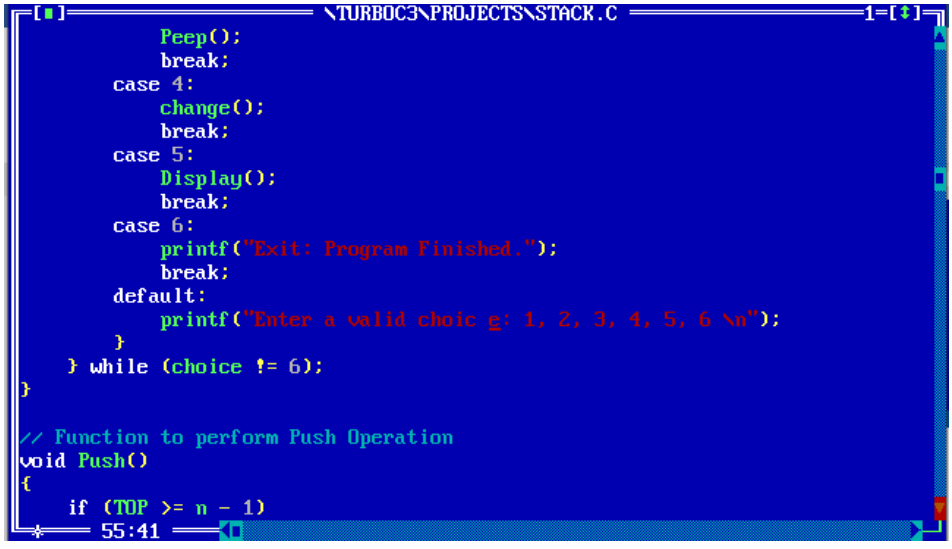
5) Example:
1) Deck of card - We can place or remove a card from the top of the stack only.
2) Pile of Books - We can place or remove book from the top of the stack only.

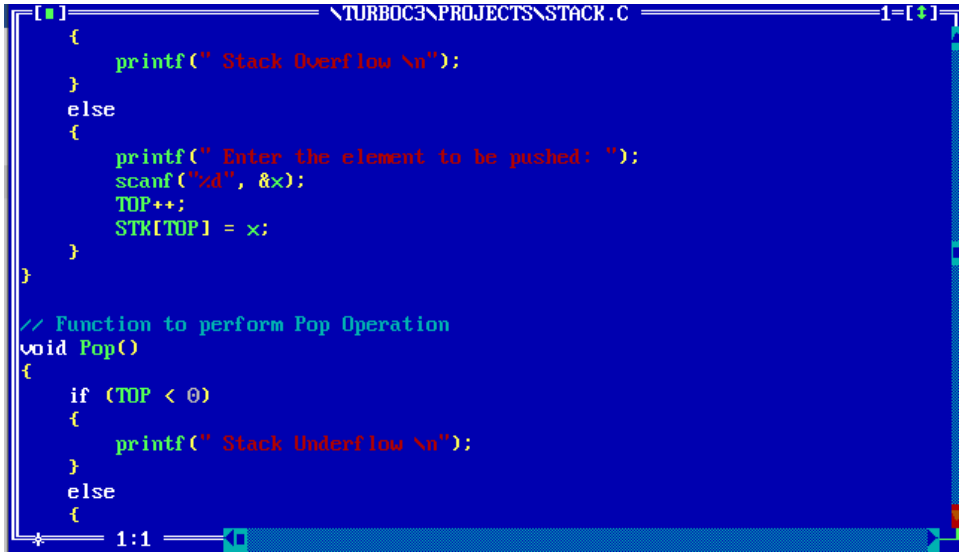
Conclusion: From this experiment, we lern how to implement stack and real life application of stack. How to perform various operation of stack with the help of algorithm.

Outcome:
Apply the concepts of stacks for real-world applications.

