Assignment P7

Due Date: April 30

Purpose
You will use the stack abstract data type (ADT) in an almost real–life application. You can do this project individually or in pairs. If you complete the project with a partner, only one submission is necessary; be sure both names are on what you turn in.

Problem
With many scientific/graphing calculators, one can type in complicated calculations all on one line. The calculator computes the answer using normal mathematical precedence rules. You are to write a program that will simulate a calculator that can handle mathematical operators with floating point operands.

Input
The input will be a one line calculation such as the following:

\[ 5 + 3.3 \times 8 - 9 = \]

The expression is in infix order and may be any length. Valid operators are +, −, /, *, and ^ (exponentiation). Each number can be any floating point value. There are no spaces in between any individual items of the input. This makes reading the input easier – do not read the line in as a string and then convert to the constituent parts; just follow the pattern and read a float, followed by a char, followed by a float, etc. The equal sign signifies the end of the expression. NO parentheses may be input. You may assume that the input will be typed in correctly, so no error checking is necessary.

Output
The program should display, simply, the answer (!) of the calculation on the screen. The output should be displayed with 3 decimal places.

Specifics

- Computations must be done in order of precedence. For this assignment, ^ has the highest precedence, followed by *, / (same precedence), and finally +, −. Multiple operators with the same precedence are handled in left-to-right order.

- Stacks MUST be used to determine the answer. You must implement the stack using a linked-list (to get a little more pointer practice; do not use STL). Note, however, that this code is in the text.

- Several steps are necessary to solve this problem. First you must convert the input infix expression to postfix. This is also available in the text. Then find the solution to the postfix expression. One problem you may have to tackle is how to store the intermediate postfix expression. If you are really clever, you do not have to store the postfix expression at all.

Notes

- Although it may seem that you need multiple stack types, you should not implement more than one stack class. Instead, use templates and/or convert operators to numeric values (This is a hint for one possible solution!).
• You are free to use any additional data structures, including those available through STL (except the stack container).

• This problem is more involved than any previous project. You should work through an algorithm before you start coding. In any case, **DO NOT DELAY**.

• A good way to tackle this problem is to work on one part at a time. For example, make sure the infix to postfix portion works; this part should be straightforward. Then figure out how to store the postfix expression. Next, be able to compute the answer to a simple expression, like $4+7$. Add to your code incrementally to handle the other operators and the precedence.

• Related to the above, because this project is more complex than most, it will now be worth 9% of your grade rather than 7% as stated in the syllabus. The last project (P8) will be very short and will now be worth 5% of your grade.

• Submit your source code via email with the usual naming conventions. Hand in a hard copy version of your program in class on May 1st. Only one copy is necessary if you are part of a pair.

---

The answer is either $m$ or something else.

– Eva Ma, one of my grad school professors