Programming Assignment #7

Match Brackets using a Stack

CS 2308.251 and 252, Spring 2015
Instructor: Jill Seaman

Due: before class **Friday, 5/1/2015** (upload electronic copy by 9:30am)

Problem:

Given a text file, your program will determine if all the parentheses, curly braces, and square brackets match, and are nested appropriately. Your program should work for mathematical formulas and most computer programs.

Your program should read in the characters from the file, but ignore all characters except for the following:
`{ } ( ) [ ]`

Your program should use the IntStack implementation (IntStack.h and IntStack.cpp, which will be available from the class webpage).

The general algorithm is to use the stack to store the opening unmatched brackets. When a closing bracket is encountered, check it against the one on top of the stack (pop it off)--make sure it matches. When you are finished there should be no unmatched brackets left on the stack.

Input/Output:

Your program must prompt the user to enter a filename. It should then try to open the file and then check it make sure the brackets all match appropriately. If they all match, the program should output a message saying so. If not, the program should output an appropriate error message.

There are three types of errors that can happen (and they can happen with any kind of bracket):

missing } : if you reach the end of the file, and there is an opening { that was never matched, like: `int main () { x[size]=10;`
expected } but found) : this is a wrong closing bracket, like: `{x[i]=10;)...`
unmatched } : this occurs if there is a closing bracket but not an opening bracket (not even one of the wrong kind), like: `int main () { x[i]=10; } }...`

NOTES:

- As soon as you encounter an error, your program should stop and print out the appropriate error message. **Do NOT try to keep going and find more errors!**

- The stack is an int stack but you will probably want to put characters on the stack. You can push a character directly on the int stack (it will automatically be converted to an int). When you pop I recommend popping into a char variable first (this will automatically convert the int back into a character).

- It might be easier to store the expected closing character on the stack when an opening bracket is encountered. This simplifies the matching when a closing bracket is encountered.

- Beware of stack underflow! Do NOT try to pop an empty stack! If you get an error message like this when you run your program: Assertion failed: (!isEmpty()) then you are popping an empty stack: your program should prevent this and output its own appropriate error message instead.

- Follow the style guidelines from the class website.

Logistics:

For this assignment you will have only one file to submit, the one containing your main function. Name your file **assign7_xxxxxx.cpp** (where xxxxx is your NetID)..

Submit an electronic copy using the Assignments tool in TRACS before the deadline.

For this assignment, no printout is required! The feedback cover sheet will be posted to the assignments tool in TRACS (or emailed to you) after the assignments are graded (and the grade will be posted to the TRACS gradebook2 tool).