Assignment #4
CS 163 Data Structures

Submit your assignment to the D2L Dropbox (sign on via d2l.pdx.edu)

****Assignments in CS163 consist of written homework and programming****

1) **Use a Unix Debugger.** For this assignment, as part of your debugging process use gdb on unix. Write a program through ssh/ putty/ terminal, compile using g++ on the command line (g++ -g prog1.cpp member_functions.cpp), and then use gdb to debug (gdb ./ a.out). Write a paragraph about how you used it, what it helped you solve problems, and what kinds of features you wish it had.

2) **Algorithm.** As we have learned about table abstractions, we now know about hash tables, binary search trees, and advanced trees that keep the trees balanced. We analyzed non-linear solutions for organizing our data by the "value" of the data. Compare the algorithms for insert and retrieve between: between a hash table using chaining, 2-3 tree, and a red-black tree.

3) **Example Use.** Last time (in Program #3) you defined the terms Source Code Control and Makefile. Write a short makefile. Discuss in a short paragraph your findings after experimenting with the tool.

Programming – Goal: The goal of this program is to create a binary search trees (BST).

Background: Think about all of the times you have used a particular tool or data structure in a course that you have taken. One class might use a linked list or recursion and 2 years later another class references that material. CS250 covers graphs, trees, recursion, and growth rate. As does CS163 from a different perspective. CS350 then goes into the growth of functions, binary trees, and other in a deeper more complex level. Your job is to create an application that allows the user to enter in various topics and what classes these topics are related to so that someone can search a topic and find out where it occurs. In addition, it should reference directories or programming assignments done that have relevance for that particular topic (so that you can quickly bring up your BST program next time you need it!).

Data Structures: Certainly this assignment supports the notion of a table abstraction, as it will work with the value of the data. But, a hash table would be

limiting as it would not provide a way for us to get a sorted list of all of the topics of interest. Therefore, we will be creating an **abstract data type using a Binary Search Tree(s)** to store, retrieve (search), display, and remove information stored by topic (as the search key). This time we will also get our data sorted when we traverse using inorder traversal! Create TWO BSTs, one that organizes based on the topic (e.g., LLL) and the other that organizes based on the course name (e.g., all of the topics in a course).

What does retrieve need to do? It needs to supply back to the calling routine information about the item that matches. Retrieve, since it is an ADT operation, should not correspond with the user (i.e., it should not prompt, echo, input, or output data). It is possible that more than a single match will be received (so pass in an array of items so that the retrieve function can fill it for the client program).

In your design writeup, discuss other alternatives to using two BSTs to work with the two search keys. **What other choices are there and what would have been their corresponding efficiency?**

Things you should know...as part of your program:

1) Do not use statically allocated arrays in your classes or structures. All memory must be dynamically allocated and kept to a minimum!
2) All data members in a class must be private
3) Never perform input operations from your class in CS163
4) Global variables are not allowed in CS163
5) **Do not use the String class! (use arrays of characters instead and the cstring library!)**
6) Use modular design, separating the .h files from the .cpp files. Remember, .h files should contain the class header and any necessary prototypes. The .cpp files should contain function definitions. You must have at least 1 .h file and 2 .cpp files. **Never "#include" .cpp files!**
7) Use the iostream library for all I/ O; do not use stdio.h.
8) Make sure to define a constructor and destructor for your class. Your destructor must deallocate all dynamically allocated memory.
9) Remember that 20% of each program's grade is based on a written discussion of the design. *Take a look at the style sheet which gives instruction on the topics that your write-up needs to cover.*