Project 4 – Sorting

Computer Science 2413 – Data Structures
Fall 2014

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

User Request:

"Create a system to read, merge, purge, search, write, and sort bibliographic data efficiently."

Objectives:

1.	Use C++ file IO to read and write files, while using `cin` and `cout` for user interaction.	*5 points*
2.	Encapsulate all primitive arrays inside classes that provide controlled access to the array data and retain information on array capacity and use.	*5 points*
3.	Integrate appropriate exception handling into classes that encapsulate arrays.	*5 points*
4.	Sort the initial bibliographic entries in a vector by name.	*25 points*
5.	Efficiently merge bibliographic entries in a vector sorted by name.	*15 points*
5.	Efficiently remove bibliographic entries from a vector sorted by name.	*15 points*
6.	Integrate appropriate exception handling into classes that implement 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 PublicScholar. You are encouraged to reuse and build on your code from previous versions of this project. PublicScholar4.0 will provide all of the functionality provided by PublicScholar3.0. However, PublicScholar4.0 will have a major change "under the hood" – rather than storing the list of bibliographic entries using an AVL tree, PublicScholar4.0 will store the bibliographic entries using a vector (resizable array). However, rather than keeping the vector sorted as it is created, as you did for PublicScholar 1.0, you will create the vector initially unsorted and then sort the vector. Depending on the sort method you choose to implement, this will allow for an improvement in PublicScholar4.0's run-time performance over that of either PublicScholar1.0 or PublicScholar2.0.

Operational Issues:

From the user's perspective, PublicScholar4.0 will behave as described for PublicScholar3.0.

Implementation Issues:

In most areas, PublicScholar4.0 will be implemented just as was PublicScholar3.0. This includes how PublicScholar reads and writes files, 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 bibliographic entries. For PublicScholar4.0, you are no longer allowed to store the bibliographic database in an AVL tree – instead, PublicScholar4.0 must store the database in a vector. This vector will be initially populated by adding entries in the order they are found in the input file. Once the end of the file is reached, PublicScholar4.0 will sort the entries based on name.

You may choose the sorting method to implement in PublicScholar4.0 for the initial sort after reading in the input file. However, in doing so, you should make choices that will make PublicScholar4.0 as efficient as reasonably possible.

You will also need to carefully consider the merge and purge operations. For example, if you have PublicScholar4.0 add to the database each item from the merge file as it is read in, while keeping the database sorted during insertion process, this will be a very expensive operation. Likewise, if you have PublicScholar4.0 delete from the database each item from the purge file as it is read, while keeping the database sorted during the deletion process, this will also be a very expensive operation. You should consider alternatives to keep PublicScholar4.0 efficient.

Given these efficiency considerations, for this project you should also turn in a separate cover sheet (typed up and saved in PDF) that explains how you decided to implement sorting the original file, merging, and purging, justifying your choices with reference to Big O running time of your choices and alternatives.

Also on this cover sheet, you must document all of the exceptions you are using in PublicScholar4.0 and how these are used within your code to ensure that PublicScholar4.0 runs properly.

Due Date:

You must submit an electronic copy of your source code through the dropbox in D2L by **Monday, November 24th by 2:45pm**.

Notes:

In this project, the only libraries you will use are iostream, fstream, and 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.