CS515 – Lab 13

The purpose of this lab is to give you a bit more practice using ADT's before the exam by having you write a word game program. Download all files from `~cs515/public/13L`

Task 1: Four-letter Word Check

Write a program to find the set of words in a dictionary that differ by only one letter from any given input four-letter word. For example, given input word "desk", we can have the second letter replaced by 'i' and get "disk". However, the word "does" doesn't qualify since the last three letters of "desk" have all been changed to others letter (although the two words have three common letters)

Your program should create the set of words based on the dictionary file called **`words`** (it should be located in the same directory as your program). This dictionary contains about 480,000 words. In your program, words that have non-alphabetic letters should be ignored and capitalized words will be treated as all lower-case words. The program should verify all queries are 4-letter words.

A sample run of the program is shown below:

```
./wordcheck
Please input a 4 letter word:win
wrong length!
wine
aine bine cine dine eine fine gine hine kine line mine nine pine rine sine tine vine wane
wene wice wide wife wile wime wina wind wing wini wink winn wino wins wint winy wipe wire
wise wite wive wone wyne
Total 41 words.
Please input a 4 letter word:root
boot coot foot hoot loot moot poot riot roit rood roof rook rool room roon roop roos rort
rost rout rowt royt ryot soot toot
Total 25 words.
Please input a 4 letter word:nice
bice dice fice lice mice nace nica nich nick nico nide nife nike nile nine niue nixe pice
rice sice tice vice wice
Total 23 words.
Please input a 4 letter word:
```

If you test the program with keyboard inputs, the input should be ended by `^D` (`control-d`). Or, you can test the program with input file redirection. A sample input file **`w_input0`** is provided in the starter code. You should hard code the name and path of the dictionary file in your source code for this lab.

Requirements:

- You are not allowed to directly search through the dictionary file for each query. Your program should read the file only once before processing all queries.
- A quick way to convert a string to all lower case is to use the STL algorithm **`transform`**:
  ```
  std::transform(str.begin(), str.end(), str.begin(), ::tolower);
  ```
  If you do this, make sure to include header file <`cctype`> so its `tolower()` function can be passed into `transform()`.
- You should use appropriate STL containers in your program. In particular, consider using a compound container.

Task 2: C++ proficiency question—must be done individually

Complete the 30-minute quiz on Canvas entitled "Lab13L—Task 2". Even if you did not go to lab, completing this quiz will count for attendance purposes for the lab (guaranteeing a minimum of 50 points for the lab). Make sure you do not start the quiz until you are ready to devote a full 30 minutes, and make sure to submit whatever code you have to the quiz before time runs out. Initial code submission for this task occurs through Canvas, but then you should also submit the same file through the normal procedure*. You will be penalized if you submit a different file using Mimir than you did on Canvas*. This question is worth 20 points for the lab.

Submission:

- Submit your source files and a **`README`** file.
- You should also fill out the README file and submit it as well. To submit the files in Mimir, follow the link for this lab in MyCourses. Here is a list of files you are expected to submit:

  `wordcheck.cpp task2.cpp makefile README`

- As always, do not turn in executables or object code, and make sure your submission compiles successfully on Agate. Programs that produce compile time errors or warnings will receive a zero mark (even if it might work perfectly on your home computer.) To check this, use the make command with the provided makefile, and check that your submission for Task 1 passes tests on Mimir (agate is the only place to make sure Task 2 is working correctly).

Important:

You must include the standard comment block in each of your source files, including your name, section, date and collaboration details. You must also finish the README file along with your programs.

You should also include detailed collaboration declaration information in your comments. If you worked in pairs in this lab, each of you must include the partner's name in your program comment. Both you and your partner must complete an individual submission in order to earn a grade. If you work by yourself, you must indicate in the program comments that you have worked on your own independently.

You can resubmit as needed—your last submission will be graded. Here is a tentative grading scheme:

wordcheck test run 0	20
wordcheck test run 1	20
wordcheck test run 2	40
Task 2 quiz	20
Others	
open file in wordcheck multiple times	-60