CS515 – Lab 12

The purpose of this lab is to give you practice choosing data structures and using STL container classes by having you write a program that works with Google online N-gram word frequency data. To start, download all files from `~cs515/public/12L`

Task: Google 1-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 2008. 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 lab, 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:

```
word year frequency volume_count
```

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 do 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 /home/csadmin/cs515/public/12L/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 1-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. Try to use compound STL containers. You actually need only one map in the program.

Submission:

- Submit your source files, a **`makefile`**, and a **`README`** file.
- The `makefile` **must** be able to make the target named **`wordfreq`** when the `make` command is called without any arguments.
- 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:

  `wordfreq.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 passes tests on Mimir.

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:

test run 1	20
test run 2	20
test run 3	30
test run 4	30
others	
word freq: open file multiple times	-80