North South University
CSE 225: Data Structure
Lab 09: Stack (Linked List Based)

```cpp
// StackType.h

#ifndef STACKTYPE_H_INCLUDED
#define STACKTYPE_H_INCLUDED
class FullStack
{};
class EmptyStack
{};
template <class ItemType>
class StackType
{
    struct NodeType
    {
        ItemType info;
        NodeType* next;
    };
public:
    StackType();
    ~StackType();
    void Push(ItemType);
    void Pop();
    ItemType Top();
    bool IsEmpty();
    bool IsFull();
    void ReplaceItem(int oldItem, int
newItem);
    void printlist();
private:
    NodeType* topPtr;
    int length;
};
#endif // STACKTYPE_H_INCLUDED
```

```cpp
//StackType.cpp
#include <iostream>
#include "StackType.h"

template class StackType<int>;

template <class ItemType>
StackType<ItemType>::StackType()
{
    topPtr = NULL;
}
template <class ItemType>
bool StackType<ItemType>::IsEmpty()
{
    return (topPtr == NULL);
}
template <class ItemType>
ItemType StackType<ItemType>::Top()
{
        return NULL;
}

template <class ItemType>
bool StackType<ItemType>::IsFull()
{
    NodeType* location;
    try
    {
        location = new NodeType;
        delete location;
        return false;
    }
    catch(std::bad_alloc& exception)
    {
        return true;
    }
}
template <class ItemType>
void
StackType<ItemType>::Push(ItemType
                          newItem)
{

}
template <class ItemType>
void StackType<ItemType>::Pop()
{

}
template <class ItemType>
StackType<ItemType>::~StackType()
{
    NodeType* tempPtr;
    while (topPtr != NULL)
    {
        tempPtr = topPtr;
        topPtr = topPtr->next;
        delete tempPtr;
    }
}
```

North South University
CSE 225: Data Structure
Stack (Linked List Based)

In this class, We are going to implement a stack data structure using array, and perform some operations.

Operation to Be Tested and Description of Action	**Input Values**	**Expected Output**
Create a stack		
Check if the stack is empty		Stack is Empty
Push four items	5 7 4 2	
Check if the stack is		Stack is not Empty
Check if the stack is full		Stack is not full
Print the values in the stack		2 4 7 5
Push another item	3	
Print the values in the stack		3 2 4 7 5
Check if the stack is full		Stack is not full
Pop two items		
Print top item		4

Add a function **ReplaceItem** to the StackType class which replaces all occurrences of **oldItem** with **newItem** in the Queue.

void ReplaceItem(int oldItem, int newItem);

Sample Input &Output:

Stack items:	**ReplaceItem(26, 9)**	Stack items:
21 26 13 26 29	$\longrightarrow$	21 9 13 9 29

Take home assignment:

Perform all above operations with std::stack.