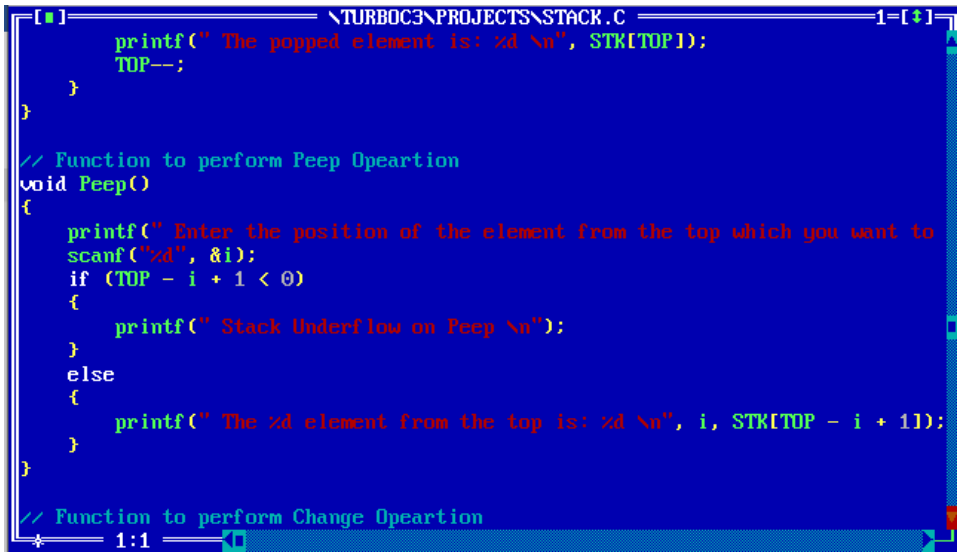

\TURBOC3\PROJECTS\STACK.C
1=[↕]
/*************************************
Implementation of Stack Using Array
*************************************/
#include <stdio.h>
#include<conio.h>
int STK[50], TOP = -1, i, n, x, choice;
void Push();
void Pop();
void Peep();
void change();
void Display();
void main()
{
printf("\t WELCOME to Implementation of STACK using array. \n");
printf("Enter the size of Stack (Maximum size = 50): ");
scanf("%d", &n);
do
{
1:9


\TURBOC3\PROJECTS\STACK.C
1=[↕]
printf("\n STACK OPERATION \n");
printf ("---\n");
printf (" 1 --> PUSH \n");
printf (" 2 --> POP \n");
printf (" 3 --> PEEP \n");
printf (" 4 --> CHANGE \n");
printf (" 5 --> DISPLAY \n");
printf (" 6 --> EXIT \n");
printf ("---\n");
printf("\n Enter your choice: ");
scanf("%d", &choice);
switch (choice)
{
case 1:
Push();
break;
case 2:
Pop();
break;
case 3:
1:1

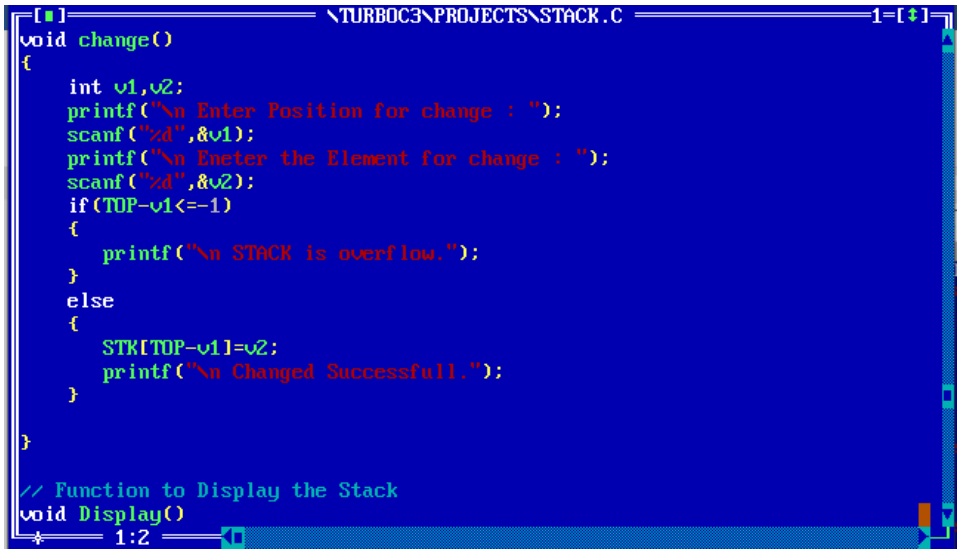

\TURBOC3\PROJECTS\STACK.C
1=[↕]
Peep();
break;
case 4:
change();
break;
case 5:
Display();
break;
case 6:
printf("Exit: Program Finished.");
break;
default:
printf("Enter a valid choic e: 1, 2, 3, 4, 5, 6 \n");
}
} while (choice != 6);
}
// Function to perform Push Operation
void Push()
{
if (TOP >= n - 1)
55:41



