# Project 3 – AVL Trees

*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, and write 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. | Store the bibliographic entries in an AVL tree sorted by name. | *20 points* |
| 5. | Find bibliographic entries stored in the AVL tree based on name. | *15 points* |
| 5. | Remove bibliographic entries from the AVL tree based on name. | *20 points* |
| 6. | Integrate appropriate exception handling into classes that implement AVL trees. | *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 from Project 2. You are encouraged to reuse and build on your code from Project 2. PublicScholar3.0 will provide all of the functionality provided by PublicScholar2.0. However, PublicScholar2.0 will have a major change "under the hood" – rather than storing the list of bibliographic entries using a linked list data structure, PublicScholar3.0 will store the bibliographic entries using an AVL tree. This will improve PublicScholar3.0's run-time performance over that of either Publicscholar1.0 or Publicscholar2.0.

## *Operational Issues:*

From the user's perspective, PublicScholar3.0 will behave as described for PublicScholar2.0 except that it will, in general, operate more quickly because of its underlying implementation.

## *Implementation Issues:*

In most areas, PublicScholar3.0 will be implemented just as was PublicScholar2.0. 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 PublicScholar3.0, you are no longer allowed to store the bibliographic database in a linked list – instead, PublicScholar3.0 must store the database in an AVL tree.

### *Due Date:*

You must submit an electronic copy of your source code through the dropbox in D2L by **Wednesday, November 5th 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.