CS819 FALL 2016 DUE: OCT 13, 2016
ASSIGNMENT 4

Purpose

The purpose of this assignment is to give you more practice using STL container classes. You will write two programs: one that works with Google online N-gram word frequency data, and the other is a word game.

Problem

Task 1: Google N-gram word frequency

Google Labs' N-gram Viewer (books.google.com/ngrams) is a tool that lets you search for words in a database of 5 million books from across centuries. The data set provides about 3 terabytes of information about the frequencies of all observed words and phrases in books published between 1500 and 2012. Google also provides a viewer tool that allows users to visualize the relative historical popularity of words and phrases. You may watch this TED talk to learn how the N-gram Viewer works and what surprising facts we can learn.

http://www.ted.com/talks/what_we_learned_from_5_million_books

In this assignment, you will be working with sets of data that are abstracted from Google N-gram dataset (storage.googleapis.com/books/ngrams/books/datasetsv2.html). The files are **`some_a_words.csv`**, **`words_start_with_q.csv`** and **`all_words.csv`**. Each of the files consists of compressed tab-separated data; each line has the following format (where "\t" is a TAB, and "\n" is a NEWLINE):

```
word \t year \t match_count \t volume_count \n
```

You are asked to write an interactive program that answers user query of word occurrence given the starting year (no earlier than 1800), until the present. The program takes one command line argument that is the name of the data file, and prompts the user for queries. Each query consists of a word and the year from which to display the word's frequency (up until the most recent year).
Note that since **`all_words.csv`** is a fairly large file, you don't really need to keep a local copy (for the sake of saving your disk quota). You may directly pass the full path of the filename on Agate to the program as follows:

```
./wordfreq ~cs819/public/4P/all_words.csv
```

Below is a sample run:

```
./wordfreq some_a_words.csv
Which word starting which year? advised 1999
1999 138301
2000 155644
2001 159699
[…(more lines here)…]
2006 215711
2007 227259
2008 287491
Which word starting which year? airport 1990
1990 76600
1991 81484
1992 84860
[…(more lines here)…]
2006 167451
2007 175702
2008 173294
Which word starting which year?
```

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 **`d_input0`** is provided in the starter code. To test your programs, you should start with smaller dataset and progressively move to larger files.

Using your program, you may discover interesting facts from the `all_words.csv` data set. For example, you can see the trends for the words "computer" and "horse" since the 19th century. It seems while people have a growing interest in computers, horses seems to maintain their popularity in the past centuries. Try different queries and see what you can discovery. Note that our largest dataset, the 90 Megabyte `all_words.csv` is only a small subset of the Google full N-gram words so it may not contain all the "cool" words that you want to test.

Requirements:

- Your program should process the data file and close the file before it prompts for queries. For each query, the program will look up the data stored in memory, not from the file directly.
- You should use appropriate STL containers and iterators to solve this problem. You actually need only one map in the program.

Task 2: 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 other letters (although the two words have three common letters)

Your program should create the set of words based on the dictionary file located at **`/usr/share/dict/words`** on Agate. 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 words: win
wrong length!
Please input a 4 letter words: wine
aine bine cine dine eine fine gine hine kine line mine nine pine rine sine
tine vine wane wene wone wyne wice wide wife wile wime wipe wire wise wite
wive wina wind wing wini wink winn wino wins wint winy
Total 41 words.
Please input a 4 letter words: boot
coot foot hoot loot moot poot root soot toot biot blot brot bsot boat bolt
bort bott bout boyt boob bood boof book bool boom boon boor boos booz
Total 29 words.
Please input a 4 letter words: rice
bice dice fice lice mice nice pice sice tice vice wice race ribe ride rife
rile rime rine ripe rise rite rive rica rich rici rick rico rics
Total 28 words.
Please input a 4 letter words:
```

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.

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);
  ```

  Here 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.