CPSC1012 Assignment 3 – Methods, Arrays, and File I/O

Weight: 15% of your final mark Due: July 14 by 11:59pm

Submission Requirements: Submit a copy of your Visual Studio 2022 project folder (named Assignment03-*FirstInitialLastName* that implements the requirements specified in this document) on or before the due date. **This must be done in the supplied assignment repository**, do not create your own repository for this assignment. It is your responsibility to ensure that your work is in the repository. Prior to the due date, it is **recommended** that you set up a time with your instructor to demonstrate the working functionality of your program (remote classes can demonstrate via MS Teams). **Work not in the repository will not be marked**. **Late submissions will not be graded**.

Program Specifications

A researcher is tracking the pH levels on a local riverbed. There has been a lot mining activity in the area and local authorities are concerned about the acidity levels in the water. For reference, pH is the measure of how acidic or basic water is on a scale of 0 to 14. Typically, water has a pH between 6.5 and 8.5 on this scale, with 7.0 being neutral. For rivers, the optimum pH is around 7.4. Every day at 0630 hours, the researcher tests the pH levels of the river and records the results. A program is required to allow the researcher to enter their results and perform some simple data analysis on the numbers.

Your program must meet the following requirements:

- Must be able to enter daily pH values
- Must be able to save daily pH values to a log file
- Must be able to load previous monthly pH log files
- Must be able to view and edit previously entered pH values (current or previous monthly data)
- Must be able to view simple analysis of the currently loaded log file:
    - Mean average daily pH
    - Median of daily pH
    - Highest daily pH
    - Lowest daily pH
    - Chart of daily pH

Implementation Details

Use a main menu for top-level options and a sub-menu for the analysis options in your solution. The program should continue to run until the user chooses to quit the program.

Ask the user to supply the desired filename when saving a new monthly log. Assume that the user will enter new log values for an entire month (how you input dates and pH values is up to you, but ensure that the date values are valid MM-dd-yyyy format and that pH values are between 0 and 14).

Use two associative arrays for storing the data in your program (one for date values and one for corresponding pH values). Keep an accurate record count for the number of days of data that have been loaded/entered.

Ensure that duplicate entry dates (when entering data) are not allowed.

The format of the pH log files should be as follows (assume valid file format for input):

- Include a header record with the following headings: `Date` and `pH Level`
- pH results are recorded to one decimal place
- Data files must include the date in MM-dd-yyyy format and be ordered in ascending date order:

```
Date,pH Level
05-01-2020,7.9
05-02-2020,8.1
05-03-2020,7.8
. . .
```

Sample log file format

You must use a modular approach when constructing this program. Ensure that, at a minimum, the following methods are present and used (difficulty level is rated 1-easy, 2-moderate, 3-difficult, 4-extreme):

- void DisplayMainMenu() → displays the main menu options [difficulty 1]
- void DisplayAnalysisMenu() → displays the analysis menu options [difficulty 1]
- string Prompt(string promptString) → displays the prompt string and returns user-entered string (allow empty string to be returned) [difficulty 1]
- double PromptDouble(string promptString) → displays the prompt string and returns user-entered double (ensure that the program does not crash and always returns a valid double value) [difficulty 1]
- int EnterLogEntries(double[] phValues, string[] dates) → allows the user to enter log entries (dates and pH readings) into the arrays; returns the number of entries entered [difficulty 2]
- int LoadLogFile(double[] pHValues, string[] dates) → loads the records from a log file into the associative arrays used by the program; returns the record count (i.e. how many days of data were loaded) [difficulty 2]
- void SaveLogFile(string filename, double[] pHValues, string[] dates, int count) → writes the associative array data to a file (filename) in the correct format [difficulty 2]

- void DisplayEntries(double[] pHValues, string[] dates, int count) → displays the current log entries in a formatted table (i.e. ensure that proper columns and alignment are used). You must use a ***for*** loop to loop through the arrays and produce the display [difficulty 2]

