Programming Assignment #7

Match Brackets using a Stack

CS 2308.251, 252, and 257 Spring 2016
Instructor: Jill Seaman

Due: Wednesday, 4/27/2016: upload electronic copy by 9:00am

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: `{ } ( ) [ ]`

The general algorithm is to use a 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.

Your program should first implement a char stack. **CharStack.h** is provided on the website. You must supply the **CharStack.cpp** file that includes the implementations of the functions in the class declaration. Note that the stack elements will be stored in a string, and no variable named top is necessary. You can complete this assignment using these string functions: at(int), size(), append() or +=, and substr(int, int).
Note: Do NOT use these functions: push_back, pop_back, or back.

Input/Output:

Your Driver 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:

- Your program must compile and run, otherwise you will receive a score of 0.
- There is NO Test Case 0 for this assignment.
- 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!**
- 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! Your pop function should include a check that will abort the program if the driver attempts to pop an empty stack. Your driver program should avoid attempting to pop an empty stack. In other words when I run your driver program, it should **never** abort/crash.
- Follow the style guidelines from the class website.

Logistics:

For this assignment you need to submit two files: CharStack.cpp and your driver. Please zip these into one file, **assign7_xxxxxx.zip** (where xxxxx is your NetID). Then submit the zip file

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).