EXPERIMENT 4

Aim:

1. Implement a stack using an array having following functionalities:
    a. isEmpty – to check if the stack if empty or not
    b. isFull – to check if the stack if full or not
    c. push – to insert the element into the stack
    d. pop – to delete an element from the stack
    e. print_top – to print the top most element of the stack.

Theory (Stack using Array):

- **Stack** is a linear data structure following the Last In, First Out (LIFO) principle.
- **Array-based stack** uses a fixed-size array to store elements, with an index variable top pointing to the most recently added element.
- **Operations**:

    a. **Push (Insert)**: Adds an element to the top of the stack. If the stack is full (top == n-1), an overflow condition occurs.

    b. **Pop (Delete)**: Removes and returns the top element from the stack. If the stack is empty (top == -1), an underflow condition occurs.

    c. **isFull() and isEmpty()**: Check if the stack is full or empty before performing operations.

    d. **printTop()**: Displays the top element without modifying the stack.

- **Time Complexity**: Each operation (push, pop, check) has O(1) time complexity, ensuring constant time for basic stack manipulations.

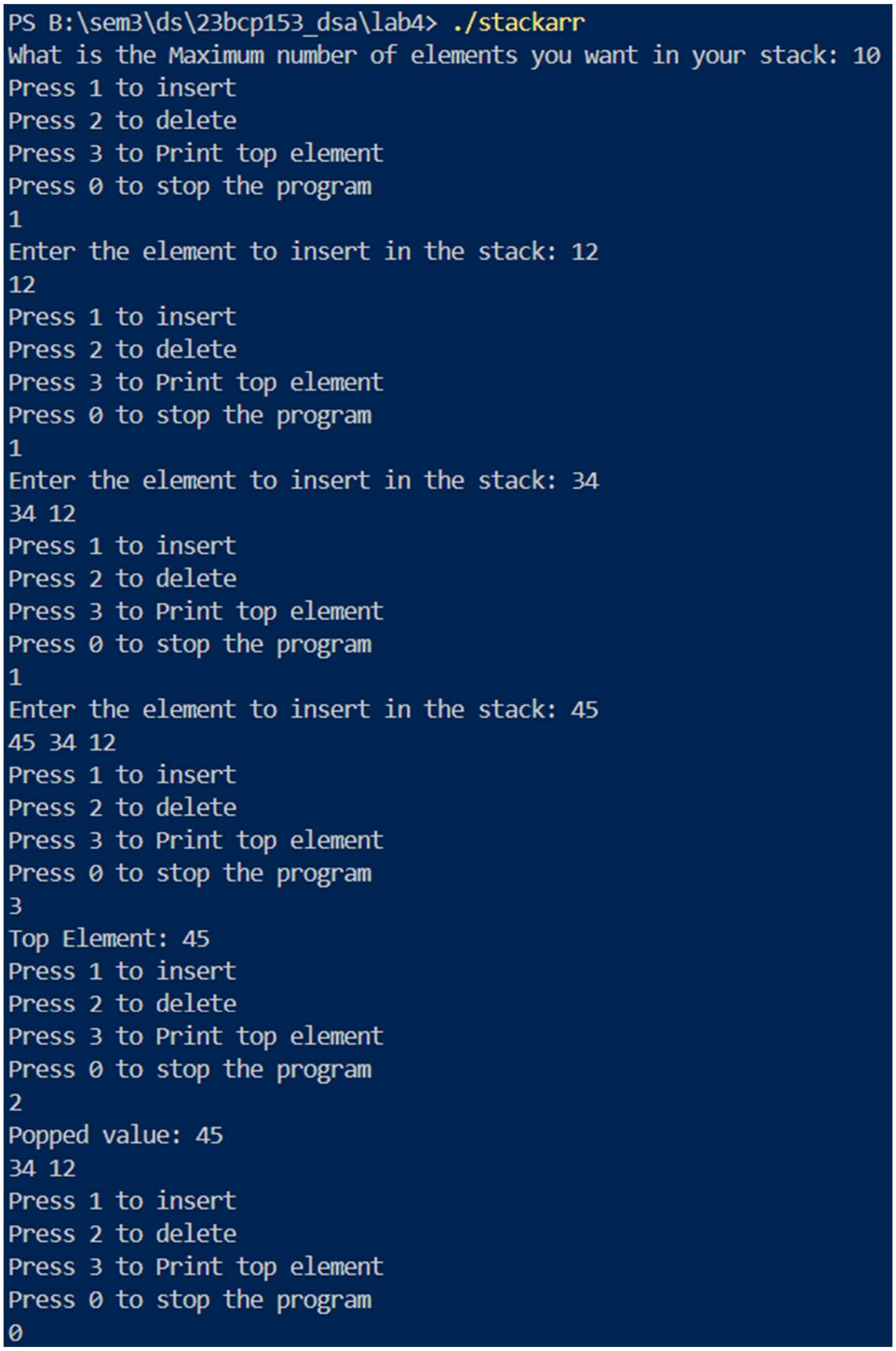
Program:

```
#include <stdio.h>
#include <stdbool.h>

bool isEmpty();
bool isFull(int n);
void insert(int arr[], int ele);
void delete(int arr[]);
void printStack(int arr[]);
void printTop(int arr[]);
int top = -1;

int main(void)
{
   int n;
   printf("What is the Maximum number of elements you want in your stack: ");
   scanf("%d", &n);

   int arr[n];
   int ele;
```

```
    while(true)
    {
        int operation;
        printf("Press 1 to insert\nPress 2 to delete\nPress 3 to Print top element\nPress 0 to stop the
program\n");
        scanf("%d", &operation);

        switch (operation)
        {
        case 1:
            printf("Enter the element to insert in the stack: ");
            scanf("%d", &ele);

            insert(arr, ele);
            break;
        case 2:
            delete(arr);
            break;
        case 3:
            printTop(arr);
            break;
        default:
            return 0;
            break;
        }

    }

    return 0;
}

bool isEmpty()
{
    if (top == -1)
    {
        return true;
    }
    return false;
}

bool isFull(int n)
{
    if (top == n - 1)
    {
        return true;
    }

    return false;
}

void insert(int arr[], int ele)
```

```
{
    if (isFull(top))
    {
        printf("Array is Full!\n");
        printStack(arr);

        return;
    }

    top++;

    arr[top] = ele;
    printStack(arr);

    return;
}

void delete(int arr[])
{
    if(isEmpty())
    {
        printf("Stack is Empty\n");
        return;
    }

    int popped = arr[top];
    top--;
    printf("Popped value: %d\n", popped);
    printStack(arr);
    return;
}

void printStack(int arr[])
{
    for (int i = top; i >= 0; i--)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return;
}

void printTop(int arr[])
{
    printf("Top Element: %d\n", arr[top]);
}
```

Output:



```
PS B:\sem3\ds\23bcp153_dsa\lab4> ./stackarr
What is the Maximum number of elements you want in your stack: 10
Press 1 to insert
Press 2 to delete
Press 3 to Print top element
Press 0 to stop the program
1
Enter the element to insert in the stack: 12
12
Press 1 to insert
Press 2 to delete
Press 3 to Print top element
Press 0 to stop the program
1
Enter the element to insert in the stack: 34
34 12
Press 1 to insert
Press 2 to delete
Press 3 to Print top element
Press 0 to stop the program
1
Enter the element to insert in the stack: 45
45 34 12
Press 1 to insert
Press 2 to delete
Press 3 to Print top element
Press 0 to stop the program
3
Top Element: 45
Press 1 to insert
Press 2 to delete
Press 3 to Print top element
Press 0 to stop the program
2
Popped value: 45
34 12
Press 1 to insert
Press 2 to delete
Press 3 to Print top element
Press 0 to stop the program
0
```

Time Complexity:

Each operation (push, pop, check) has $O(1)$ time complexity, ensuring constant time for basic stack manipulations.