Aim:

Write a program to implement a Stack using Array. The menu-driven program must have the following operations:

1) Push

2) Pop

3) Peek

4) Display

5) Exit

Algorithm:

**Step 1** - START

**Step 2** - declare functions push, pop, peek, and display.

**Step 3** - declare six integers ‘c’, ‘top’, ‘n’, ‘x’, ‘i’, and one array ‘s’ of size 100 as integer data type.

**Step 4** - define functions push, pop, peek, display.

**Step 5** - under push function, check if the stack is full or not by comparing top with (n-1). If the stack is full, display "Stack Overflow!".

**Step 6** - else, get an integer from the user to be added to the stack and store it in variable x. Then, increment top by 1 and set s[top]=x, which pushes the element x into the address pointed by top.

**Step 7** - under pop function, check if the stack is empty or not by comparing top with the base of the array, 0. If the stack is empty, display "Stack Underflow!".

**Step 8** - else, decrement top by 1.

**Step 9** - under peek function, display the value stored in the stack pointed by top.

**Step 10** - under display function, Initialize variable i = top.

**Step 11** - check for condition (i >= 0). If the condition is true, go to step 12 otherwise exit the loop.

**Step 12** - display the elements s[i], decrement i by 1, and go to step 11.

**Step 13** - initialize top = -1.

**Step 14** - get the size of the stack from the user and store the value in variable ‘n’.

**Step 15** - display the title of the program “STACK IMPLEMENTATION IN ARRAY” and the list of operations: 1) Push, 2) Pop, 3) Peek, 4) Display, and 5) Exit.

**Step 16** - introduce a do-while loop, where the body of the do-while loop is executed once and then condition (c<=4) is evaluated. If True, go to step 17. Else, go to step 23.

**Step 17** - under the body of the do-while loop, get the operation to be performed on the stack from the user and store the value in variable c.

**Step 18** - introduce a switch statement with variable c as its input. Check if variable c satisfies the expected cases.

**Step 19** - if variable c is 1, call the push function and go to step 6. Then break from the statement.

**Step 20** - if variable c is 2, call the pop function and go to step 8. Then break from the statement.

**Step 21** - if variable c is 3, call the peek function and go to step 10. Then break from the statement.

**Step 22** - if variable c is 4, call the display function and go to step 11. Then break from the statement.

**Step 23** - if variable c is 5, break from the statement.

**Step 24** - if neither of the cases is satisfied, provide a default statement that displays "Invalid Input! - Enter 1,2,3,4 or 5" and break from the statement.

**Step 25** - STOP

Result:

The program is successfully executed and achieved the aim of the program.

Output:

  