9.8 Coding lab #4: File I/Os and STLs: identifying unique word-pairs and their frequencies in a textfile

Honor Code

- Your answers to this homework must be your own work.
- You are not allowed to share your solutions.
- You may not engage in any other activities that will dishonestly improve your results or dishonestly improve or damage the results of others.

Plagiarism

Plagiarism is when you copy words, ideas, or any other materials from another source without giving credit. Plagiarism is unacceptable in any academic environment.

A gentle reminder: This project is similar to the first problem in the algorithm design & pseudo code homework.

Project description:

The attached file, *SteveJobsSpeech2005.txt*, contains the commencement speech delivered by Steve Jobs in 2005 at Stanford University. (A copy of this file is also uploaded to iLearn. You can download it to test your program locally before submitting to zyBook.) In this project, you are going to process this file to identify all the co-occurring word-pairs and their frequencies. Two words are said to co-occur if they appear in the same sentence. We will refer to a pair of co-occurring words as word-pairs. The frequency of a word-pair is defined as the number of sentences that consist of this word-pair. You are required to include three files in this project:

- fileIOs_wordPairs.h: see its content below.
- fileIOs_wordPairs.cpp: implement the functions declared in the above header file
- fileIOs*wordPairs*main.cpp: test all the functions implemented

Header file: fileIOs_wordPairs.h

You are going to include the following function prototypes in this header file (but please feel free to introduce other helper routines if you see necessary) :

1. sentenceSplitter.

```
bool sentenceSplitter( string& fname, vector<string>& sentences);
```

This function converts a text file with the name *fname* into a list of sentences. The list of sentences will be stored in the *sentences* vector in the same order as it appears in the input file. This function returns true if it is successful; false otherwise.

What will be considered as sentence delimiters? Given a paragraph of multiple sentences, the following punctuations will be used to split this paragraph into individual sentences

- period: .,
- question mark: ?
- period + double quotation mark: ."
- question mark + double quotation mark: ?"

Assume the input file contains the following three paragraphs:

```
The first story is about connecting the dots.
```

```
I dropped out of Reed College after the first 6 months, but then stayed around as a drop-in for
another 18 months or so before I really quit. So why did I drop out?

It started before I was born. My biological mother was a young, unwed college graduate student,
and she decided to put me up for adoption. She felt very strongly that I should be adopted by
college graduates, so everything was all set for me to be adopted at birth by a lawyer and his
wife. Except that when I popped out they decided at the last minute that they really wanted a
girl. So my parents, who were on a waiting list, got a call in the middle of the night asking: "We
have an unexpected baby boy; do you want him?" They said: "Of course." My biological mother later
found out that my mother had never graduated from college and that my father had never graduated
from high school. She refused to sign the final adoption papers. She only relented a few months
later when my parents promised that I would someday go to college.
```

The above function will identify a total of 12 sentences as follows:

```
- The first story is about connecting the dots
- I dropped out of Reed College after the first 6 months, but then stayed around as a drop-in for
another 18 months or so before I really quit
- So why did I drop out
- It started before I was born
- My biological mother was a young, unwed college graduate student, and she decided to put me up
for adoption
- She felt very strongly that I should be adopted by college graduates, so everything was all set
for me to be adopted at birth by a lawyer and his wife
- Except that when I popped out they decided at the last minute that they really wanted a girl
- So my parents, who were on a waiting list, got a call in the middle of the night asking: "We
have an unexpected baby boy; do you want him
- They said: "Of course
- My biological mother later found out that my mother had never graduated from college and that my
father had never graduated from high school
- She refused to sign the final adoption papers
- She only relented a few months later when my parents promised that I would someday go to college
```

2. identify unique word-pairs and calculate their frequencies.

```
bool wordpairMapping( vector<string>& sentences, map< pair<string,string>, int>
&wordpairFreq_map);
```

Given a list of sentences stored in the first argument `sentences`, this function identifies all the all the unique word-pairs and each word-pair's frequency. The identified (word-pair, frequency)'s will be stored into `wordpariFreq_map`, which is a map of (key, value) pairs. The key of this map a word-pair and the value is the frequency of this word-pair. This function will return `true` if the mapping is successful; `false`otherwise.

