Experiment - 1

Student Name: Ayush Ranjan
UID: 23BCS10187
Branch: BE-CSE
Section/Group: KRG-2-B
Semester: 5th
Date of Performance:24/7/25
Subject Name: DAA
Subject Code: 23CSH-301

1. **Aim:** Analyze if the stack is empty or full, and if elements are present, return the top element in the stack using templates. Also, perform push and pop operations on the stack.

2. Procedure:

- Start by defining a **class** for the stack, so it works for any data type (like int, float, char, etc.).
- Inside the class, create an array to hold stack elements and variables to track top and maximum size.
- Implement push() to add an element if the stack is not full.
- Implement pop() to remove the top element if the stack is not empty.
- Provide isEmpty() and isFull() methods to check the stack condition.
- Write a peek() function to return the top element without removing it.
- In the main() function, test all these operations with user-defined values.

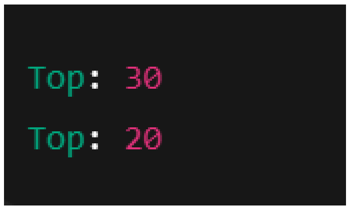
3. Code:

```
#include <iostream>
using namespace std;

class Stack {
    int arr[100];
    int top;
    int size;
public:
    Stack(int s = 10) {
        size = s;
        top = -1;
    }
    bool isEmpty() {
        return top == -1;
    }
    bool isFull() {
        return top == size - 1;
    }
    void push(int val) {
        if (isFull()) {
            cout << "Stack Full" << endl;
        } else {
            arr[++top] = val;
        }
    }
    void pop() {
        if (isEmpty()) {
            cout << "Stack Empty" << endl;
        } else {
            top--;
        }
    }
    int peek() {
        if (isEmpty()) {
            cout << "Stack Empty" << endl;
            return -1;
        } else {
            return arr[top];
        }
    }
};

int main() {
    Stack s(5);
    s.push(10);
    s.push(20);
    s.push(30);
    cout << "Top: " << s.peek() << endl;
    s.pop();
    cout << "Top: " << s.peek() << endl;
    return 0;
  }
```

4. Output:



```
Top: 30
Top: 20
```

5. Learning Outcomes:

- Learnt how to implement a stack using arrays and basic operations.
- Gained understanding of push, pop, and peek functionalities in stack.
- Understood how to check and handle stack overflow and underflow conditions.
- Practised using object-oriented approach for stack implementation in C++.
- Developed confidence in applying stack operations to solve real problems