Simple Arrays — xy Balance

For this assignment, you will implement a C function that determines whether a C-style string (i.e., null-terminated `char` array) is "balanced", according to the rules given below. Your solution must conform to the following interface:

```
//////////////////////////////////////////////////////////////////////
//  We'll say that a String is xy-balanced if for all the 'x' chars in the
//  string, there is a 'y' char somewhere later in the string. So "xxy" is
//  balanced, but "xyx" is not.
//
//  Return true if the given string is xy-balanced.
//
//  xyBalance("aaxbby") --> true
//  xyBalance("aaxbb")  --> false
//  xyBalance("yaaxbb") --> false
//
bool xyBalance( const char *String ) {

   . . .
}
```

You will submit a single `.c` file, containing nothing but the implementation of the specified function and any `include` directives that your solution needs.

Your submission will be compiled with a test driver and graded according to how many cases your solution handles correctly.

You will be allowed up to ten submissions for this assignment. Use them wisely. Test your program thoroughly before submitting it. Make sure that your program produces correct results for every test case you can think of. If you do not get a perfect score, analyze the problem carefully and test your fix with the test data shown in the Curator grade report, before submitting again. The highest score you achieve will be counted.

The *Student Guide* and other pertinent information, such as the link to the proper submit page, can be found at:

http://www.cs.vt.edu/curator/

Pledge:

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the submitted file:

```
//    On my honor:
//
//    - I have not discussed the C++ language code in my program with
//      anyone other than my instructor or the teaching assistants
//      assigned to this course.
//
//    - I have not used C++ language code obtained from another student,
//      or any other unauthorized source, either modified or unmodified.
//
//    - If any C++ language code or documentation used in my program
//      was obtained from another source, such as a text book or course
//      notes, that has been clearly noted with a proper citation in
//      the comments of my program.
//
//    - I have not designed this program in such a way as to defeat or
//      interfere with the normal operation of the Curator System.
//
//    <Student Name>
```

Failure to include this pledge in a submission is a violation of the Honor Code.

The program source code:

```
// Project 1 for CS 1044, Summer I 2009
// Programmer:     <your name here>
// Last modified:  <time/date of last change that you made>
//
// Purpose
//   This program reads a sequence of temperature and wind speed observations,
//   adjusts the reported temperature for wind chill, and produces a report
//   file including the adjusted temperature and the amount of adjustment.
//
#include <iostream>   // for cout, if needed for debugging
#include <fstream>    // for file stream support
#include <iomanip>    // for formatting support
#include <cmath>      // for standard math functions, if needed
#include <string>     // for character string type
using namespace std;  // put all of that in scope

int main() {

   string obsTime;               // (read) time of observation
   int    obsTemp;               // (read) observed temperature
   int    obsSpeed;              // (read) observed wind speed
   double wcTemp;                // (computed) temperature adjusted for wind
   double wcTempSum =   0.0;     // (computed) sum of adjusted temperatures
   double minWCTemp = 150.0;     // (computed) minimum adjusted temperature
   int    numObservations = 0;   // (computed) number of observations

   ifstream Data("WCData.txt");           // open input file
   ofstream Log("WindChillReport.txt");   // open output file
   Log << fixed << showpoint;             // prepare output stream for decimal
                                             // output

   Log << "Time     WC temp     WC Effect" << endl;   // write output header
   Log << "-----------------------------" << endl;

   Data.ignore(INT_MAX, '\n');         // skip over first two lines of input
   Data.ignore(INT_MAX, '\n');

   Data >> obsTime >> obsTemp >> obsSpeed; // read first line of observed data

   while ( Data ) {

      // Add C++ code to perform necessary computations on input data here:

      // Do not modify any of the rest of the code!

      Log << obsTime;                                  // write data table
      Log << setw( 8) << setprecision(1) << wcTemp;
      Log << setw(14) << setprecision(1) << wcTemp - obsTemp;
      Log << endl;

      Data >> obsTime >> obsTemp >> obsSpeed;  // try to read another line
                                                  // of data
   }

   Log << "-----------------------------" << endl;  // delimit end of table
   Log << endl;
```

```
   // perform additional computations:


   // write final line of report
   Log << "The average adjusted temperature, based on " << numObservations
      << " observations, was " << setprecision(1) << avgWCTemp << endl;

   Data.close();   // close the file streams
   Log.close();
   return 0;
}
```