Programming Assignment #3
CS 163 Data Structures

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

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. A hash table comes to mind as a good alternative for when working by "value" (when sorting is not required). Chaining is a strategy for collision resolution.

Goal: The goal of the third program is to create a hash table using chaining per Topic#6 and Lab #6. Hash tables are very useful in situations where an application needs to quickly find information about some data using a "search key". You could think of them as similar to an array, except the client program uses a "key" instead of an "index" to get to the data. The key is then mapped through the hash function which turns it into an index! Every key that maps to the same index causes a "collision" which in turn will be inserted into a "chain" implemented using a linear linked list.

As Carrano indicates, the success of a hash table implementation is related to the choice of a good hash function. We want one that is easy to compute but still evenly distributes the data. Hash functions should be able to distribute the data especially when there are large quantities of data in existence. Make sure to spend time developing a hash function that meets these criteria! The hash function needs to be your own creation and your own code. I want you to experiment with two different hash functions (with different table sizes as well) and evaluate the number of collisions and how data clusters.

Specifics: The Portland housing market is really growing. Finding housing can be challenging when we need to move. Do you find a place near the city and pay the higher rents? Or, are you willing to commute from Gresham or Hillsboro? I know some people even commute from Woodburn or Salem in order to be able to afford housing.

As you search for a new place to live, you are giving names to each place you are considering. For example, I am thinking about the "Open floorplan", "By the water" and "Cozy". I will want to keep information about (a) the location (e.g., Gresham), (b) the size (e.g., square feet), (c) the number of bedrooms (e.g., 3) and (d) the distance from PSU (e.g., 15 miles). You can also keep additional information (such as the time it takes to commute). **This information will be best if you use an external data file to read in the data.**

In Program #3, we will be implementing a hash table using chaining as the collision resolution technique that will assist you in looking up housing available, using the name of the unit that you've given

The **Table ADT operations that must be performed on this data are:**

1) Add a new housing option
2) Remove by keyword
3) Remove all that are beyond a given # of miles (passed in as an argument)
4) Retrieve the information about a particular match (using the keyword)
    - Retrieve is NOT a display function and should supply the matching information back to the calling routine through the argument list
5) Display all information for a particular matching key

Data Structure: Write a C++ program that implements and uses a **table abstract data type using a hash table (with chaining)**. Evaluate the performance of storing and retrieving items from this table. Monitor the number of collisions that occur for a given set of data that you select. Experiment with what would make for a good hash function in the event there were large quantities of information, like we talked about in class. Make sure your hash table's size is a **prime number**. Try different table sizes, and evaluate the performance (i.e., keep track of the length of the chains!). **See what happens** if you have a hash table size that is a power of 2 instead of a prime number. How does this affect the distribution of your data? **The 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 that is part of this Table ADT!
2) All data members in a class must be private
3) Never perform user 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 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) Implement ONE FUNCTION recursively!
10) With the destructor, use POINTER ARITHMETIC rather than the subscript operator.** IMPORTANT to do this with run-time efficiency in mind**