North South University
CSE-225L(Data Structures & Algorithm)
Summer - 2018
Lab-08 (Stack – Array Based)

Class "ItemType":

itemtype.h

```
#ifndef ITEMTYPE_H_INCLUDED
#define ITEMTYPE_H_INCLUDED
#include <iostream>
using namespace std;

const int MAX_ITEMS = 5;

class ItemType
{
    public:
        ItemType();
        void Print();
        void Initialize(int number);
    private:
        int value;
};

#endif
```

itemtype.cpp

```
#include "itemtype.h"

ItemType::ItemType()
{
    value = 0;
}

void ItemType::Initialize(int number)
{
    value = number;
}
void ItemType::Print()
{
    cout<<value<<" ";
}
```

<u>Class "StackType"</u>:

<u>stacktype.h</u>

```
#ifndef STACKTYPE_H_INCLUDED
#define STACKTYPE_H_INCLUDED
#include "itemtype.h"

class FullStack
{
    // Just a dummy class to be thrown as an exception object
};

class EmptyStack
{
     // Just another dummy class to be thrown as an exception object
};

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
```

<u>stacktype.cpp</u>

```
#include "stacktype.h"

StackType::StackType()
{
    top = -1;
}

bool StackType::IsEmpty()
{
    return (top == -1);
}
```

```
bool StackType::IsFull()
{
    return (top == MAX_ITEMS);
}

void StackType::Push(ItemType item)
{
    if(IsFull())
    {
        throw FullStack();
    }

    top++;
    items[top] = item;
}

void StackType::Pop()
{
    if(IsEmpty())
    {
        throw EmptyStack();
    }
    top--;
}

ItemType StackType::Top()
{
    return items[top];
}
```