Programming Assignment #3
CS 163 Data Structures

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

Programming – Goal: The goal of the third program is to create a hash table using chaining per Topic#6 and Lab #4. Hash tables are very useful in situations where an individual wants to quickly find their data by the "value" or "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!

Programming – Problem Statement: Have you ever used Monster.com to perform a job search? I just typed in "Software Engineer" with "C++ programming skills" for Portland and got 17 matches! There's Experis to test games (and yes, they want Xbox experience!), MSEI (requiring 10 years of Java and/or C++), CyberCoders (using linux!!!), and so on.

However Monster has some issues. Each company's posting is different and takes time to decipher what they are really expecting and what you would be doing as an employee. *So I thought we could improve things!*

We will be implementing a hash table using chaining as the collision resolution technique that uses a listing of jobs to determine if one matches your choice. The client program will provide a job title (e.g., Software Engineering) and a list of skills or keywords (C++, linux). Your program will hash on the job title and then search through each match for matching skills. Each item in our table will have the following information (at a minimum):

1) Employers name
2) Location (city, state)
3) Job Title
4) Job Description
5) Required Experience
6) Pay rate expected

The jobs being inserted should come for from an external data file (called jobs.txt); you can decide on the format. Just make sure to submit it to D2L. Your ADT will load the jobs from the file to perform the desired searches.

Data Structures: Write a C++ program that implements and uses a **table abstract data type using a hash table (with chaining)** to load from a file, retrieve (e.g., find a match), and display.

What does retrieve need to do? It should take the job title desired and the list of keywords. Retrieve, since it is an ADT operation, should not correspond with the user (i.e., it should not prompt, echo, input, or output data). Retrieve should have an array of jobs as an argument that is filled by the ADT. The jobs may be stored as a struct or class object.

As stated previously, hash on the desired title. Then YOU DECIDE which data structure to use for the second layer search of keywords. Are you going to do a sequential search through the chain to search for matching keywords in the description, required background or salary? Or will you build a second layer hash table? **(EXTRA CREDIT (+5 points) for applying hashing to the keywords of skills!)**

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. Make sure your hash table's size is a prime number. Try different table sizes, and evaluate the performance (i.e., the length of the chains!). **Your design writeup must discuss what you have discovered.**

In summary, the required functions for your TABLE ADT are:

1. **Constructor,**
2. **Destructor**
3. **Load jobs** *(use external data files)*
4. **Retrieve** *(based on job title and skills/keyword matches)*
5. **Display all**

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.*