CS260 - Programming Assignment 3 - Stacks and expression processing

The purpose of this assignment is to give you some practice working with stacks as provided in the standard template library. And you'll get some practice converting expressions from infix to postfix and evaluating them. Super fun!

Overview

For this assignment you'll write a program that obtains an infix expression from the user, converts it to postfix, and then evaluates it.

Details

Your program should do the following:

- Prompt the user for an infix expression consisting of **single-digit numbers*** the operators +, -, * and /, and ( )
- Verify it doesn't contain any invalid characters and strip out the white space
- Print the expression out as infix
- Convert it to postfix and print it (if it's invalid, print an error message)
- Evaluate it and print the result
- repeat until user wants to quit

* You can do multi-digit numbers if you want. It doesn't change the conversion or evaluation at all, but does add some complexity to the parsing of the entered infix expression.

If the expression entered is invalid, the user should be notified.

A reminder: Functions should do one thing, do it well, and return the result. So each step above should be a separate function.

Example I/O:

```
Enter Expression: 5*2   +  3

Infix:  5*2+3
Postfix: 52*3+
Result:  13
```

```
Enter Expression: 3 + 5 * (7-2)

Infix:  3+5*(7-2)
Postfix: 3572-*+
Result:  28
```

Algorithm for converting infix to postfix (can be found all over the internet)*

```
For each item in the infix expression
    If it is a number
        append it to postfix string.
    else If it is a left parenthesis
        push it onto the stack
    else If it is an operator (+,-,*,/)
        If the stack is not empty
            while the operator on the top of the stack is >= precedence
                pop the operator and append it to postfix expression
        push operator onto the stack
    else If it is right parenthesis then
        while the stack is not empty and top not equal to left parenthesis
            Pop from stack and append to postfix expression
        pop out the left parenthesis and discard it
While stack is not empty
    pop and append to the postfix string.
```

* I think this algorithm is correct, but there are no guarantees. You might want to doublecheck it.

Algorithm for processing postfix expression
is given in book and on wikipedia.

STL stack

The C++ STL stack template is described here:
http://www.cplusplus.com/reference/stack/stack/

We only need to use it in its most simple form, an example of which is shown in pop:
http://www.cplusplus.com/reference/stack/stack/pop/ (except we'll use chars, not ints)

Very Important Stuff

Program should be well written, function properly, and be both easy and efficient to use.

All programs should follow the class's Coding Conventions
Submit the following:

- A zip file containing all your .cpp and .h files and your executable
- The executable as a separate file (so the .exe gets uploaded twice)