CSE225L – Data Structures and Algorithms Lab
Lab 08
Stack (Array-based)

In today's lab we will design and implement the Stack ADT using array.

`stacktype.h`

```
#ifndef STACKTYPE_H
#define STACKTYPE_H

const int SIZE = 5;

// Exception class thrown by Push when the stack is full
class FullStack {};

// Exception class thrown by Pop and Top when the stack is empty
class EmptyStack {};

template<class T>
class StackType
{
private:
    T* data;
    int top;
public:
    StackType();
    ~StackType();
    bool IsFull();
    bool IsEmpty();
    void Push(T);
    void Pop();
    T Top();
};

#endif // STACKTYPE_H
```

`stacktype.cpp`

```
#include <iostream>
#include "stacktype.h"

using namespace std;

template<class T>
StackType<T>::StackType()
{
    data = new T[SIZE];
    top = -1;
}

template<class T>
StackType<T>::~StackType()
{
    delete[] data;
}

template<class T>
bool StackType<T>::IsEmpty()
{
    return (top == -1);
}
```

```
template<class T>
bool StackType<T>::IsFull()
{
    return (top == SIZE - 1);
}

template<class T>
void StackType<T>::Push(T value)
{
    try
    {
        if (IsFull())
        {
            throw FullStack();
        }
        else
        {
            top++;
            data[top] = value;
        }
    }
    catch (FullStack e)
    {
        cout << "Error: Stack is full" << endl;
    }
}

template<class T>
void StackType<T>::Pop()
{
    try
    {
        if (IsEmpty())
        {
            throw EmptyStack();
        }
        else
        {
            top--;
        }
    }
    catch (EmptyStack e)
    {
        cout << "Error: Stack is empty" << endl;
    }
}

template<class T>
T StackType<T>::Top()
{
    try
    {
        if (IsEmpty())
        {
            throw EmptyStack();
        }
        else
        {
            return data[top];
        }
    }
    catch (EmptyStack e)
    {
        cout << "Error: Stack is empty" << endl;
    }
}
```

Generate the **driver file (main.cpp)** where you perform the following tasks. Note that you cannot make any change to the header file or the source file.

Operation to Be Tested and Description of Action	**Input Values**	**Expected Output**
Create a stack of integers		
Check if the stack is empty		Stack is Empty
Push four items	5, 7, 4, 2	
Check if the stack is empty		Stack is not Empty
Check if the stack is full		Stack is not full
Print the values in the stack (in the order the values are given)		5, 7, 4, 2
Push another item	3	
Print the values in the stack		5, 7, 4, 2, 3
Check if the stack is full		Stack is full
Pop two items		
Print top item		4
Take strings of parentheses as input from the user and <u>use a stack</u> to check if each string is balanced.	()	Balanced
	(( )) ( ) (( ) ( )) ( )	Balanced
	(( )) ( ) ((( )	Not Balanced
	(( )))) ((( )	Not Balanced
	(( ))))))))	Not Balanced