Computer Science 250

Project 2: Balanced Parenthesis

**Due:** Friday, Mar. 29, at the beginning of class

Write a program that determines if the input contains balanced parentheses. The input will consist of any characters, but only the characters representing parentheses are important. These characters are parentheses: '(', ')', '{', '}', '[', ']', '<', '>'.

For our purposes, a set of parentheses is balanced if every open parenthesis comes before its companion close, and only other strings of balanced parenthesis separate the two characters. The following string contain balanced parentheses:

* (){}
* ({})
* [<>{()}[<()>]]

These strings do not contain balanced parentheses:

* )( Close comes before open
* [{} Missing close parenthesis
* (<()>{)} No matching parenthesis for '{' inside the outer "()"

The algorithm to check for balanced parentheses uses a stack. When an open parenthesis is seen, it is pushed on the stack. When a close parenthesis is seen, the stack is popped and the result is compared against the close. If the two parentheses do not match, the parentheses are not balanced. The stack must be empty at the end of the input.

Write a program that reads input from the user and determines if the parentheses in the input are matched. Ignore all other characters in the input.

Here is an example of what you program must look like when it is run. User input is shown in bold:

Enter some text:

**(value + [time - {time \* time} a - g](f([m], t) + < c + time – a / c > - value / time + country ));**

Parenthesis match.

Enter some text:

**<**

Parentheses do not match.

Structure your program in three modules:

1. A stack module that contains the data and function definitions necessary to implement the stack. All stack operations should be performed by this module.
2. An input module that contains the definitions necessary to handle the input. At a minimum, it should include a function that reads input from the user and returns the next parenthesis character in the input. Have this function return EOF if there are no more parenthesis in the input.
3. A main module that implements the main logic of your algorithm. The main module must be called "balanced.c".

Be sure to properly hide any external variables and functions that are defined within a module but which should not be visible in other modules.

In addition to the source code, you must include a Makefile that contains rules to build each module, a rule to build the entire program, and rules for all and clean.

**What to turn in**: When you are ready to turn in your program, create a tar file that contains your Makefile, along with all of the source required to build your program. Email me a copy of the tarball as an email attachment. Make sure the subject of your email is "CS 250 – Project 2 - <your last name, your first name>". Both the hard copy and email must be received by the due date to be considered an on-time submission.