

**Arrays & Functions:****Tracking Precipitation**

The National Climatic Data Center has hired you to write a program to fix a major mess with one of their precipitation report generating systems. Normally this would be straightforward, but a computer glitch has caused some of the data to appear out of order, to be missing, or to have a corrupted date. 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 in 2003.

**Sample input data:**

Here is a sample input file, named `Precip.txt`, for the program

```
CS1044 Project 4 Summer II 2003
Blacksburg, VA
December, 2003
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
```

The first line contains human readable information that is not needed by the program. The second line contains the location of the weather reporting station. 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 2003 for recorded information. Note that 2003 is not a leap year. Each of the remaining lines contains two pieces of data for a single date in the month, separated by whitespace (either spaces or tabs):

- date                                    an integer that may be valid or invalid for the specified month
- precipitation                        the amount of precipitation recorded on this date

There is no specified limit on the number of data lines, so your program must be designed to detect when it's out of input and stop automatically.

**Calculations and error checking:**

Your program will read and store the precipitation information in an array. The array must be large enough to store information for any month, although the entire array may not always be used. Remember to initialize the array values to some impossible precipitation value so that you can tell if a given day has valid data. Once all of the input data has been read in, 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 too large). The second is that a file might contain multiple entries for the same day. These errors should be recognized, and an error message should be displayed, including the line number where the error occurred. See the sample output file below for the exact format of the error messages.

### Sample output:

Here is a sample output file, named `Report.txt`, which corresponds to the input data given above:

```
Programmer: Rich Wheaton
CS 1044 Project 4 Summer II 2003

Precipitation report for Blacksburg, VA during December, 2003

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      Maximum      Average
 0.00         4.12         0.21
```

The first two lines just identify the programmer and project, and the third line is a blank. The fourth line contains the location, month, and year of the precipitation data. The fifth line is blank. Remaining information is printed as follows:

- If any errors were found in the input data, then a line of header information is printed for the error output. The errors should be listed in the same order as they were discovered in the input file. For each error, list the type of error (either "Invalid" or "Repeated"), the Day value listed in the input file, and what line number in the input file the error was detected. After the errors, a blank line should be output.
- The next line contains the headers for the precipitation histogram. For each day, the date number should be listed, followed by the amount of precipitation for that day. If no precipitation is available for a day, then NA should be printed for that day. Following the day and amount, a graph containing one star for each .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. If no precipitation was detected on a given day, or if no data is available, then no stars should be displayed. After the histogram graph for the whole month, there should be a blank line.
- The maximum, minimum, and average precipitation amounts are printed. **The average is the sum of the valid precipitation values divided by the number of days in the month.** If no information is available for the entire, NA should be printed for all three values.

If you have read the description of how the Curator scores your program in the *Student Guide*, you know that is important that you use the same spelling and capitalization for all the labels shown above. The horizontal spacing does not effect scoring unless you combine things that should be separate or separate things that should be combined. You are free to experiment with the horizontal layout but you should try to align the columns neatly.

### Other Requirements:

- Your program, at a minimum must contain 5 functions, in addition to main().
- At least one of these functions must return a non-trivial value.
- All functions need a header comment in addition to any necessary in-line comments.
- Your basic data structure for the precipitation data must be an array.

### Evaluation:

Everything that you have been told about testing in class applies here. Do not waste submissions to the Curator in testing your program! There is no point in submitting your program until you have verified that it produces correct results on the sample data files that are provided. If you waste all of your submissions because you have not tested your program adequately then you will receive a low score on this assignment. You will not be given extra submissions.

Your submitted program will be assigned a score, out of 200, based upon the runtime testing performed by the Curator System. We will also be evaluating your submission of this program for documentation style and a few good coding practices. This will result in a deduction (ideally zero) that will be applied to your score from the Curator to yield your final score for this project.

Read the *Programming Standards* page on the CS 1044 website for general guidelines. You should comment your code in the same manner as the code given for the first two programming assignments. In particular:

- You should have a header comment identifying yourself, and describing what the program does.
- Every constant and variable you declare should have a comment explaining its logical significance in the program.
- Every major block of code should have a comment describing its purpose.
- Adopt a consistent indentation style and stick to it.

Your implementation must also meet the following requirements:

- Choose descriptive identifiers when you declare a variable or constant. Avoid choosing identifiers that are entirely lower-case.
- Use named constants instead of literal constants when the constant has logical significance.
- Use C++ streams for input and output, not C-style constructs.
- Use C++ string variables to hold character data, not C-style character pointers or arrays.

- Note: you are explicitly forbidden to write any user-defined functions for this program. This may make the program somewhat repetitive, and physically longer than the alternative. To some extent, that's the reason for this restriction.

Understand that the list of requirements here is not a complete repetition of the *Programming Standards* page on the course website. It is possible that requirements listed there will be applied, even if they are not listed here.

### Submitting your program:

You will submit this assignment to the Curator System (read the *Student Guide*), and it will be graded automatically. Instructions for submitting, and a description of how the grading is done, are contained in the *Student Guide*.

You will be allowed up to five submissions for this assignment. Use them wisely. Test your program thoroughly before submitting it. Make sure that your program produces correct results for every sample input file posted on the course website. If you do not get a perfect score, analyze the problem carefully and test your fix with the input file returned as part of the Curator e-mail message, before submitting again. The highest score you achieve will be counted.

The *Student Guide* and submission link 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 header comment for your program:

```
// 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.**