San José State University
Department of Computer Engineering

CMPE 180-92

Data Structures and Algorithms in C++

Spring 2017
Instructor: Ron Mak

Assignment #11

Assigned:	Saturday, April 15
Due:	Thursday, April 20 at 11:59 PM
URL:	http://codecheck.it/codecheck/files/1704150938cqxxb51yqsfe3prgx1983jkxc
Canvas:	Assignment #11. STL Vector and Map
Points:	100

STL Vector and Map

This assignment will give you practice with the built-in STL map by comparing its performance with the STL vector. Your program will read a text file and build two concordance tables, one from a vector and one from a map.

A concordance table is an alphabetized list of words from a document. The table also stores, for each word, the number of times that word appears in the document. Your input data will be a text file of the U.S. constitution and its amendments:
http://www.cs.sjsu.edu/~mak/CMPE180-92/assignments/11/USConstitution.txt

Your program should read each word of the text. If the word is not already in the concordance, your program should enter it, initially with a count of 1, into both the vector and map. If the word already exists in the concordance, your program should increment the word's count by one in both the vector and the map. The words in the vector and the map should be unique, and, of course, both should end up with the same number of words. Do not include numbers or punctuation marks.

Timings

Your program should keep track of how much time it takes to enter all the words into the vector, and how much time it takes to enter all the words into the map. For each word, it should compute the elapsed time only of the operation of either entering a new word into the concordance or incrementing the count of an existing word. Your program should not include the time to read the word from the input text file. Compare the total time for the vector vs. the total time for the map. The timings should be in microseconds (μsec).

Your program should keep track of how much time it takes to do 100,000 random word searches in the concordance. Again, compare the total time for the vector vs. the total time for the map. Since your vector will be sorted, you should use a binary search.

Other operations

Your program should make (untimed) spot checks of the completed vector and map versions of the concordance to make sure they agree on the frequency counts of some chosen words, including nonexistent words. Your program should also iterate over the completed vector and the map in parallel to ensure that they contain the same data in the same order.

Sample output

Since there are timings, CodeCheck will not compare your output. Your output should be similar to:

```
Timed insertions ...
          Lines:    865
     Characters:  8,420
          Words:  7,541

    Vector size: 1,138
       Map size: 1,138

    Vector total insertion time:   33,555 usec
       Map total insertion time:    2,513 usec

Spot checks of word counts ...
    amendment: vector:35 map:35
    article: vector:28 map:28
    ballot: vector:5 map:5
    citizens: vector:18 map:18
    congress: vector:60 map:60
    constitution: vector:25 map:25
    democracy: vector:(not found) map:(not found)
    electors: vector:16 map:16
    government: vector:8 map:8
    law: vector:39 map:39
    legislature: vector:13 map:13
    people: vector:9 map:9
    president: vector:121 map:121
    representatives: vector:29 map:29
    right: vector:14 map:14
    trust: vector:4 map:4
    united: vector:85 map:85
    vice: vector:36 map:36
    vote: vector:16 map:16

Checking concordances ... both match!

Timed searches (100,000 searches) ...

    Vector total search time:   60,949 usec
       Map total search time:   10,881 usec
```

What to submit

Submit the signed zip file into **Canvas: Assignment 11. STL Vector and Map**. Also submit the text file containing the output from the map and table dumps.

You can submit as many times as necessary to get satisfactory results, and the number of submissions will not affect your score. When you're done with your program, click the "Download" link at the very bottom of the Report screen to download the signed zip file of your solution.

Extra credit (20 points)

Find a hash function that lowers the average number of probes for all three hash tables (with the given table sizes) with the input file. Submit a copy of your `hash()` function in class `HashTable`. Make a separate run using your hash function and submit a copy of the output that shows lower probe count averages. Include map and hash table dumps in this run.

Rubrics

Criteria	Max points
Statistics (should be the same as the sample output) • Total words • Distinct words (vector and map sizes)	**20** • 10 • 10
Word insertions • Vector timing • Map timing	**30** • 15 • 15
Checks • Spot checks (same frequency counts as the sample output) • Matching vector and map concordances	**20** • 10 • 10
Word searches • Vector timing • Map timing	**30** • 15 • 15

Extra credit (10 points max)

Instead of scanning the sorted vector from the beginning to determine where to insert a new word, use a binary search to find the correct insertion position.