Extra Credit Project

You can use this project to make up your CSE-230 score. The highest score you can earn is 100 points. The points you earn from this extra credit project will be added to the total score you earn for quizzes, other projects and exams before your grade is calculated.
Caution: problems with more points are more difficult.

Word Ladders

In this project, you are asked to deal with word ladders – sequences of words in which each word differs from the previous by only one letter. An example of a word ladder from worm to care is ***worm word ward card care.***

Notice that there may be many possible word ladders between any pair of words. Think about the following problems and how they are related to the notion of word ladders:

> Do a Google search for maggon-ismail. What happens?
> When your spell checker comes to a misspelled word, it usually offers you some alternatives.
> Given two images, develop an algorithm to gradually morph one of the images to the other.

We will not solve these problems, but they are related to word ladders. The course website contains an archive all.zip that contains a file dict4.txt which contains approximately all the 4 letter English words. Write a JAVA program which reads the words in dict4.txt into a ArrayList<String> variable. The problems below add further functionality to this program. Name your program and format your output files exactly as the problem asks. For all problems, sample files are found in the archive. **Choose one problem to solve (You will earn points for only one problem because solving a more difficult problem requires you to solve an easier problem).**

1. [4 pts] Program ladder1.java: reads a file words1.txt and writes out a file found1.txt. The file words1.txt contains words (one per line). The file found1.txt should contain the same list of words (one on each line) and next to each word, separated by a space, should be the position in dict4.txt at which it is found. If it is not found, then the position should be -1. Note that the first word in dict4.txt is at position 0.

2. [8 pts] Program ladder2.java: reads a file words2.txt and writes out a file found2.txt. The file words2.txt contains pairs of words (one pair per line). The file found2.txt should contain the same pairs of words (one pair on each line) and next to each word pair, separated by a space is a 1 if the pair is a valid word ladder, and a 0 if not. Note that if any word in the pair does not appear in dict4.txt, then it is not a valid word ladder pair.

3. [12 pts] Program ladder3.java: reads a file words3.txt and writes out a file found3.txt. The file words3.txt contains 4-letter word ladders (one ladder per line). The file found3.txt should contain a 1 or 0 (on each line), corresponding to each word ladder. A 1 if the word ladder is valid word ladder, and a 0 if not. Note that if any word in a ladder does not appear in dict4.txt, then it is not a valid word ladder.

4. [15 pts] Program ladder4.java: reads a file words4.txt and writes out a file found4.txt. The file words4.txt contains a pair of words (one pair per line). The file found4.txt should contain (on each line) one valid word ladder for the pair of words (beginning with the first word in the pair and ending in the second in the pair, there may be many possible word ladders), or a 0 if no word ladder exists. Note that all words in a valid word ladder must appear in dict4.txt.

5. [20 pts] Program ladder5.java: reads a file words5.txt and writes out a file found5.txt. The file words5.txt contains a pair of words (one pair per line). The file found5.txt should contain (on each line) one valid shortest word ladder for the pair of words (beginning with the first word in the pair and ending in the second in the pair, the shortest ladder may not be unique) or a 0 if no word ladder exists. Note that all words in a valid word ladder must appear in dict4.txt. You may also try your algorithms on dict5.txt (a dictionary of 5 letter words) to see how fast they are.