- void EditEntries(double[] pHValues, string[] dates, int count) → allows the user to view all current entries and choose one to edit (i.e. overwrite) [difficulty 3]

- void DisplayChart(string filename, double[] pHValues, string[] dates, int count) → displays a chart of the pH log data in the following format:

  ```
   pH
  ----
  7.9|                         *  *  *
  7.8|                *
  7.7|             *     *  *
  7.6|    *                                   *
  7.5| *        *
  7.4|       *
  -------------------------------------     ---
  Day|01 02 03 04 05 06 07 08 09 10 11  ...  30
  ```

  Use the maximum and minimum values in the pH log to determine the y-axis and the record count (i.e. number of log entries) for the x-axis. In the chart above, the ellipsis (...) are used for demonstration only, include all days in your implementation. [difficulty 3]

- double Median(double[] nums, int count) → returns the median average from the array of numbers (requires that the original ordering of the arrays be retained) [difficulty 3]

- **NOTE**: Although you may perform these operations directly in your main method, you may also want to create and use methods to return the mean average, return the highest pH, and return the lowest pH.

The program should never crash and must deal with errors gracefully.

Aside from what's been presented in this document, do not make any assumptions. Seek clarity from your instructor if you do not understand something in this document.

Stretch Goals **[the following are not necessary, but nice touches]**

Ensure that duplicate dates are not allowed when entering new log records (how you do this is up to you).

Only display the menu when the user chooses to view it (i.e. display the menu once, and then only again if the user enters some type of 'View Main Menu' option).

Only allow display of entries, analysis options, edit entries, if there are records that have been entered or loaded (i.e. there need to be records present in order to run certain program functions).

pH Log Sample Run

A test file (**spring-2023-ph.csv**) with only 10 records has been used to produce the following sample output; the file is provided for you to test with (place this file in the default run location to avoid having to enter the full path to the file).

Display of Entries:

```
Current Log entries
===================

Date       pH Value
---------- --------
05-01-2020      7.7
05-02-2020      7.9
05-03-2020      8.0
05-04-2020      7.6
05-05-2020      7.8
05-06-2020      8.2
05-07-2020      7.9
05-08-2020      7.7
05-09-2020      7.7
05-10-2020      7.6
```

Chart of Entries:

```
Chart of Entry Data
===================
pH
----
 8.2|                 *
 8.1|
 8.0|        *
 7.9|     *              *
 7.8|              *
 7.7|  *                    *  *
 7.6|           *                 *
-----------------------------------
Day | 01 02 03 04 05 06 07 08 09 10
```

Mean Average pH:

```
Mean average pH: 7.8
```

Highest pH:

```
Highest pH: 8.2
```

Lowest pH:

```
Lowest pH: 7.6
```

Median pH:

```
Median pH: 7.8
```

A sample GIF of the above run through has been provided along with these specifications.

Coding Requirements

- A C# comment block at the beginning of the source file describing the purpose, input, process, output, author, last modified date of the program
- Write only one statement per line.
- Use camelCase for local variable names.
- Use ALL_CAPITALS for any constant variable names
- Use defensive programming where necessary
- Ensure graceful handling of exceptions

Evaluation [30 marks total]

Marking Rubric [Program Completion and Modularization]

