CS2250 Fall 2014 Hauschild

PROGRAMMING ASSIGNMENT 5:
Due date: 12/11/2014

In this assignment, you will be using the STL stack and your own written dynamic stack to implement a calculator that calculates arithmetic expressions. The trick is, that this calculator will take as input an expression in prefix notation, otherwise known as polish notation, for the expressions.

Note: This assignment requires two different files implementing this project, one using the STL stack and another using a dynamic stack.

Definition:
In contrast to normal arithmetic notation, in prefix notation, operators are placed to the LEFT of their operands, not inbetween them. One of the advantages of this notation is that it removes the need for parentheses.

For example, the notation
(5 - 4) * 3
in polish notation is
* - 5 4 3

To evaluate, you can simply go through the string from right to left (reverse from reading direction) until you find an operator, then replace the operator and the two values to the right of the operator with the result. Continue doing this until you get the final result.

The expression
3 - (4 * 2)
in polish notation is
- 5 * 4 2

So what is our algorithm to implement this with a computer? Simply use a stack! This is again an example of how a stack can make some tasks much easier to deal with. Going from right to left, push the tokens onto the stack (something separated by a space). When you encounter an operator, pop the top two elements of the stack, perform that operation, and then push the result back onto the stack. Then continue on with the string. Once you have no more string left, the value in the top of the stack should be the result.

Task:
Write a program that takes a string from the user. This string should be a prefix expression, with the allowable operators +, -, * and /. The allowable operands are any number from 1-1000. Skip any blank space in the input string (dont push it on the stack or do anything with spaces). Spaces are the only delimiters necessary.

The program should then calculate the value of that prefix expression and output the result.

Details:
When going through your stack and evaluating any operands, I want you to display to the screen the results of subexpressions before you push them onto the stack. This is for my own grading purposes (and should help you with debugging!).

Another larger example:
− * / 15 − 7 + 1 1 3 + 2 + 1 1
is equivalent to
((15 / (7 − (1 + 1))) * 3) * (2 + (1 + 1)) = 5

Since the user is entering a string and you will be pushing various substrings onto the stack, you will need to do conversions from strings to integer values before pushing them onto the stack. I suggest using the following syntax to do so:
atoi(str.c_str())
You can truncate the result of any divisions.

Turnin details:
Ensure that you turn in two versions, one using the STL stack and another using a dynamic stack. Also a makefile is required if your executable for either of them takes more than one file to compile.