CSE225L – Data Structures and Algorithms Lab
Lab 04
Stack (array based)

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

stacktype.h

```
#ifndef STACKTYPE_H_INCLUDED
#define STACKTYPE_H_INCLUDED

const int MAX_ITEMS = 5;

class FullStack
// Exception class thrown
// by Push when stack is full.
{};
class EmptyStack
// Exception class thrown
// by Pop and Top when stack is emtpy.
{};

template <class ItemType>
class StackType
{
    public:
        StackType();
        bool IsFull();
        bool IsEmpty();
        void Push(ItemType);
        void Pop();
        ItemType Top();
    private:
        int top;
        ItemType  items[MAX_ITEMS];
};

#endif // STACKTYPE_H_INCLUDED
```

stacktype.cpp

```
#include "StackType.h"
template <class ItemType>
StackType<ItemType>::StackType()
{
    top = -1;
}
template <class ItemType>
bool StackType<ItemType>::IsEmpty()
{
    return (top == -1);
}
template <class ItemType>
bool StackType<ItemType>::IsFull()
{
    return (top ==  MAX_ITEMS-1);
}
template <class ItemType>
void StackType<ItemType>::Push(ItemType newItem)
{
    if( IsFull() ) throw FullStack();
    top++;
    items[top] = newItem;
}
template <class ItemType>
void StackType<ItemType>::Pop()
{
    if( IsEmpty() ) throw EmptyStack();
    top--;
}
template <class ItemType>
ItemType StackType<ItemType>::Top()
{
    if (IsEmpty()) throw EmptyStack();
    return items[top];
}
```

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 as input)		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 from the user as input and use a stack to check if the string of parentheses is balanced or not	( )	Balanced
	(( )) ( ) (( ) ( )) ( )	Balanced
	(( )) ( ) ((( )	Not balanced
	(( )))) ((( )	Not balanced