**Objective:**

* To understand the concept of templates in C++.

**Challenge - 1:** S*tack Implementation*  **(**15**)**

Imagine you're working at a cafeteria during lunchtime rush hour. Customers are lining up to get their meals, and you're tasked with stacking trays as they come in. You have a limited amount of space to stack trays, and you need to ensure that you manage them efficiently to avoid any chaos or accidents.

**Explanation:**

In this scenario, you are essentially managing a stack of trays, where you can only add new trays on top and remove trays from the top. This concept is analogous to a data structure called a "stack" in computer science.

**Stack:**

A stack is a linear data structure that follows the Last In, First Out (LIFO) principle. It means that the last element added to the stack is the first one to be removed. Imagine a stack of plates where you can only add or remove plates from the top. This analogy illustrates the behavior of a stack data structure.

**Characteristics of a Stack:**

Last In, First Out (LIFO): Just like stacking trays, the last tray added to the stack is the first one to be removed. This principle is known as Last In, First Out (LIFO).

**Operations:**

Push: Adding a new element on top of the stack.

Pop: Removing the top element from the stack.

Top (Peek): Viewing the top element without removing it.

|  |
| --- |
| **Stack.h** |
| template<typename T>  class Stack  {  T \*elements;  int size;  int topIndex;  void reSize();  public:  Stack();  Stack(int size);  ~Stack();  bool isEmpty() const;  bool isFull() const;  void push(const T& value);  void pop();  T& top();  }; |

|  |  |
| --- | --- |
| **Sample Run** | **Output** |
| int main()  {  // Create a stack of integers  Stack<int> intStack; |  |
| // Push elements onto the stack  intStack.push(10);  intStack.push(20);  intStack.push(30); |  |
| // Display the top element  cout << "Top element: " << intStack.top() <<endl; | Top element: 30 |
| // Pop elements from the stack  intStack.pop();  intStack.pop(); |  |
| // Display the top element after popping  cout << "Top element after popping: " << intStack.top() << endl; | Top element after popping: 10 |
| cout << intStack.isEmpty();  return 0;  } | 0 |