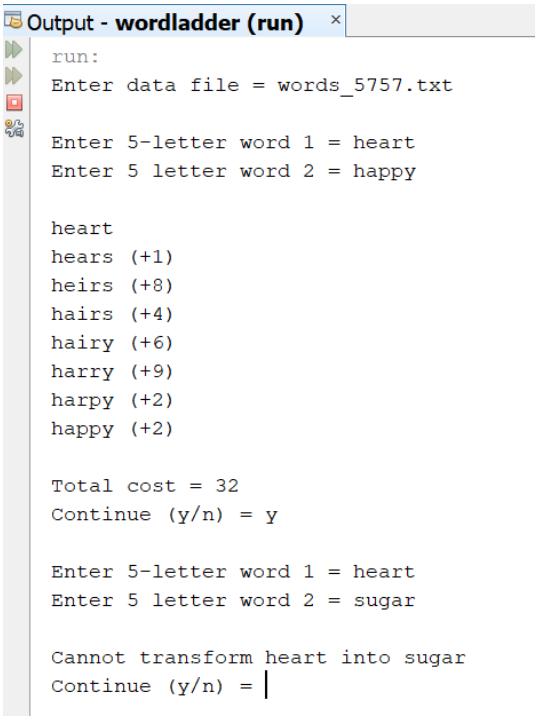
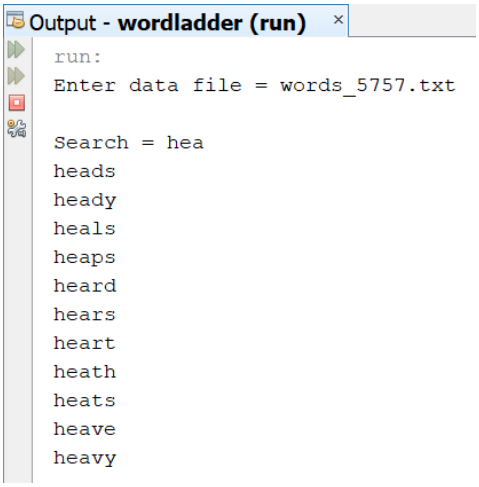
EGCI 221 – Project 2 (Word Ladder)

Given a data set (words_5757.txt), find at least 1 solution to transform one word into another and report all transformation steps. Only 1 character can be changed at a time. The cost of changing = distance between old and new characters.

E.g. hears → heirs distance = 8 as 'a' and 'i' differ by 8 characters





Requirements : the program must be written in Java

1. The program must at least
   - 1.1 Read a data file
   - 1.2 Ask for source and target words from the user. Generate word ladder that transforms source word into target word. Report the total cost of these transformations. In some cases, the program may not be able to transform words. Report that the transformation cannot be done
   - 1.3 Since the data file contains thousands of words, the program must also have a search function that lists all words containing or starting with certain string

2. This word ladder puzzle is another well-known puzzle, with a few different solutions. Some are based on stack/queue, some are based on graph, etc. You can search & use any algorithm (or even pieces of code), provided that the sources are properly acknowledged.

3. Write a report describing at least the following
   - 3.1 Short user manual
   - 3.2 The algorithm to find at least 1 word ladder for given source-target words. Use flowcharts, pictures, etc. & give examples to show that you understand how the algorithm works. Source code listing without any explanation isn't counted as a proper report. Explain at least
     - Which data structure(s) do you use and why ? What values are kept in your data structure(s) ?
     - How do you check if a word can be changed to another word ?

       E.g. why "hear**t** → hear**s**" is OK, but "hea**rt** → hea**ts**" is not ?

- Among many possible words, which criteria do you use to pick the next one in the ladder ?
  E.g. why "heart → hears" is chosen instead of "heart → heard" ?

3.3 Any limitation of your program. Convenient limitation isn't counted as a proper limitation. For example, don't give a limitation that your program will crash if data file is not found just because you are too lazy to handle exceptions

3.4 References. No reference means that you claim you did everything by yourself. If it is found later that you did not, this will be counted as cheating

*** The project can be done in a group of 3-4 students (** group of 1 student is not allowed). **Each group must do it by themselves.** Copying code/report from other group and using other group as your reference, however honest it is, is considered cheating – **everyone involved will get ZERO point**. If you can't design algorithms or implement the whole program by yourselves, at least have some integrity to do proper research from proper sources

Marking

1 point Correct word ladder

1 point Your word ladder gives minimum cost of transformation

1 point Search function

2 points User interface

- This doesn't mean fancy output, but rather how your program is easy to understand & to use even without user manual. For example, can the user continue playing without having to restart the program ? Are the instructions clear enough ? Do you give warning about valid input ? Your program needs not have the same user interface as the example's
- This part must be your own effort. Even if 2 groups take algorithms or pieces of code from the same source, it would be impossible to have the same/similar coding for the user interface. Therefore, same (or too similar) user interface coding is considered cheating

4 points Programming (2 points) + no brute force, with Java collection or JGraphT APIs (2 points)

- Although this is not a programming course, your program should still be well structured – with proper design of classes, proper use of Java collections, proper exception handling
- Even if you use pieces of code from the Internet. Try converting it to proper OOP style

6 points Report. Points are given based on content and format

Submission : due Thursday 28 March, 18.00

1. A report in DOC or PDF. The front page must contain names and IDs of everyone in your group
2. Source files (*.java) and readme.txt containing names and IDs of all members in the group
3. Put all files in a folder. Name the folder after the ID of one member, zip & submit it to rangsipan@gmail.com
   - Write EGCI 221 – Project 2 as the email subject
   - In case of multiple submissions, only the earliest one will be marked