Arrays: Tracking Precipitation Amounts

NOTE: For this assignment, there are additional requirements you may be graded upon in addition to producing the correct output. They are: (1) you should write an outline of your program design that reflects your top-down functional/modular decomposition – this should be included in your comments at the beginning of your program, (2) your program should contain good documentation, (3) you must write and use four additional functions in your program. The GTAs will HAND-CHECK your code on this assignment to grade you on these items. You should be sure to read and apply the documentation standards as referenced in the Programming Standards section of this specification.

The National Climatic Data Center has hired to you write a report generating program for daily precipitation amounts. Normally, this task would be straightforward, but a computer glitch has caused some of the data to appear out of order, to be lost, or to have the recorded date corrupted. You are responsible for storing as much valid data as possible, recognizing errors in the data file, and displaying a summary of daily precipitation amounts for the specified month.

You can find out more about the National Climatic Data Center and find real online precipitation data at http://www.ncdc.noaa.gov/ol/ncdc.html

The input file:

The input file for this program is named "`Precip.txt`". The first line contains human readable information that is not needed by the program. The second line contains the location of the weather station reporting precipitation. This line will always contain a newline immediately following the location. The third line contains the full month name, which is capitalized, followed immediately by a comma, and then the year for recorded information. The fourth and subsequent lines contain the day for recorded information and the amount of precipitation in inches reported on that day, which are separated by whitespace characters (spaces or tabs).

```
CS1044 Project 5 Fall 2008
Blacksburg, VA
December, 2006
2  0
4 0
5 0
26 0
6 0
8 0.01
9 0
10  0.02
13 0
114 0
15 0
17 0.55
19     1.8
8 0.08
20 4.12
24 0
32 0.73
28 0
11    0.05
12 0.01
29 0
```

Calculations and error checking:

Your program will read and store the precipitation information in an array. The array will have to be large enough to store information for any month, although the entire array might not be used. You should initialize the entries in the array to some well-known, but impossible precipitation value so that you can readily identify when data is unavailable. Once all of the precipitation data has been read, you will need to calculate the minimum, maximum, and average precipitation values.

The input file may contain two kinds of errors. The first is that the day given in the input might be invalid (too small or large). The second is that a file might contain multiple entries for the same day. These errors should be recognized and an error message displayed, including the line number where the error occurred. See the sample output file below for the exact format of the error messages.

You will need to determine if a year is a leap year to determine the number of days properly. A year divisible by 4 but not 100 is generally a leap year. Centuries divisible by 400 are also leap years, (as was the case with the year 2000).

The output file:

The output file is named "`Report.txt`". An output file, which corresponds to the given input file, is shown below. The first two lines contain programmer and project information. The third line is blank. The fourth line contains the location, month, and year of the precipitation data. The fifth line is blank. Remaining information is printed out as follows:

- Any errors detected in the input file. If errors are detected, a blank line is printed after the last error.
- A histogram for the precipitation amounts. The day and precipitation amount to two digits of precision are printed. Following the day and amount, a graph containing one star for each 0.25 inches or part thereof is displayed. For example, 0.01-0.25 inches will display one star, 0.26-0.50 two stars, 0.51-0.75 three stars, etc. If no precipitation data is available for a day, `NA` should be printed for the amount with no stars following. A blank line is printed after the graph.
- The maximum, minimum, and average precipitation amounts are printed. If no information is available for the entire month, `NA` should be printed for all three values.

```
Programmer: Dwight Barnette
CS 1044 Project 5 Fall 2008

Precipitation report for Blacksburg, VA during December, 2006

Error         Day       Line
Invalid       114         13
Repeated        8         17
Invalid        32         20

Day Amount Graph
  1     NA
  2   0.00
  3     NA
  4   0.00
  5   0.00
  6   0.00
  7     NA
  8   0.01 *
  9   0.00
 10   0.02 *
 11   0.05 *
 12   0.01 *
 13   0.00
 14     NA
 15   0.00
 16     NA
 17   0.55 ***
 18     NA
 19   1.80 ********
 20   4.12 *****************
 21     NA
 22     NA
 23     NA
 24   0.00
 25     NA
 26   0.00
 27     NA
```

```
 28   0.00
 29   0.00
 30     NA
 31     NA

Minimum Precipitation: 0.00
Maximum Precipitation: 4.12
Average Precipitation: 0.21
```

Documentation and other requirements:

You must meet the following requirements (in addition to designing and implementing a program that merely produces correct output):

- Your program, at a minimum, must contain 4 functions in addition to main().
- You must make appropriate use of C++ parameter-passing mechanisms.
- At least one of your functions must be a value-returning function.
- You must include a top-down design of your program in comments following your pledge statement.
- Document each function with a function documentation header as described in the notes
- Define your functions after your main program
- Use function prototypes appropriately
- Write a header comment with your identification information, the required pledge statement (below), and a brief description of what the program does.
- Write a comment explaining the purpose of every variable and named constant you use.
- Write comments describing what most of the statements in your program do.
- Use descriptive identifiers for variables and for constants.
- Use named constants instead of "magic values" whenever it is appropriate.
- Do **NOT** use `break`, `continue`, `return`, or `goto` statements to exit a loop prematurely.
- Do **NOT** use global variables. Global constants are OK.

Testing:

Obviously, you should be certain that your program produces the output given above when you use the given input file. However, verifying that your program produces correct results on a single test case does not constitute a satisfactory testing regimen. You should test your program on all the posted input/output examples given along with this specification. At this point, you're ready to submit your solution to the auto-grader.

Pledge:

As with the previous projects, each of your submissions to the Curator must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the header comment for your program. Failure to include this pledge in a submission is a violation of the Honor Code.

```
//    On my honor:
//
//    - I have not discussed the C++ language code in my program with
//      anyone other than my instructor or the teaching assistants.
//
//    - 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 Automated Grader.
//
// Pledge: On my honor, I have neither given nor received unauthorized
// aid on this assignment.
//
// <student's name>
```

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