CSCI 335
Third assignment
Total: 100 points

Due 4/2/2015
Please, follow the blackboard instructions on writing and submitting programming assignments
We will not debug your assignment. It should run correctly to receive credit.

Hashing (30 points)

Implement quadratic probing and then run tests on the number of probes required at various load factors to obtain a graph like the one shown in Figure 5.12.

For this assignment, results and analysis are particularly important. You're not just programming; you're experimenting with a program and reporting on the results of those experiments.

To keep things simple, you may assume that items to be stored are integers and that there will never be more than 1,000 of them stored at any one time. Do not rehash.

Maps and Hashing (70 points)

(a) Implement the most efficient version of the "word puzzle" algorithm of section (the one shown in Figure 4.72) using the STL map. Use the exact implementation provided with the book. Your executable should run as follows:
./testMaps <word_file.txt>
where <word_file.txt> is a file that contains a list of words (dictionary).

1. Compute and print out the time it takes for the computation of the adjacent words.
2. Prompt the user to provide a word, and then print out all its adjacent words. Print out also the time it took for searching for that word.

(b) For this part of the assignment DO NOT use the STL map. Create your own *templated* hash-map class that only supports lookup (find), insert and delete (you can use lazy deletions) operations. You should also provide an operation that visits all the elements in the hash-map (in the way they are stored in the hash table). The hash-map (use quadratic probing) will store a key with an associated value. Note that you need to provide a good hash function for strings. You don't have to implement iterators for your hash-map implementation. Now using your own hash-map implementation, implement the most efficient version of the "word puzzle" algorithm. Your executable should run as follows:
./testMyHashMap <word_file.txt> <initial_size_of_hash_table>

The first argument is the list of words (dictionary) and the second argument an integer that specifies the initial size of the hash table.

Provide the tests (1) and (2) as in part (a). Experiment with different initial sizes (from very small, i.e. 51 to larger) and measure the change in running time.