22C:16 Homework 9

Due via ICON on Wednesday, April 20th, 4:59 pm

Submit the solutions to all 4 problems, but we will grade some 2 problems of our choice from your submission.

1. I have made available a file called `words.dat` that contains all valid 5-letter English words. A pair of words from this file are called *neighbors* if they differ in exactly one position. For example, `above` and `abode` are neighbors because they differ only in the fourth position. Write a function called `makeNeighbors` that reads `words.dat` and returns a dictionary whose keys are the words in `words.dat` and for each key `k`, the corresponding value is the list of words that are neighbors of `k`. The neighbors of `k` need not appear in any particular order in this list. Also, there is the possibility that some words have no neighbors and therefore the list corresponding to such a word should be empty.

   Note that the function `makeNeighbors` should not have any arguments and it is fully responsible for opening, reading, and closing the file `words.dat`.

2. Write a program that calls the function `makeNeighbors` and uses the dictionary returned by `makeNeighbors` to compute and output the following information:

   (a) All the words with the maximum number of neighbors. For each word with maximum number of neighbors, your program should print all of the neighbors of these words also.

   (b) All the words that have 0 neighbors (if any).

   (c) All pairs of words that are only each others neighbors.

3. This problem is also about the words in the file `words.dat`. Two words x and y in this file are said to be at most i *hops* from each other, if there is a sequence of words

   $$x, w_1, w_2, \ldots, w_{i-1}, y$$

   such that all of these words occur in `words.dat` and each word in the sequence (except the first) is the neighbor of the previous word. Notice that there are $i-1$ intermediate words in this sequence and if you imagine that it takes one hop to go from a word to the next word in this sequence, then it takes i hops to go from word x to word y. For example, the words `stale` and `fresh` are at most 8 hops from each other because of the word sequence:

   ```
   stale stalk stack slack flack flask flash flesh fresh
   ```

   Write a function called `threeNeighborhood` that takes a word `w` in `words.dat` as argument and computes a list of all words in `words.dat` that are at most 3 hops away from `w`.

4. Write a *recursive* function for converting integers in decimal to equivalent binary numbers. Your function should use the following algorithm.

   > If the given integer n is even, then compute the binary equivalent of $n/2$ and append "0" to it. If n if odd, compute the binary equivalent of $n/2$ and append a "1" to it.

   I have deliberately left out any description of the base cases in the above pseudocode. Use the following function header:

   ```
   def recursiveI2B(n):
   ```