Algorithms in C++: Final Exam Assignment
Finding Anagrams

1. **Objective**
   Your goal is to write a program that scans the dictionary (a text file of words) and finds the longest words that have one or more anagrams present in the dictionary. An anagram is a word whose letters are rearranged to form another word found in the dictionary.

2. **Problem**
   Create a C++ project with a single source file called *anagramfinder.cpp*. The program takes in a single command line argument <dictionary file>. This file contains up to 500,000 words, each of which will be encoded in extended ASCII, with values ranging from 0 to 255. Each word will be from 1 through 50 characters in length. When finding anagrams, you should take a case-insensitive approach. Letters *A* through *Z* can be interchanged with letters *a* through *z*. If another word has the same letters, regardless of case, as well as hyphens and/or apostrophes, it is an anagram. Do not worry about the case of extended ASCII letters. Those characters may be found in some words, but those words should be discarded. Process just the words containing the uppercase letters *A* through *Z,* lowercase letters *a* through *z*, hyphens, and apostrophes. Line endings in the dictionary file will be UNIX style, with just '\n' and not '\r\n' characters.

   Sort the output so the words in each group are alphabetized as well as the overall result. You should use the typical ASCII ordering, which means words beginning with capital letters appear before those beginning with lowercase letters.

   Consider the input of this dictionary:

```
pots
stop
tops
a
stops
c
eat
ate
trace
tea
d
posts
f
react
cater
```

   Your program should produce the following output for this file:

```
Max anagram length: 5
cater
react
trace
```

```
posts
stops
```

A test script is supplied for you to check the correctness of your program on small inputs. Once you are confident that it works, you should download a large dictionary file from the Internet. Then focus on improving the efficiency of your implementation. Be sure to

a) Use valgrind to check your program for memory leaks.
b) Make sure your code compiles and runs correctly in the lubuntu virtual machine supplied at the start of the semester.
c) Change -g in the makefile to -O3 so that the compiler optimizes the code it generates.
d) Use the time command line utility to test your work.

```
time ./anagramfinder dictionary.txt

…output…

real    0m0.354s <- This is the amount of time it takes to run your program.
user    0m0.344s
sys     0m0.020s
```

3. Submission

Create a zip file called *anagramfinder.zip* that contains *anagramfinder.cpp* and the *makefile*. Do not submit any extraneous files.

Your submission must be received by the deadline posted in Canvas. Since this project is in lieu of the final exam, the typical late policy does not apply. **If your submission is not received by the deadline, a zero will be recorded for the grade on this assignment.**

Your work must be completed individually. Moss will be used to check for code similarity. **If Moss finds your code similar to that of others in the class, all offending parties will be reported to the Honor Board.** Remember, with a fully open-ended project, the likelihood of your code overlapping with someone else's is minimal.

4. Bonus

The student in the class with the lowest real time on the largest input will earn 5 points of extra credit.