Project 1 – Searching and Sorting

Computer Science 2413 – Data Structures – Fall 2017

Due by 11:59 pm CST on Tuesday, 17 October 2017

This project is individual work. Each student must complete this assignment independently.

User Request:

"Create a simple system to read, write, sort, and search eclipse data with error checking."

Objectives:

1	Use C++ file I/O to read and write files while using C++ standard I/O (`cin`, `cout`) for user interaction, using appropriate exception handling.	*10 points*
2	Encapsulate primitive arrays inside a templated class that provides controlled access to the array data, retains information on array capacity and use, and can be used to store data of any class or primitive type.	*5 points*
3	Integrate appropriate exception handling into the templated array class.	*5 points*
4	Efficiently sort the data based on the field specified by the user.	*35 points*
5	Efficiently search the data based on the field specified by the user.	*15 points*
6	Integrate appropriate exception handling into classes and/or functions that implement searching and sorting.	*10 points*
7	Develop and use an appropriate design.	*10 points*
8	Use proper documentation and formatting.	*10 points*

Description:

For this project, you will revise and improve EclipseR from your labs in several ways. You are encouraged to reuse and build on your code from your labs. In addition to the functionality provided by EclipseR 0.4, your new *EclipseR 1.0* will take user input that allows users to specify the files in which data is stored and to sort and search for entries based on various data fields.

Operational Issues:

EclipseR1.0 will read eclipse data files (text files) via C++ file I/O. The names of the data files will be specified by the user using standard input. When EclipseR 1.0 starts, it will enter a data input loop, prompting the user for the name of a data file. If the user enters the name of an available data file, EclipseR 1.0 will open the file using C++ file I/O and read the data. If the user enters the name of a file that is not accessible, EclipseR 1.0 will report the error to the user ("`File is not available.`") and continue in the loop. If the user hits enter without entering anything else, EclipseR 1.0 will exit the data input loop. If no data has been read in when EclipseR 1.0 exists the data input loop, EclipseR 1.0 will exit. Otherwise, EclipseR 1.0 will move on to a data manipulation loop (see below).

The data files will be organized as they were in the labs. However, for this project, the eclipse data may be spread across multiple files. When EclipseR reads in each file, it must make sure the data in that file is internally consistent and alert the user to any errors encountered (as described in Labs 1 and 2). If a eclipse catalog number appears in more than one data file, EclipseR will retain the data read **most recently** (note the difference from the labs). As each file is read in, EclipseR must update internal tallies of the total number of eclipse data lines it attempted to read, the total number of valid eclipses read in, and the total number of eclipses currently stored in memory. (Note that these totals may be the same but they may be different, as some data lines may be invalid and some valid eclipse data lines may have the same catalog numbers.)

After reading in and storing all the eclipse data from all of the specified files, EclipseR will enter a data manipulation loop (as mentioned above). In this loop, EclipseR 1.0 will prompt the user with four options: 'O' for output, 'S' for sort, 'F' for find, and 'Q' for quit.

If the user selects output, EclipseR 1.0 will prompt the user for a filename and read it in from the console. If the user hits enter without specifying a filename, EclipseR 1.0 will send the output to standard out. Otherwise, it will attempt to open the specified file for writing. If the user enters the name of a file that is not accessible, EclipseR 1.0 will report the error ("`File is not available.`") and repeat the prompt for the data manipulation loop. If the file is accessible or the user has specified the standard out, EclipseR 1.0 will print out the data starting with the ten headers rows, followed by all eclipse data in whichever order it is presently ordered (whether sorted or unsorted), followed by a row showing the internal tallies, for example:

```
Data lines read: 15,847; Valid eclipses read: 13,347; Eclipses in memory: 11,879
```

Other than this trailing row of tallies, the data output format of EclipseR 1.0 will exactly match the *input format* of the file. **Note that this is different from the output format used by EclipseR 0.1 through EclipseR 0.4.**

If the user selects sort, EclipseR 1.0 will prompt the user for the data field on which to sort. The user may select any numeric value from 1 to 18 (inclusive), which corresponds to the column number on which to sort. If the user enters an invalid value for the data field on which to sort, EclipseR 1.0 will return to the data manipulation loop. If the user selects to sort by any valid data field, EclipseR 1.0 will sort the data based on that field. For sorting, most fields can be sorted numerically or lexicographically. However, Field 4 (month) needs to be sorted according to standard English three-letter month abbreviations (i.e., Jan, Feb, Mar, Apr, …, Dec). Also, note that fields 17 and 18 will be empty for some eclipses. These should be treated as maximum values for those fields, placing those eclipses at the end of listings sorted on those fields.

