Project 1 – Searching and Sorting

Computer Science 2413 – Data Structures
Fall 2015

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

User Request:

"Create a simple system to read, write, sort, and search exoplanetary data."

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, automatically resizes as data is added, and can be used to store data of any class or primitive type.	*15 points*
3.	Integrate appropriate exception handling into the templated array class.	*10 points*
4.	Efficiently sort the data based on the field specified by the user.	*25 points*
5.	Efficiently search the data based on the field specified by the user.	*10 points*
6.	Integrate appropriate exception handling into classes that implement searching and sorting.	*10 points*
►	Develop and use an appropriate design.	*10 points*
►	Use proper documentation and formatting.	*10 points*

Description:

For this project, you will revise and improve ExoPlanIt from Project 0 in several ways. You are encouraged to reuse and build on your code from Project 0. In addition to the functionality provided by ExoPlanIt0.0, your new ExoPlanIt1.0 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. Further, ExoPlanIt1.0 will allow for the automatic resizing of arrays of data.

Operational Issues:

ExoPlanIt1.0 will read exoplanetary data files (text files in csv format) via C++ file I/O. The names of the data files will be specified by the user using standard input. When ExoPlanIt1.0 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, ExoPlanIt1.0 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, ExoPlanIt1.0 will report the error to the user and continue in the loop. If the user hits enter without entering anything else, ExoPlanIt1.0 will exit the data input loop. If no data has been read in when ExoPlanIt1.0 exists the data input loop, ExoPlanIt1.0 will exit. Otherwise, ExoPlanIt1.0 will move on to a data manipulation loop (see below).

The data files will be organized as they were in Project 0. However, for this project, you may not make assumptions on the number of data rows in the files. Instead, ExoPlanIt1.0 should start out by allocating a small array to hold the data, then dynamically allocate more memory as necessary. As ExoPlantIt1.0

reads the data files, it should make the same sanity checks that were made by ExoPlanIt0.0. Note that while the data may be spread across several files, the data for each system will be contained within a single file. In addition, ExoPlanIt1.0 will discard any exoplanet and any system for which there is incomplete data. That is, if a data field is blank for a given exoplanet, that exoplanet will be discarded from the data in memory but the file will be unchanged. Likewise, if discarding an exoplanet leaves a system with fewer exoplanets than indicated in the second field of the data file, that entire system will be discarded from the memory of ExoPlanIt1.0 but the data file will not be changed.

After reading in and storing all the exoplanetary data from the file, ExoPlanIt will enter a data manipulation loop (as mentioned above). In this loop, ExoPlanIt1.0 will prompt the user with four options: 'P' for print, 'S' for sort, 'F' for find, and 'E' for exit.

If the user selects print, ExoPlanIt1.0 will print out all of the data it read in, just as with ExoPlanIt0.0.

If the user selects sort, ExoPlanIt1.0 will prompt the user for the data field on which to sort. The user may select 'N' for name (which is the combination of system name and exoplanet name, just as it is given in the first field of the data files) or 'M', 'A', 'P', 'E', 'O', 'T', or 'K' (which correspond to the third through ninth fields of the data files, respectively). Note that the use may not attempt to sort based on the second nor tenth field. If the user enters an invalid value for the data field on which to sort (that is, not 'N', 'M', 'A', 'P', 'E', 'O', 'T', or 'K'), ExoPlanIt1.0 will return to the data manipulation loop. If the user selects to sort by any valid data field, ExoPlanIt1.0 will sort the data based on that field.

If the user selects find from the data manipulation loop, ExoPlanIt1.0 will prompt the user for the data field on which to search. As with sort, the user may select 'N', 'M', 'A', 'P', 'E', 'O', 'T', or 'K' and if the user selects any other value, ExoPlanIt1.0 will return to the data manipulation loop.

If the user chooses to find by name, ExoPlanIt1.0 will prompt the user for the name on which to search. If the user hits enter without entering a name, ExoPlanIt1.0 will return to the data manipulation loop. Otherwise, ExoPlanIt1.0 will ensure that the value entered is in the correct form for a full exoplanet name (some string, followed by a space, followed by a single character) and, if it is, search for it.

If the user chooses to find by any other valid field, ExoPlanIt1.0 will prompt the user for the value on which to search. If the user hits enter without entering a value, ExoPlanIt1.0 will return to the data manipulation loop. Otherwise, ExoPlanIt1.0 will ensure that the value entered is valid (can be converted to a double) and, if it is, search for it.

Note that if the data is unsorted, or sorted on a different field from the one on which the user is searching, ExoPlanIt1.0 will not be able to conduct a binary search for the data. In this case, ExoPlanIt1.0 should conduct a linear search instead. On the contrary, if the data is sorted on the field on which the user is searching, ExoPlanIt1.0 should conduct a binary search.

If ExoPlanIt1.0 finds an exoplanet matching a user request, ExoPlanIt1.0 should display for the user all of the exoplanets in the system where the matching exoplanet is located, in the same format as when printing the data on all systems.

If the user selects exit from the data manipulation loop, ExoPlanIt1.0 should exit.

Due Date:

You must submit an electronic copy of your source code through the dropbox in D2L by **Wednesday, October 14th by 11:00pm**.

Notes:

In this project, the only library you will use are iostream, fstream, string, and/or cstring.

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.