CS515 Assignment 2

The purpose of this assignment is to review the stack ADT and use of C++ STL (standard template library). You may start by downloading all starting files from `~cs515/public/2P` on agate.

You will finish writing a simple expression evaluator to evaluate infix arithmetic expressions. In this assignment, an expression is a sequence of tokens. A token can be one of the following:

- An operand, which is an integer number (without + or – sign).
- An operator: +, -, * or / (integer division),
- Parentheses which include three types of braces ( ), [ ] and { }.

Your program will repeatedly prompt the user to type in an expression from keyboard; the result will be output to the screen until the user types "`exit`" or control-d to terminate the input.

You can assume all tokens are valid, all numbers are integers and all braces are balanced. Spaces are allowed in the expression. Although all operands are positive, the expression may be evaluated to a negative result.

Below is a sample run from the command line prompt.

```
$ ./evaluation
Expression: 1+2
Answer: 3
Expression: 1* ( 2 + 3 )
Answer: 5
Expression: (6 - 2 )*10  + 2
Answer: 42
Expression: [(6 - 2)* 10  + 2 ]/(  4 + 2 )
Answer: 7
Expression: 10 -  [(6 - 2) * 10  + 2 ]  / ( 4 + 2 )
Answer: 3
Expression: 10 - [ ( 3 - 2 ) * 10 ] / 5
Answer: 8
Expression:           ← newline here
exit
```

Empty lines typed at the prompt are ignored. Sample test cases are provided in the starting files.

Requirements:

1. Code already exists that evaluates a given infix expression by first converting it to a postfix expression using the following algorithm:
   http://en.wikipedia.org/wiki/Shunting_yard_algorithm
   All you need to do is use the output of that code (which may be in the reverse order from what you need) to write the postfix evaluation algorithm as described here:
   http://en.wikipedia.org/wiki/Reverse_Polish_notation
2. You will need to compile your program on agate with `generateRPN.o` (how this is done is in the makefile). This object file will only work on agate and other Linux machines that are running the same version of the compiler.
3. You are allowed to use appropriate C++ STL containers in this class.
4. You may also use other libraries, such as `<cctype>` for alpha/digit check and `<cstdlib>` for alpha to digit/float conversion.
5. We may run this program (as with most programs) by redirecting input from a file, like this:

```
$ ./evaluation < input1
```

Submission:

- Submit your source files, but **not** your **`makefile.`**
- You should also fill out the README file and submit it as well. Submit the following files to the appropriate assignment on Mimir:
  `evaluation.cpp README`
- Do not turn in executables or object code. Programs that produce compile time errors or warnings will receive a zero mark (even if it might work perfectly on your home computer.)
- Be sure to provide comments in your program. You must include the information as the section of comments below:

```
/** CS515 Assignment 2
File: XXX.cpp
Name: XXX
Section: X
Date: XXX
Collaboration Declaration: assistance received from TA, PAC etc.
*/
```

Some notes on grading:

- Programs are graded for correctness (output results and code details), following directions, and using specified features, documentation, and style.
- Below is a tentative grading scheme.

directions	100
evaluation test run 1	10
evaluation test run 2	10
evaluation test run 3	10
evaluation test run 4	15
evaluation test run 5	15
evaluation test run 6	15
evaluation test run 7	15
evaluation test run 8	10