Programming Assignments #3 and 4
CS 163 Data Structures

Submit your assignment to Canvas

Background: As we begin the next phase of CS163, we will be working with "Table Abstractions". A Table ADT allows us to search on the "value" of the data without requiring a particular location to be known. It allows us to store the data using non-linear techniques. There are two non-linear pointer based data structures we have learned to support Table ADTs:

1. Program #3: Hash tables (implemented as an array of linear linked lists). A hash table comes to mind as a good alternative for programs that require working by "value" but that don't require that the data be sorted.
2. Program #4: Binary Search Trees (implemented as a non-linear pointer based data structure where every node has a left and right pointer).

Specifics: Have you ever wanted to find information very quickly? Maybe you have been collecting information like comic books and now you want to find a particular issue really quickly. My daughter and her husband collect superhero comics. She could talk forever with her friends about superheroes and comic book adventures. There was a three hour car ride once where she and her best friend debated the benefits and drawbacks of Superman versus Aqua man for the entire trip. It was a very animated discussion! We are going to apply this to the concepts of a Table ADT and develop software that will help us find information quickly.

For this programming assignment, we will create a program called "CS Comic Collector" that will help us collect and catalog comics that we (or others) are interested in. We will be using an external data file for program #3. The information that we want to keep track of includes:

1. The name of the comic (Amazing Spider-Man)
2. The publisher (e.g., DC Comics, Marvel, Dark Horse, etc.)
3. The superhero (e.g., Spider Man)
4. The issue (Issue #2 of the Amazing Spider-Man)
5. A review (from you or someone else)
6. Other items you are interested in?

- The data file's name will be "Comic.txt"
- The delimiter will be a '|'
- You may share data files among the people in our class
- You may add or modify the above fields if there is other information you think would be better (or matches your data set)
- You may use data obtained from the web

ADT Functions Required for both assignments: Your Table ADT must implement these functions:

a. Add new comic into the table,
b. Retrieve information about the comics entered by the superhero name
c. Remove a comic based on the name of the comic book
d. Display all comics featuring a superhero
e. Display all (program #3: not ordered, program #4: sorted by superhero)

PROGRAM #3 - Details

Program #3: The goal of the third program is to create a **hash table using chaining**. Hash tables are very useful in situations where an individual wants to quickly find their data by the "value" or "search key". The client program uses a "key" instead of an "index" to get to the data. The key is then mapped through a **hash function** which supplies an index! You will want to create a hash function that creates an index LARGER than the array size so that the mod operation will help in distributing the data.

The data must originally be retrieved from an external data file. This is required for Program #3 assignment. To properly evaluate the hash function chosen, we need a large data set! **However:** your program will not be removing or modifying the data from the file.

For program #3, use the superhero name as your key for the hash function. So all comics with a particular super hero will be inserted into the same chain.

In your efficiency writeup evaluate the performance of storing and retrieving items from a hash table. Monitor the number of collisions that occur for a given set of data that you select. Make sure your hash table's size is a prime number. **Try AT LEAST TWO different table sizes**, and evaluate the performance (i.e., the length of the chains!). **Your efficiency writeup must discuss what you have discovered.**

PROGRAM #4 - Details

Program #4: The goal of the fourth program is to move from a hash table and implement a binary search tree. The advantage of a binary search tree is the ability to retrieve our data using a logarithmic performance if the tree is relatively balanced and be able to search for a range of information and obtain our data in sorted order. **PLEASE NOTE – we are REPLACING the concept of a hash table with a binary search tree when moving from program #3 to program #4.**

In the Program #4 efficiency writeup, evaluate the performance of storing and retrieving items from a BST. Monitor the height of the tree and determine how that relates to the number of items. If the number of items is 100 and the height is 90, we know that we do not have a relatively balanced tree!! Use the information from the Carrano reading to assist in determine if we have a reasonable tree, or not. **Your efficiency write up must discuss what you have discovered.**

==========

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.