If the user selects find from the data manipulation loop, EclipseR 1.0 will prompt the user for the data field on which to search. As with sort, the user may select any numeric value from 1 to 18 (inclusive) and if the user selects any other value, EclipseR 1.0 will return to the data manipulation loop.

If the user chooses to find by any numeric column (see Lab 2), EclipseR 1.0 will prompt the user for the numeric value on which to search. If the user enters any text (doesn't just hit enter), EclipseR 1.0 will ensure that the value entered is valid (can be converted to an number of the appropriate type) and, if it is, search for the given value.

If the user chooses to find by Field 4 (month), EclipseR 1.0 will prompt the user for the month abbreviation on which to search. If the user enters any text (doesn't just hit enter), EclipseR 1.0 will ensure that the value entered is valid (is one of the standard English three-letter month abbreviations) and, if it is, search for the given value.

If the user chooses to find by any other valid field, EclipseR 1.0 will prompt the user for the value on which to search. If the user enters any text (doesn't just hit enter), EclipseR 1.0 will search for the given value but does not need to perform any validity check on the input data.

Note that, for all field values except 17 and 18, if the user hits enter without entering a value, EclipseR 1.0 will return to the data manipulation loop without performing a search. However, with Field 17 and

Field 18, if the user hits enter without entering a value, EclipseR will search for eclipses for which those data fields are missing (i.e., all partial eclipses).

If EclipseR 1.0 finds one or more fields matching a user request, EclipseR 1.0 should display for the user all of the corresponding rows, in the same format as when printing the data on all eclipses, including the header rows at the top but with a trailing row that indicates the number of eclipses found in the format shown in the following example:

```
Eclipses found: 1897
```

If the user selects quit from the data manipulation loop, EclipseR 1.0 should display a friendly parting message and exit.

Implementation Issues:

Given that the data may be read in from multiple files and that eclipse catalog numbers may be repeated across files but only the last read data for a given eclipse should be retained, EclipseR 1.0 will need to search for each eclipse catalog number before the eclipse data is inserted into the array of data. Consider how to most efficiently maintain the array and search for each catalog number as its data is read in. **You will create simple design document that explains and justifies your design choices with respect to these issues.** Please make this a PDF file and name it "**design.pdf**" in your submission.

If the data is unsorted or is sorted on a different field from the one on which the user is searching, EclipseR 1.0 will not be able to conduct a binary search for the data. In this case, EclipseR 1.0 should conduct a linear search instead. On the contrary, if the data is sorted on the field on which the user is searching, EclipseR 1.0 should conduct a binary search. Note that you will have to modify the binary search in an important way if the data in a given field is not guaranteed to be unique. **Your design document will explain and justify how you modified binary search to account for potential duplicates.** Further note that the only data field for which data is guaranteed to be unique is catalog number.

The only libraries you may use for this assignment are `iostream`, `iomanip`, `string`, and **`fstream`** (`#include <iostream>`, `#include <iomanip>`, `#include <string>`, **`#include <fstream>`**).

Be sure to use good object-oriented design in this project. That includes appropriate use of encapsulation, inheritance, overloading, overriding, accessibility modifiers, etc.

Be sure to use good code documentation. This includes header comments for all classes and methods, explanatory comments for each section of code, meaningful variable and method names, consistent indentation, etc.

You may write your program from scratch or may start from programs for which the source code is freely available on the web or through other sources (such as friends or student organizations). If you do not start from scratch, you must give a complete and accurate accounting of where all of your code came from and indicate which parts are original or changed, and which you got from which other source. Failure to give credit where credit is due is academic fraud and will be dealt with accordingly.

Due Date:

You must submit an electronic copy of your Eclipse project **and your design document** to the appropriate dropbox in Canvas by **11:59 pm CST on Tuesday, 17 October 2017**.