Homework Assignment #10: Programming Project #2

Due Time/Date

2:10PM Tuesday, January 3, 2017. Late submission will be penalized by 20% for each working day overdue.

Task Description

Develop a C++ application, called `myDictionary`, that reads a text file named `knownWords.txt` to set up an initial collection of English words (or Chinese phrases, if you prefer) and then waits to accept and process the user's queries or updates (one by one from the command line).

Each line of the file `knownWords.txt` starts with a word, followed by a blank space and a string that give definitions of the word, or a single dash (`-`), indicating continuation of the definitions for the word in the previous line. Below is a small sample `knownWords.txt` with just two words.

```
curiosity 1. a desire to know about something; 2. something that is
- unusual.
curious 1. interested; 2. unusual or difficult to understand.
```

After setting up the initial collection of words, `myDictionary` enters a loop waiting for the user to type a command such as the following:

- `find curiosity`

  Look up the definitions of "curiosity".

- `new webpage "1. ... 2. ... 3. ..."`

  Add a new word "webpage" with the definitions enclosed in the pair of double quotes to the collection of words.

- `count`

  Count the number of words in the collection.

- `quit`

  Exit the application

Be careful with illegal inputs. When the input is illegal, your program should be able to report an error and resume the state of waiting for the next input.

Submission Guidelines

- Pack everything, excluding compiler-generated files, in a .zip file, named with the pattern "`b047050xx-ds2016-hw10.zip`".
- Upload the .zip file to the Ceiba course site for Data Structures 2016:

  `https://ceiba.ntu.edu.tw/1051ds2016`.
- If you use a Makefile, make sure that it outputs "`myDictionary`". Otherwise, make sure that the whole application can be compiled by the command "`g++ myDictionary.cpp`".

Grading

This assignment constitutes 5% of your grade (of this course). Your work will be graded according to its completeness, correctness, and presentation. You should provide evidences (such as tests) showing that your program is correct. You should also organize and document (by adding comments to) your program in such a way that other programmers, for example your classmates, can understand it. Below is a more specific grading policy:

Criteria	Score
incomplete or doesn't compile	≤ 20
complete, compiles, but with major errors	≤ 40
size of collection is fixed	≤ 60
size of collection may grow	≤ 80
main data structures imported from libraries	≤ 90
main data structures implemented by yourself	≤ 100
own implementation of delete	+5
handle range queries	+5
find the word with a particular rank	+5
tested to hold over $100,000$ words	+5

Notes:

- As an example of deletion, "`delete Webpage`" deletes "Webpage" (and its definitions) from the collection.
- A range query may be posed as "`find word1 word2`". The output should be a list of words between `word1` and `word2`.
- To find the word with a particular rank, the user types a command like "`find 1000`", which would get the 1000-th word (according to lexicographic order) in the collection.
- If you test your application with a `knownWords.txt` containing 100,000 words or more, be sure to provide an evidence, but do not include the large `knownWords.txt` in the submission.