# Homework Assignment #10: Programming Project #2

## Due Time/Date

2:10PM Tuesday, January 2, 2018. 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 their definitions. It then waits to accept and process the user's queries or updates (one by one from the command line).

A word whose definitions are to be given always starts on a new line in the file `knownWords.txt`. The word is followed by its pronunciation in some phonetic symbols (enclosed in '[' and ']'), a part-of-speech (POS) label, and its meanings under that POS class. A single dash (`-`) at the beginning of a line indicates continuation of the definitions of a word in the previous line. A double dash (`--`) indicates an additional POS label for the word. If the word has a different pronunciation under that POS class, it is enclosed in '(' and ')'. Below is a small sample `knownWords.txt` with just three words.

```
contract [...] n. 1. an agreement between two parties. 2. a document
- describing the terms of a contract. 3. ...
-- v. ([...]) 1. to establish or undertake by contract. 2. to reduce to
- smaller size. 3. ...
curiosity [...] n. 1. a desire to know about something. 2. something that
- is unusual.
curious [...] adj. 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 "[...] n. 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 "`b057050xx-ds2017-hw10.zip`".
- Upload the .zip file to the Ceiba course site for Data Structures 2017.
- 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 code 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 | $\leq 20$ |
| complete, compiles, but with major errors | $\leq 40$ |
| size of collection is fixed | $\leq 60$ |
| main data structures imported from libraries | $\leq 80$ |
| main data structures implemented by yourself | $\leq 100$ |
| own implementation of delete | +5 |
| handle range queries | +5 |
| find all POS classes of a word | +5 |
| tested to hold over 10,000 real English 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 all POS classes of a word, the user types a command like "`findpos word`", which would get a list of all POS classes of `word`.
- If you test your application with a `knownWords.txt` containing 10,000 real English words or more, be sure to provide an evidence, but do not include the large `knownWords.txt` in the submission.