ECS 60 **Programming Assignment #4 (50 points, 45 minutes)** **Fall 2015**

Due: November 18th, 11:59pm to p4 directory
Primary source file name: checker.cpp, checker.h Executable name: checker.out Makefile name: Makefile
Minimum handin command: handin cs60 p4 checker.cpp checker.h Makefile authors.csv

You are to write a Checker class that will check the spelling of the words in a file. The constructor of your class will receive an array of the words in the dictionary file named words.txt. The name of a tester file will be passed as the only command line parameter to the program. For each misspelled word, your program will provide an alphabetically sorted list of any word in the dictionary that is the same length, and differs by only one letter from the misspelled word. For each properly spelled words, your Checker class will provide a list containing just that word.

You may assume that your program will be tested with a file in which ten percent of the words are misspelled. Of those misspelled words, about four percent will not match any word when single letters are changed, in which case your list count should be zero. You can construct your own test files using WordsMaker.out. You will find my executable, checkRunner.h, checkerRunner.cpp, a barebones checker.cpp, a barebones checker.h, a barebones Makefile, words.txt, WordsMaker.out, and some test files in ~ssdavis/60/p4. Remember to hand in ALL the files that your program depends upon, except words.txt, CPUTimer.h, checkerRunner.h, and checkerRunner.cpp. These last three files will be copied into your directory before make is called. You may assume that words.txt will never be changed.

Word test file format:

1. Since checkerRunner.cpp reads and processes this file, this information is just to help you understand the files. The Checker class will never deal with the files directly.
2. words-<number of words>-<percent misspelled>-<seed for random number generator>.csv.
3. The first line has the number of words in the file.
4. All succeeding lines have the format: <word to match>, <# of matches> [, matching word][, matching word]…

Grading:

The program will be run with three 100,000 word test files.

1. Proper operation with no errors = 30 points
2. Time score is only possible if there are no errors. It is possible to earn five points extra credit.
   Time Score = min (25, 20 * Sean's Total CPU Time / Your Total CPU Time).
3. Programs must be compiled without any optimization options. You may not use any precompiled code, including the STL and assembly. Other than Weiss files, you must have written all of the code you submit.
4. You may not have any static, or global variables larger than 1000 bytes, since they would be created before the CPU timer begins.
5. If your CPU time exceeds 30 seconds for a 100,000-word file, then please send me an e-mail when you submit your program so I can run it separately. If you do not send me such a warning, I will deduct 5 points from your score.
6. Miscellaneous:
   6.1. Brainstorm a lot before starting to code. Think about which operations are done a lot, and minimize their cost.
   6.2. Even though the Chris will not see my solution, I have instructed him to not help you with your design. He will help with debugging though.
   6.3. Keep things simple, and get things running first, and only then use gprof to learn where things are going slowly.
   6.4. Remember to turn in dsexceptions.h if your program needs it! I suggest you create a directory, and copy all of the files you think you need into it, and then try compiling before you use handin.

```
[ssdavis@lect1 p4]$ checker.out words-100000-10-4.csv
CPU Time: 1.12913
[ssdavis@lect1 p4]$ checker.out words-100000-10-5.csv
CPU Time: 1.16145
[ssdavis@lect1 p4]$ checker.out words-1000*6.csv
CPU Time: 1.07524
[ssdavis@lect1 p4]$
```

```
[ssdavis@lect1 p4]$ WordsMaker.out
150499
Number of words >> 10
Percentage bad (0 - 100) >> 50
Seed: 4
Done
[ssdavis@lect1 p4]$ cat words-10-50-4.csv
10
must,1,must
suizable,2,seizable,suitable
elopeo,3,eloped,eloper,elopes
carageen,1,carageen
rfrrusive,0
granuloblast,1,granuloblast
elopers,1,elopers
bieaching,2,bleaching,breaching
succinic,1,succinic
rhinorrhagia,1,rhinorrhagia
[ssdavis@lect1 p4]$
```

```
int main(int argc, char **argv)
{
  char word[MAX_LENGTH + 1], matchingWords[100][MAX_LENGTH + 1];
  int numWords, count;
  DictionaryWord *words = readWordsFile();
  MatchingWords *matchingWordsKey = readTesterFile(argv[1], &numWords);
  CPUTimer ct;
  Checker *checker = new Checker((const DictionaryWord*) words, NUM_WORDS);
  delete words;

  for(int i = 0; i < numWords; i++)
  {
    strcpy(word, matchingWordsKey[i].word);
    checker->findWord(word, matchingWords, &count);

    if(count != matchingWordsKey[i].count)
    {
      cout << "Incorrect count for trial# " << i << " for "
        << matchingWordsKey[i].word << " should be "
        << matchingWordsKey[i].count << " but received " << count << endl;
    }  // if incorrect count
    else // correct count
    {
      for(int j = 0; j < count; j++)
        if(strcmp(matchingWordsKey[i].matches[j], matchingWords[j]) != 0)
        {
          cout << "Words don't match for trial# " << i << " for "
             << matchingWordsKey[i].word << " match# " << j << " should be "
             << matchingWordsKey[i].matches[j] << " but received "
             << matchingWords[j] << endl;
        } // if invalid match
    } // else correct count
  }  // for each word

  cout << "CPU Time: " << ct.cur_CPUTime() << endl;
  return 0;
}  // main()
```