Project 2 – Linked Lists

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, merge, purge, sort, search, and write exoplanetary data."

Objectives:

1.	Use C++ file IO to read and write files, while using C++ standard I/O (`cin` and `cout`) for user interaction, using appropriate exception handling.	*5 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.	*5 points*
5.	Efficiently search the data based on the field specified by the user.	*5 points*
6.	Integrate appropriate exception handling into classes that implement searching and sorting.	*5 points*
4.	Store exosystems in a linked list maintained in order, sorted by name.	*25 points*
5.	Remove entries from the exosysm linked list while maintaining its order.	*15 points*
6.	Integrate appropriate exception handling into classes that implement linked lists.	*5 points*
7.	Add user options to select additional functionality (merge and purge), prompt the user for additional file names, and provide error messages in case of file access errors.	*5 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 1 in several ways. You are encouraged to reuse and build on your code from Project 1. In addition to the functionality provided by ExoPlanIt1.0, your new ExoPlanIt2.0 will allow users to merge and purge explanet data from multiple files in its working memory throughout its use. Exoplanet2.0 will also have a major change "under the hood" – rather than storing the list of exosystems using an array that automatically resizes as entries are added, ExoPlanIt2.0 will store the list of exosystems, and the lists of exoplanets within each system, using linked list data structures. (Note that the master list of exoplanets to be sorted and searched using binary search will still use a resizable array.)

Operational Issues:

ExoPlanIt2.0 will behave as described for ExoPlanIt1.0, except that rather than have separate data input and data manipulation loops, there will be a single user option loop that will allow the user to merge new

data or purge existing data at any time by selecting ‘M’ for merge and ‘P’ for purge alongside the options present for print, sort, find, and exit options. (However, since ‘P’ was previously used for print, that option will be remapped to ‘W’ for write. The codes for the other options will remain unchanged.)

The user will be prompted for a data file name when ExoPlanIt2.0 is first started, so that it has at least some data present before it enters the user option loop. As ExoPlanIt2.0 reads that first data file, it will ensure that the list is maintained in order based on name. (If ExoPlanIt2.0 has trouble accessing the data file named by the user, it should provide the user with an appropriate error message.)

If the user selects merge, ExoPlanIt2.0 will prompt the user for the name of an exosystem data file at the terminal prompt. ExoPlanIt2.0 will then attempt to read that file and merge its entries into the existing database, keeping the systems sorted in order by name. If, in the course of reading this file, ExoPlanIt2.0 encounters an entry name equivalent to one already present in the file, the old entry will be replace by the new entry. (If ExoPlanIt2.0 has trouble accessing the merge file named by the user, it should provide the user with an appropriate error message.)

If the user selects purge, ExoPlanIt2.0 will likewise prompt the user for the name of an exosystem file at the terminal prompt. ExoPlanit2.0 will then attempt to read that file but in this case it will purge (delete) from the database the entry with the same name as that of the entry just read from the file. If no such entry is found at the appropriate place in the list, ExoPlanIt2.0 will display to the user a message that an entry with that name was not found and move on to the next entry in the file. Essentially, ExoPlanIt2.0 is performing a set difference operation between the database and the entries in the purge file. (If ExoPlanIt2.0 has trouble accessing the purge file named by the user, it should provide the user with an appropriate error message.)

At the end of each merge and purge operation for the exosystem list, ExoPlanIt2.0 will replace the sorted array of exoplanets with a new array of exoplanets sorted by name by discarding the old exoplanet array and copying the data from the exosystem linked list to a new exoplanet array.

Implementation Issues:

In most areas, ExoPlanIt2.0 will be implemented just as was ExoPlanIt1.0. This includes how ExoPlanIt reads files (for the most part, see below) and prints data, carries out user interaction via standard in and standard out, encapsulates C primitive arrays, and how exception handling is implemented for arrays and similar classes. The big implementation change will be the data structure used in the code to hold the list of exosystem entries and the lists of exoplanets within each exosystem. For ExoPlanIt2.0, you are no longer allowed to store the exosystem list or the lists of exoplanets within each system in arrays – instead, ExoPlanIt2.0 must store those lists as linked lists. The only place ExoPlanIt2.0 will use arrays is for the master list of exoplanets that is sorted and searched.

Relatedly, while each line of each data file will follow the same format as previously, the lines may be shuffled such that the exoplanets for a given exosystem may not all be grouped together, nor will they necessarily be in name order.

Note that ExoPlanIt2.0 must keep the list of exosystems sorted by name as each entry is read in. Similarly, ExoPlanIt2.0 must keep the individual lists of exoplanets within each exosystem sorted by name as each entry is read in. These tasks must be accomplished by performing linear searches to find the appropriate places in the linked lists to insert each entry and changing pointers as necessary to accommodate each entry. This is true for both the initial file and any merge files specified by the user.

Similarly, ExoPlanIt2.0, purge files will be processed by linearly searching each list for each entry to delete and deleting it while retaining the order of each list.

Due Date:

You must submit an electronic copy of your source code through the dropbox in D2L by **Thursday, October 29th by 11:00pm**.

Notes:

In this project, the only libraries 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.