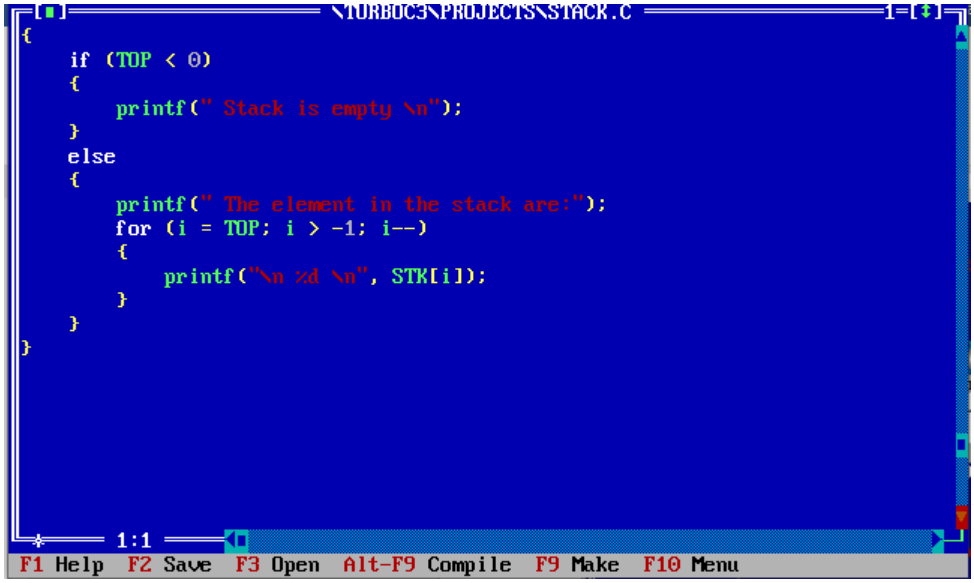
```
\TURBOC3\PROJECTS\STACK.C
    {
        printf(" Stack Overflow \n");
    }
    else
    {
        printf(" Enter the element to be pushed: ");
        scanf("%d", &x);
        TOP++;
        STK[TOP] = x;
    }
}

// Function to perform Pop Operation
void Pop()
{
    if (TOP < 0)
    {
        printf(" Stack Underflow \n");
    }
    else
    {
1:1
```