Mark	Description
13-15	Excellent – program implements, at a minimum, four difficulty 1 methods, four difficulty 2 methods, 2 difficulty 3 methods (total 10). Methods are correctly implemented, including proper documentation comments, and are called appropriately in the main program. Functionality of each method is correct and each task is fully implemented (e.g. returns expected values, proper file read/write, proper array management, etc.).
10-12	Very Good – program implements, at a minimum, four difficulty 1 methods, three difficulty 2 methods, 1 difficulty 3 method (total 8). Methods are correctly implemented, including proper documentation comments, and are called appropriately in the main program. Functionality of each method is correct and each task is fully implemented (e.g. returns expected values, proper file read/write, proper array management, etc.).
7-9	Acceptable – program implements, at a minimum, four difficulty 1 methods and three difficulty 2 methods (total 7). Methods are correctly implemented, including proper documentation comments, and are called appropriately in the main program. Functionality of each method is correct and each task is fully implemented (e.g. returns expected values, proper file read/write, proper array management, etc.).
4-6	Needs Work – program implements, at a minimum, three difficulty 1 methods and two difficulty 2 methods (total 5). Methods are correctly implemented, including proper documentation comments, and are called appropriately in the main program. Functionality of each method is correct and each task is fully implemented (e.g. returns expected values, proper file read/write, proper array management, etc.).
1-3	Unsatisfactory – program implements, at a minimum, two difficulty 1 methods and 1 difficulty 2 method (total 3). Methods are correctly implemented, including proper documentation comments, and are called appropriately in the main program. Functionality of each method is correct and each task is fully implemented (e.g. returns expected values, proper file read/write, proper array management, etc.).
0	Not done; poorly attempted; program implements none (0) of the required methods.

Marking Rubric [Array Implementation]

Mark	Description
5	Excellent – program implements and correctly uses both associative arrays; correct element count is kept; arrays are correctly loaded and traversed for output, including analysis algorithms (e.g. finding the median value)
4	Very Good – program implements and correctly uses both associative arrays; element count is always accurate; arrays are correctly loaded and traversed for output; analysis algorithm may contain a minor error
3	Acceptable – program implements and correctly uses both associative arrays; element count is accurate in most cases; arrays are correctly loaded and traversed for output in most cases; analysis algorithm may contain minor errors
2	Needs Work – program implements and uses both associative arrays; element count is not accurate; program suffers from minor error(s) when loading or traversing arrays for output; analysis algorithm not implemented or contains major error(s)
1	Unsatisfactory – program implements and uses only one array; element count is not accurate; program suffers from major error(s) when loading or traversing arrays for output; analysis algorithm not implemented or contains major error(s)
0	Not done; poorly attempted; major logic errors; major design problems; program does not implement any arrays.

Marking Rubric [File I/O Implementation]

Mark	Description
5	Excellent – program correctly reads and writes files as required by the program specifications; errors when reading/writing files will not crash the program; required CSV file format is implemented for written files; defensive programming techniques are applied when appropriate
4	Very Good – program correctly reads and writes files as required by the program specifications; errors when reading/writing files will not crash the program; required CSV file format is implemented for written files with minor error(s); defensive programming techniques are applied in some cases
3	Acceptable – program correctly reads and writes files as required by the program specifications; errors when reading/writing files will crash the program in most cases; required CSV file format is implemented for written files with minor error(s); defensive programming techniques are not applied in most cases
2	Needs Work – program reads or writes files with slight error(s); errors when reading/writing files will crash the program; required CSV file format is not implemented; defensive programming techniques are not applied
1	Unsatisfactory – program only reads or writes files with; program reads or writes to files with errors; errors when reading/writing files will crash the program; required CSV file format is not implemented; defensive programming techniques are not applied
0	Not done; poorly attempted; files are neither written or read by the program

Marking Rubric [Program Best Practices and Validity]

Mark	Description
5	Excellent – program passes all test cases; coding follows best practices and class standards; logic structure is efficient with no redundant code; program does not crash
4	Very Good – program produces the expected results for most of the test case results; coding does not follow best practices and class standards in few instances; minor logic errors; redundant code; program does not crash
3	Acceptable – program produces the expected results for some of the test case results; coding does not follow best practices and class standards in many instances; major logic errors; redundant code; program may crash unexpectedly
2	Needs Work – program fails to produce expected results; coding does not follow best practices and class standards; major logic errors; redundant code; program crashes unexpectedly
1	Unsatisfactory – program fails to produce expected results; coding does not follow best practices and class standards; major logic errors; redundant code; aborts when executed/ program crashes always
0	Not done; poorly attempted; major logic errors; major design problems; a scaffold was submitted