Programming Assignment 2

Working with Dictionaries

# **Due date**: at 11:55 p.m. on Nov. 11 (Sunday).

**Submit two zip files along with your ReadMe file** **to ReggieNet.**

**Description**:

You are to write a C++ program to implement a spelling checker. The program will first build a dictionary of correctly spelled words by reading the words from the files dict.txt in the current directory. It will then ask the user for the name of a text file to spell check. The program must print to standard output a list of misspelled words and the line numbers on which they occur. Also, for each misspelled word, list any words in the dictionary that are obtainable by applying any of the following rules:

1. Add one letter to the word (at any position)
2. Remove one letter from the word
3. Exchange adjacent characters

Note that line numbers must begin with 1 for the first line in the file.

You are determining what the interface of this program looks like, but you will be evaluated partially on the quality of the interface. Make sure that your output is clear and clean. Overly wordy output is just as bad as cryptic output. Do make sure you spell well – this is a spelling checker, after all.

Remember that good design is also important to the evaluation of your program.

**Specifics:**

You will actually submit this program in two parts.

**The first part** will use a dictionary class that is implemented using an AVL tree. That class is provided for you, and can be found in ReggieNet. Make sure you read the comments carefully. Sample dictionary and text files will also be made available in ReggieNet. In Part 1, you will submit your working spell checker that uses the given AVL tree.

**The second part** of the program will involve replacing the dictionary class internals with a hash table. Note that you are not permitted to change any of the interface to the class (the prototypes of the public functions). You should not need to change your program at all, just the class. Note that even though you may not use all of the capabilities in the Dictionary class, you must provide correct implementations for all of the public functions.

**Hashing Requirements:**

* + Use the hash function from figure 5.4 in page 195.
  + Use quadratic probing to resolve collisions.
  + Rehash when the array becomes 50% full.

**Simplifying Assumptions:**

* We don’t care about case; convert everything to lower case.
* Words are strings of letters. Treat everything that isn’t a letter as breaking up words.
* Each line ends with a newline character.
* All files are in the same directory as the program.

**Submit two zip files along with your ReadMe file**:

1. For Part 1, submit **your** source code (not the Dictionary class files) and Makefile
2. For Part 2, submit your entire program and Makefile

**Grading Standards:**

Note that any modifications to the Dictionary class for Part 1 or to the interface of the Dictionary class for Part 2 constitute failure to follow the assignment.