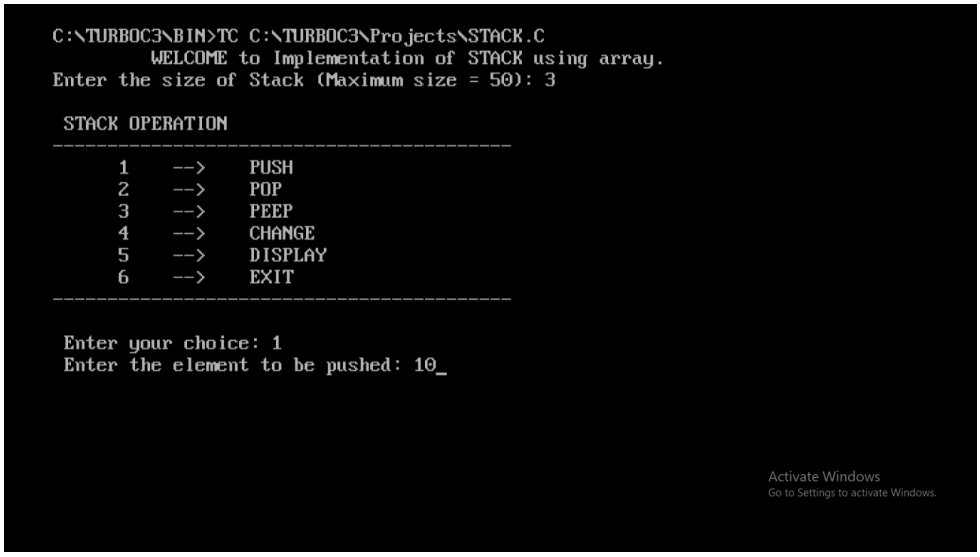


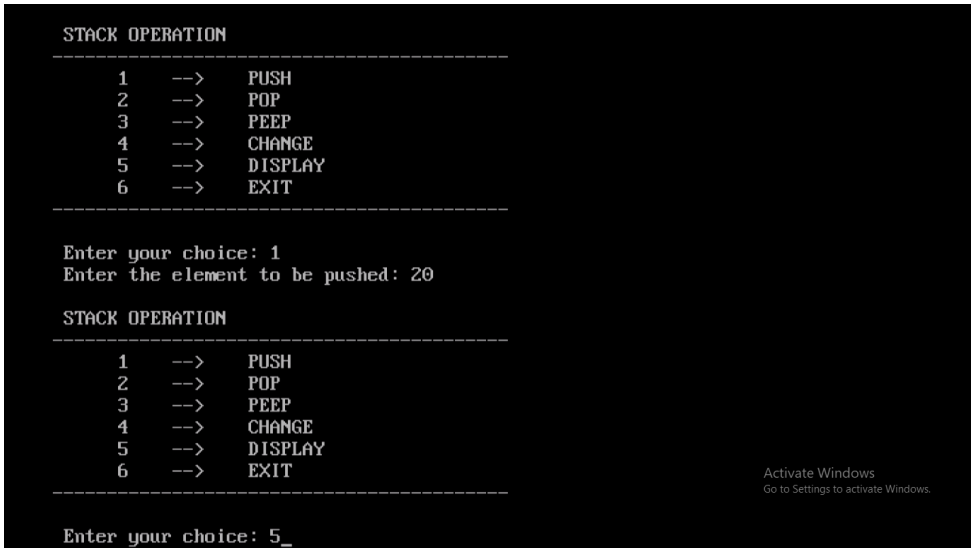
```
\TURBOC3\PROJECTS\STACK.C
        printf(" The popped element is: %d \n", STK[TOP]);
        TOP--;
    }
}

// Function to perform Peep Opeartion
void Peep()
{
    printf(" Enter the position of the element from the top which you want to
    scanf("%d", &i);
    if (TOP - i + 1 < 0)
    {
        printf(" Stack Underflow on Peep \n");
    }
    else
    {
        printf(" The %d element from the top is: %d \n", i, STK[TOP - i + 1]);
    }
}

// Function to perform Change Opeartion
1:1
```



```
\TURBOC3\PROJECTS\STACK.C
void change()
{
    int v1,v2;
    printf("\n Enter Position for change : ");
    scanf("%d",&v1);
    printf("\n Eneter the Element for change : ");
    scanf("%d",&v2);
    if(TOP-v1<=-1)
    {
       printf("\n STACK is overflow.");
    }
    else
    {
       STK[TOP-v1]=v2;
       printf("\n Changed Successfull.");
    }

}

// Function to Display the Stack
void Display()
1:2
```


\TURBOC3\PROJECTS\STACK.C
{
if (TOP < 0)
{
printf(" Stack is empty \n");
}
else
{
printf(" The element in the stack are:");
for (i = TOP; i > -1; i--)
{
printf("\n %d \n", STK[i]);
}
}
}
1:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu


C:\TURBOC3\BIN>TC C:\TURBOC3\Projects\STACK.C
WELCOME to Implementation of STACK using array.
Enter the size of Stack (Maximum size = 50): 3
STACK OPERATION
1 --> PUSH
2 --> POP
3 --> PEEP
4 --> CHANGE
5 --> DISPLAY
6 --> EXIT
Enter your choice: 1
Enter the element to be pushed: 10
Activate Windows
Go to Settings to activate Windows.


STACK OPERATION
1 --> PUSH
2 --> POP
3 --> PEEP
4 --> CHANGE
5 --> DISPLAY
6 --> EXIT
Enter your choice: 1
Enter the element to be pushed: 20
STACK OPERATION
1 --> PUSH
2 --> POP
3 --> PEEP
4 --> CHANGE
5 --> DISPLAY
6 --> EXIT
Enter your choice: 5
Activate Windows
Go to Settings to activate Windows.

```
     1    -->    PUSH
     2    -->    POP
     3    -->    PEEP
     4    -->    CHANGE
     5    -->    DISPLAY
     6    -->    EXIT
----------------------------------------

Enter your choice: 5
The element in the stack are:
20

10

STACK OPERATION
----------------------------------------
     1    -->    PUSH
     2    -->    POP
     3    -->    PEEP
     4    -->    CHANGE
     5    -->    DISPLAY
     6    -->    EXIT
----------------------------------------

Enter your choice: 2
```

