**CS41 Programming Assignment #2**

**Total Point:** 100 **Due Date:** 03/16/2017, Thursday

**Assignment Description:**

1. Implement a **stack** class (ADT) using linked list data structure.
2. Write an application program (which evaluates post-fix expressions using stacks) that will test the **stack member functions** you implemented.

A **stack** is a sequential data structure in which all operations (push, pop, top, …) are performed at the same end (the **top** of stack). It is a *Last In, First Out (****LIFO****)* data structure, means entries are taken out of the stack in the reverse order of their insertion.

* File **stack1.h** contains the stack ADT documentation which specifies the member functions that you have to implement.
* While implementing each of the ADT’s member functions, you can use any of the functions in Linked-List Toolkit (the code is provided in files node1.h and node1.cpp).
* Define the stack class generically (using a value\_type), so that it can store the type of items of your choice.
* The items in the stack are stored in a linked list, with the **top** of the stack stored at the **head** node, down to the bottom of that stack at the **tail** node. The member variable **top\_ptr** (initially NULL) is the head pointer of the linked list. This link will be NULL if the stack is empty.

**Provided code:**

File **assignment2.zip** at the “Assignments” section of Blackboard contains:

* node1.h and node1.cpp - the node documentation, node class definition and Linked-List Toolkit functions i.
* stack1.h: the documentation of stack ADT, its member functions and the pre- and post-condition of each function.

**Tasks and Requirements:**

1. Complete the stack1.h to create a Stack class definition. You can change the namespace and value\_type aif necessary.
2. Implement the functions specified in the Stack class definition and store it in file stack1.cpp. Each member function can call any functions in the Linked-List Toolkit for its implementation.
3. Write an application program to **evaluate post-fix expressions** using the Stack functions you have implemented. This program must use the stack member function you implemented in Task 2
4. For testing, use the following expressions as the input to the application program. The program should print out error message when the input is an invalid post-fi expression.

**5 3 2 8 + 4 – 5 + 5 1 2 + 4 \* + 3 -** or expressions of your choices

1. Turn in the source code, input/output dialog of your test data, and result for grade.

The program should include the following information as comment on top of application program source code:

* Programming assignment number
* Name(s) of your programming team
* Brief description on what the program will do

**Grading:**

Create and run your program using IDE of your choice (CodeBlock, Visual C++, Netbean, Eclipse). Print the **source code, program input/output text, and a copy of the executable (.exe or jar file in USB or CD)** for grading. ***The turn in documents must be legible. Extremely tiny font or light/blur inks used in printing make it impossible to read and grade your assignments.***

**Pseudocode to evaluate a postfix expression:**

*Suppose P is an arithmetic expression in postfix notation. We will evaluate it using a stack to hold the operands.*

*Start with an empty stack. We scan P from left to right.*

*While (we have not reached the end of P)*

*If an operand is found*

*push it onto the stack.*

*End-If*

*If an operator is found*

*Pop the stack and call the value A*

*Pop the stack and call the value B*

*Evaluate B op A using the operator just found.*

*Push the resulting value onto the stack*

*End-If*

*End-While*

*Pop the stack (this is the final value)*

*Notes:*

*At the end, there should be only one element left on the stack.*

*This assumes the postfix expression is valid.*