Project 10
EE312 Spring 2014
FIVE POINTS

Full circle….

General:
We're down to the very end of the semester, so by this time, I assume you're very comfortable jumping right into the middle of a software project. After all, that's what happens in the real world – it's very rare that you're working on anything from scratch. Anyway, in this final project, I have a HashTable built almost completely. It actually works (although it has poor performance) out of the box. I'm going to insist that you put on the last bit of polish for this HashTable (you'll write the resize function). I'm going to hope that you'll also take advantage of the project and review the code that I wrote. Not only does it implement a data structure that you'll be tested on, it also utilizes several C++ constructs to help make the design simpler (without making it less efficient).

Note that in this project you will be editing two files. You will make changes to HashTable.h and you will make changes to Project10.cpp. You may even end up making changes to String.h. When we grade your program we will copy only those three files, so if you think you need to modify any other files in the project be sure to check with us first.

Part 1
The resize function must double the size of the array used by the HashTable (i.e., increase cap by a factor of 2). Note that once you've doubled the array, you'll need to recalculate where all the old keys go. Words that were in the same chain previously may be in different chains, and using Knuth Multiplication, the reverse is possible too…. Words that were in different chains in the old array can be in the chain in the new, larger array. As you rebuild the chains, do not deallocate the link cells and make new ones. Rather, simply unhook each cell from its old linked list and hook it into the correct chain in the larger array.

Once you've written the resize function, the HashTable will execute all the code for part1. You'll note that Part 1 just dumps a bunch of words into the HashTable (actually, it dumps the entire American-English dictionary from Project 1 into the HashTable) and then analyzes how long the chains are. The purpose of part 1 is to discover how important it is (or isn't) to use a good hash function and/or to shuffle hash codes over array positions. You should understand what each of the tests are doing in Part 1, even though you don't have to write them. You just might find that knowledge showing up on the final exam.

Part 2
For Part 2, I want you to repeat Project 1 (remember that?). In this case, I'll give you a vector<String> for the dictionary, but once again you'll get a char* pointer for the article.

You should put all the dictionary words into a hash table (pick whichever hash table you like the best after your Part 1 experiments). Then, extract words from the article, create String objects from each word, and check to see if there is an EXACT MATCH of that word in the dictionary. Note that this time around, I want you to only detect exact matches, so the capitalization must be precisely the same as what's in the dictionary. Given how we're working with the HashTable, it's much easier to perform case sensitive matching.

Part 3

For an extra challenge (and as a requirement if you want to earn that 5^{th} point), I want you to make the spellcheck function case-insensitive. However, I do not want you to achieve this by changing the dictionary words. Instead, I want you to make a hash function that is case insensitive ("Apple" and "apple" will hash to the same value). Then modify the String operator== function so that it does a case-insensitive compare.

Good luck!