CS515 Assignment 1

The purpose of this assignment is to help you get practice with basic C++ programming. You will work with C++ stream input/output and C++ `string` library type. You do not need to create any classes nor any functions other than a main function. Download all starter files from `~cs515/public/1P` on agate or from the corresponding assignment on Mimir. (If you don't have an account on agate, please email CS support staff immediately support@cs.unh.edu)

Problem Description

Suppose a temperature sensor collects data once per minute. The sensor data could be weather data, body temperature, etc. The data are real numbers (floating-point values), in the range of [-100, 100]. A *high temperature span* for minute *n* is the number of minutes prior to minute *n* such that the temperature values for those minutes were are all lower than or equal to the value at minute *n*.

For example, 10 temperature entries are shown; values start from minute 0, 1 and so on.

37.8	38.9	39.1	40.5	37.9	38.8	40.5	39.0	36.9	39.8
0	1	2	3	4	5	6	7	8	9

At minute 2, the corresponding data is 39.1 and it is a 3-minute high. At minute 6, the corresponding data is 40.5 and it is a 7-minute high.

You are asked to write a program called **`calSpan`** to read from a file that stores sensor data, and to compute spans based on input given by the user—the user will indicate a minute of interest and the program will tell the user how long the span is for that minute. All sensor data are stored as space-delimited ASCII strings in the data files. Users type in queries at keyboard to find out the data span at any given minute. The program provides answers in a format like this:

```
Data at minute 0 is a 1-minute(s) high.
```

The program should prompt the user to input the index of a minute by displaying "`Which minute to query? `" The program should support repetitive user query until the user inputs the word `exit` (case insensitive) with a newline, at which point the program terminates. A sample run:

```
$ calSpan data0
Which minute to query? 6
Data at minute 6 is a 7-minute(s) high.
Which minute to query? 4
Data at minute 4 is a 1-minute(s) high.
Which minute to query? 8
Data at minute 8 is a 1-minute(s) high.
Which minute to query? 9
Data at minute 9 is a 3-minute(s) high.
Which minute to query? Exit
```

The data file name is passed as a command-line argument to the program. Your program should handle file-open errors and having the wrong number of command-line arguments (too many or too few). In these cases, your program should give the appropriate error message, from those shown below:

```
   Wrong number of arguments
   Error opening file
```

The sensor data could be also be corrupted. For example, the data may not be within the valid range, or contain non-numerical values. When the user queries a minute whose sensor data is corrupted, the program should report, "`Data at minute X is corrupted.`" Meanwhile, all corrupted data are treated as a minimum value. For example, given another file with corrupted data as below:

37.8	38.a	139.1	abc.5	37.9	38.8	40.5	39.0	36.9	39.8
0	1	2	3	4	5	6	7	8	9

Query at minute 1, 2, and 3 will all return "`Data at minute X is corrupted.`" (where X is 1, 2, or 3). Query at minute 4, returns "`Data at minute 4 is a 5-minute(s) high`".

The program also needs to handle user input errors. If the user queries a minute-index that does not exist in the file, the program should produce the error message, "`Query minute out of range.`" If user types in a string (other than "exit") that contains non-numeric values (like 11a) or non-integer values (like 15.2), the program should produce an error message "`Wrong query input.`"

Notes:

- You may open the data file only once and the file stream must be closed before the program starts accepting user queries. This means the program should not read again from the file while processing user queries. Use `ifstream` to process the file. E.g.
  ```
  ifstream f("input");
  f >> x;
  ```
- There are no specified limits for the number of data entries stored in the file. You need to decide the proper data structure to store the data. You may use C++ STL containers.
- `stringstream` objects are handy to help parse the data and find invalid formats. See http://www.cplusplus.com/reference/sstream/stringstream/
- The starter files include a batch test script and two simple input data files. `batch.run` contains the test cases and the expected output is given in `batch.run.output`.

Submission:

- Submit your source files and your **`makefile`**. The makefile **must** be able to make a target named **`calSpan`** when the `make` command is called without any arguments.
- You should also fill out the README file and submit it as well. Submit the following files to the appropriate assignment on Mimir:
  `calSpan.cpp makefile 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 for your program. You must include the information as the section of comments below:
  ```
  /**  CS515 Assignment 1
  File: XXX.cpp
  Name: XXX
  Section: X
  Date: XXX
  Collaboration Declaration:
      Assistance received from TA, PAC and online resources etc.
              …
    */
  ```

Some notes on grading:

- Programs are graded for correctness, following directions, and using specified features.
- A tentative grading scheme is given at right ➔

testcase 1	15
testcase 2	15
testcase 3	20
testcase 4	15
testcase 5	15
testcase 6 (command line args)	5
testcase 7 (user error inputs)	10
testcase 8 (user error inputs)	5