Note that

- Tokens are case insensitive. We will consider lower case in this project
- The two words in a word-pair are different. For example, event though the first sentence above contains two `the`, you are not going to construct a word pair `<the, the>`
- Order does not matter between two words in a word-pair. For example, the word-pair `<college, that>` is the same as `<that, college>`. You are recommended to arrange the two words in lexicographical order before inserting the pair into the map.

Suggestions:

- Use `istringstream` to tokenize a sentence.
- Use `set` to store all the unique tokens identified in a sentence.

Assume `sentences` consists of the following 3 sentences:

```
The first story is about connecting the dots.
The first story is about connecting the dots.
The first story is about connecting the dots.
```

This function is going to identify a total of 21 word-pairs as follows:

```
<about, connecting>: 3
<about, dots>: 3
<about, first>: 3
<about, is>: 3
<about, story>: 3
<about, the>: 3
<connecting, dots>: 3
<connecting, first>: 3
<connecting, is>: 3
<connecting, story>: 3
<connecting, the>: 3
<dots, first>: 3
<dots, is>: 3
<dots, story>: 3
<dots, the>: 3
<first, is>: 3
<first, story>: 3
<first, the>: 3
<is, story>: 3
<is, the>: 3
<story, the>: 3
```

3. flip the map of to a multimap of to order all the word-pairs in ascending order of frequency

```
bool freqWordpairMmap(map< pair<string,string>, int> &wordpairFreq_map, multimap<int, pair<string,
string> > &freqWordpair_mmap );
```

This function flips `the wordpairFreq_map` such that frequencies will be the keys and word-pairs will be the values. A multimap will be needed as two word-pairs can have the same frequency. This function will return `true` if the flipping is successful; `false` otherwise.

4. output the most frequent and least frequent word-pairs to a file.

```
void printWordpairs(multimap<int, pair<string, string> > &freqWordpair_multimap, string outFname,
int topCnt, int botCnt);
```

This function writes the top `topCnt` most frequent word-pairs and `botCnt` least frequent word-pairs to a file of the name `outFname`. Note that all the word-pairs are already ordered in descending order of frequency. You are going to simply use `multimap`'s `iterator` and `revserse_iterator` to access the most frequent and least frequent word-pairs. The output will be one word-pair per line in the format of `<word1, word2>: frequency`. For example:

```
<story, the>: 3
<is, the>: 3
<is, story>: 3
<first, the>: 3
<first, story>: 3
<first, is>: 3
<dots, the>: 3
<dots, story>: 3
<dots, is>: 3
<dots, first>: 3
```

Implementation file: fileIOs_wordPairs.cpp

In this program file, you are going implement all the functions declared in the above header file.

Test driver: fileIOs*wordPairs*main.cpp

You are going to include a main() function to test all the above four functions.

LAB ACTIVITY 9.8.1: Coding lab #4: File I/Os and STLs: identifying unique word-pairs and their frequencies in a textfile 0 / 60

Submission Instructions

Deliverables

`fileIOs_wordPairs.cpp` , `fileIOs_wordPairs.h` and

`fileIOs_wordPairs_main.cpp`

You must submit these file(s)

Compile command

```
g++ fileIOs_wordPairs.cpp fileIOs_wordPairs_main.cpp -Wall -o a.out
```

We will use this command to compile your code

Submit your files below by dragging and dropping into the area or choosing a file on your hard drive.

fileIO...rs.cpp Drag file here or **Choose on hard drive.**

fileIO...airs.h Drag file here or **Choose on hard drive.**

fileIO...in.cpp Drag file here or **Choose on hard drive.**

Submit for grading 75 submissions left This lab can only be submitted once every 1 minute

Latest submission

No submissions yet

Trouble with lab?

Activity summary for Coding lab: File I/Os and STLs: identifying unique word-pairs and their frequencies in a textfile (0 of 60points) Submit to moodle

Due: 03/30/2018, 11:59 PM

Section 9.8 **Hide** keyboard_arrow_up

Lab Activities

9.8 0 / 60