CS 610, Spring 2016, Prof. Calvin
Problem set #1
Due: Friday February 26, 5:00 pm.

Exercise (not to turn in). Considering only the dictionary operations insert, remove, and find, give the worst-case total running time and a sequence of n operations that have that running time if the dictionary is implemented as:

(1) an unordered array list,
(2) an unordered linked list,
(3) an ordered array list,
(4) a binary search tree,
(5) an AVL tree,
(6) a skip list.

Homework (to submit on AFS). A *homophone* is one of two or more words that are pronounced alike but are different in meaning or spelling; for example, the words "two", "too", and "to". In this assignment you will write a program that ranks English words by the number of homophones.

The file cmudict.0.7a.txt in

`/afs/cad/courses/ccs/s16/cs/610/common/P1`

contains a pronunciation dictionary downloaded from

`http://www.speech.cs.cmu.edu/cgi-bin/cmudict`

The page also contains a detailed description of the pronunciation dictionary. After an initial segment of comment lines, the file consists of lines of the form

```
ABUNDANT  AH0 B AH1 N D AH0 N T
```

The first string is an English word, which is followed by one or more *phonemes* (or *phones*) that describe the pronunciation of the word. There are 39 phonemes occurring in North American English that are used in the dictionary. The collection of 39 symbols is known as the *Arpabet*, for the Advanced Research Projects Agency (ARPA), which developed it in in the 1970's in connection with research on speech understanding.

Write a Java class called Homophones.java that contains a main method. Your program should take one integer command line argument, call it k. The output of your program should consist of k lines, the ith of which gives a word that has the ith most homophones, followed by the number of homophones.

The purpose of this exercise is to explore choices of data structures that make your program run fast. One of the factors in your score will be the average run time based on three runs of your program.

Failure to abide by the following rules may result in no credit for the assignment.

(1) Your program must read in the file cmudict.0.7a.txt in common/P1. To test your program I will replace that file with a different one (with the same format).
(2) You may use any of the data structures that we have discussed in class.
(3) You may implement your own versions of data structures, or use versions supplied in the standard Java libraries. You may obtain versions from elsewhere and modify them, but in this case write a comment indicating the original source.