Experiment-1

Student Name: Tejinderpal Singh
UID: 23BCS80343
Branch: BE-CSE
Section/Group: 23BCS_KRG_3A
Semester: 5th
Subject Code: 23CSH-301
Subject Name: DAA

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. **Objective:** To implement a generic stack data structure in C++ using class templates that supports standard stack operations such as push, pop, peek (view top element), isEmpty, and isFull. The program demonstrates type flexibility through templates, allowing stack operations on any data type, and provides real-time feedback about the stack's current status after performing operations.

3. **Procedure:**

   1. Create a stack with a fixed size array and a variable top initialized to -1.
   2. Check for stack overflow when trying to push an element.
   3. Check for stack underflow when trying to pop an element.
   4. Increment top to store an element during push operation.
   5. Store the element at the current top position.
   6. Decrement top during pop operation to remove the top element.
   7. Display the top element using the peek operation.
   8. Check if the stack is empty by verifying if top is -1.
   9. Check if the stack is full by verifying if top is equal to n - 1.
   10. Perform operations as per user input or program flow.

4. **Code:**

```
#include <iostream>
using namespace std;
const int n = 5;
template <class T>
class Stack {
   T arr[n];
```

```
    int top = -1;
public:
    void push(T a) {
        arr[++top] = a;
    }
    void pop() {
        top--;
    }
    void peek() {
        cout << "Top element: " << arr[top] << endl;
    }
    bool isEmpty() {
        return top == -1;
    }
    bool isFull() {
        return top == n - 1;
    }
};
int main() {
    Stack<int> s;
    s.push(5);
    s.push(10);
    s.push(15);
    s.push(20);
    s.peek();
    s.pop();
    s.peek();
    if (s.isEmpty()){
        cout << "Stack is empty" << endl;
    }
    else{
        cout << "Stack is not empty" << endl;
    }
    if (s.isFull()){
        cout << "Stack is full" << endl;
    }
    else{
        cout << "Stack is not full" << endl;
    }
```

```
    return 0;
}
```

5. Observation:

```
Top element: 20
Top element: 15
Stack is not empty
Stack is not full

c:\Users\singh\Desktop\VS Code\CPP>
```

6. Time Complexity: O(1) for every operation

7. Learning Outcome:

- The program demonstrates the use of C++ class templates to implement a generic stack that can handle multiple data types.
- It reinforces understanding of stack operations such as push, pop, peek, isEmpty(), and isFull() in a static array-based implementation.
- The use of conditional checks for empty and full stack conditions promotes robust programming and error prevention.
- Encapsulation of stack logic within a class helps in understanding object-oriented programming principles.
- The program enhances practical knowledge of template programming and encourages reusable, scalable code design in C++.