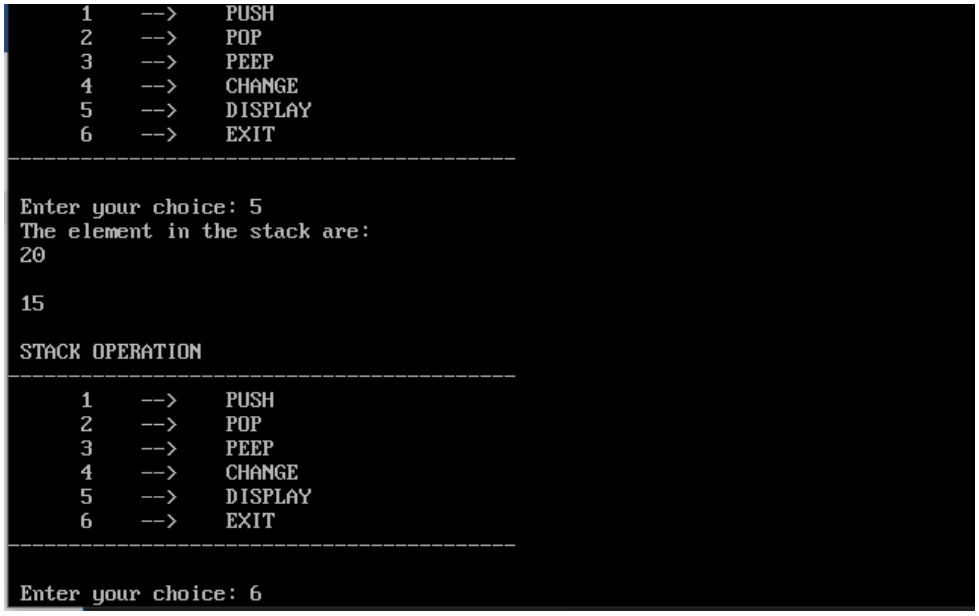
```
STACK OPERATION
----------------------------------------
     1    -->    PUSH
     2    -->    POP
     3    -->    PEEP
     4    -->    CHANGE
     5    -->    DISPLAY
     6    -->    EXIT
----------------------------------------

Enter your choice: 2
The popped element is: 20

STACK OPERATION
----------------------------------------
     1    -->    PUSH
     2    -->    POP
     3    -->    PEEP
     4    -->    CHANGE
     5    -->    DISPLAY
     6    -->    EXIT
----------------------------------------

Enter your choice: 5
```

```
     5    -->    DISPLAY
     6    -->    EXIT
----------------------------------------

Enter your choice: 5




The element in the stack are:
10

STACK OPERATION
----------------------------------------
     1    -->    PUSH
     2    -->    POP
     3    -->    PEEP
     4    -->    CHANGE
     5    -->    DISPLAY
     6    -->    EXIT
----------------------------------------

Enter your choice: 1
Enter the element to be pushed: 20
```

```
    STACK OPERATION
    ----------------------------------------
         1    -->    PUSH
         2    -->    POP
         3    -->    PEEP
         4    -->    CHANGE
         5    -->    DISPLAY
         6    -->    EXIT
    ----------------------------------------

    Enter your choice: 3
    Enter the position of the element from the top which you want to peep: 2
    The 2 element from the top is: 10

    STACK OPERATION
    ----------------------------------------
         1    -->    PUSH
         2    -->    POP
         3    -->    PEEP
         4    -->    CHANGE
         5    -->    DISPLAY
         6    -->    EXIT
    ----------------------------------------

    Enter your choice: 4
```



```
     2    -->    POP
     3    -->    PEEP
     4    -->    CHANGE
     5    -->    DISPLAY
     6    -->    EXIT
----------------------------------------

Enter your choice: 4

Enter Position for change : 1

Eneter the Element for change : 15

Changed Successfull.
STACK OPERATION
----------------------------------------
     1    -->    PUSH
     2    -->    POP
     3    -->    PEEP
     4    -->    CHANGE
     5    -->    DISPLAY
     6    -->    EXIT
----------------------------------------

Enter your choice: 5
```


1 --> PUSH
2 --> POP
3 --> PEEP
4 --> CHANGE
5 --> DISPLAY
6 --> EXIT
Enter your choice: 5
The element in the stack are:
20
15
STACK OPERATION
1 --> PUSH
2 --> POP
3 --> PEEP
4 --> CHANGE
5 --> DISPLAY
6 --> EXIT
Enter your choice: 6