CSE2122 (Spring 2020)
Homework 4, due 3/6/2020 11:59pm

Before you start, please login your stdlinux account and download the solution file and the template program from the class directory to your stdlinux directory. You can use the command "`cp /class/cse2122/hw4/* ./.`".

Submit your code to the designated dropbox on Carmen under the Assignments tab. Late submissions receive a 10% penalty for each day. No late submissions after the second day.

Skills needed to complete this assignment: linked lists, stacks.

Postfix notation is a mathematical notation in which operators follow their operands. For instance, to add 3 and 4, one would write 3 4 + rather than 3 + 4 (infix notation). If there are multiple operations, operators are given immediately after their second operands; so, the expression written 3 − 4 + 5 in conventional notation would be written 3 4 − 5 + in postfix notation: 4 is first subtracted from 3, then 5 is added to it. An advantage of postfix notation is that it removes the need for parentheses that are required by infix notation. While 3 − 4 × 5 can also be written 3 − (4 × 5), which is different from (3 − 4) × 5. In postfix notation, the former could be written 3 4 5 × −, while the latter could be written 3 4 − 5 × or 5 3 4 − ×. (from Wikipedia)

You can use a infix-to-postfix converter (like this one) to get more postfix notation examples.

A number stack can be used to evaluate expressions in postfix notations by following these steps:

- If a number is typed, push it on the stack.
- If an operation is typed, depending on the operation, pop off two numbers from the stack, perform the operation, and push the result back on the stack.

You should complete the `Stack` class and other functions in the template so that when the user enters an expression in postfix notation, the program calculates and shows the result. In this assignment the user enters one-digit positive numbers only, but the result can be any integer. Only basic operations (i.e. +, -, /, *) are entered by the user and integer division is used in calculations. User input ends with a semicolon.

Class `Stack` uses a linked list to implement a stack. It has five public member functions:

- Default constructor: initializes the stack to an empty stack
- Default destructor: frees the memory used by dynamic node objects in the stack
- `isEmpty`: returns true if the stack is empty and false otherwise
- `push`: pushes a new number to the top of stack
- `pop`: removes the top of stack
- `top`: returns the value at the top of stack, does not remove it from stack

You must use the template file and only add you code were you see `/*your code here*/`. Do not modify other parts of the code. Do not use C/C++ libraries other than `iostream` and `cstdlib`.

Input/Output example

Your output must match the solution file on Carmen exactly. Following are some example test cases and their expected outputs. User input is underlined.

```
Enter a postfix expression (ending with ; and press Enter):
3 4 - 5 +;
The result is: 4
```

```
Enter a postfix expression (ending with ; and press Enter):
3 4 - 5;
Not enough operators entered!
```

```
Enter a postfix expression (ending with ; and press Enter):
3 4 - 5 + *;
Not enough operands entered for operator: *
```

Program Submission

Compile your code with the following command. Submissions do not compile with the following command will receive zeros.

```
g++ <your file name> -std=c++11 -pedantic-errors
```

Make sure your code is formatted properly. Up to 5% of the assignment credit can be deducted if your program does not have appropriate indention and necessary comments. See textbook on section 2.5 or any code example in textbook/lecture slides.

Submit your code (the .cpp file) to the designated dropbox on Carmen. You can find the dropbox under the Assignments tab.