COMP 220 **Data Structures**

Lab/Hwk 8/7

Assigned: November 01 *Due: November 05*

Abstract

In a previous lab you implemented a specialized `stack` class using `vectors`. Now you'll implement a better stack from scratch using a linked list.

Stacks with a Linked List

Use a linked list to implement a stack. Unlike our previous stack, this stack should be able to hold any integer (including negatives), and it should NOT have a fixed capacity. It should provide the following methods:

- A default constructor
- A destructor that recycles all heap memory used by the object
- `size()` — return the number of elements currently in the stack.
- `isEmpty()` — returns `true` if the stack is empty; `false` otherwise.
- `isFull()` — returns `true` if the stack is full; `false` otherwise.
- `top()` — Returns the value at the top of the stack without removing that value from the stack.
- `pop()` — Removes the element at the top of the stack (it does not return the element)..
- `push(int n)` — Add `n` to the stack.

Error cases should be handled by throwing an exception.

During lab you should first focus on writing the header file for this class (the `class` definition, which includes the instance variables and method declarations, plus documentation), the method stubs in the implementation file, and several tests that use your stack class. Only then should you work on the actual method implementations. Feel free to copy over your tests from the last `stack` lab (lab 7), though they will require some modifications.

Due Monday, 11/05 by noon. Submit the entire assignment as *hwk7*. Your header file, documentation, and tests will be graded as lab 8, while your implementation will be graded as homework 7.