CITS3001 Algorithms, Agents and Artificial Intelligence

Labsheet 5: Word Chess – Assessed

This lab sheet is worth 3% of CITS3001. You should implement Java solutions to the following problems and you should submit them at https://secure.csse.uwa.edu.au/run/cssubmit by 5pm on Sunday 30 August. In accordance with the UWA Policy on Academic Conduct, you may discuss with other students the general principles required to understand this labsheet, but the work you submit must be the result of your own effort.

- A sample data file *Puzzles.txt* is provided (but note that those solutions are not unique). We will test your submission with different data of similar sizes in the same format. You will also need the words file *WWF uppercase.txt*.
- A reasonable amount of time will be allowed for your program to run: for the sample puzzle file you should need no more than about 30–60 seconds. Submissions that run too slowly will be terminated before they complete.
- Submissions that do not compile will earn zero marks. Submissions that have to be edited before they execute will be penalised, and may get zero.
- Normally your mark will depend only on the results produced by your program, but all programs will be compared for similarity with other submissions and with sources drawn from the Internet.
- All questions to *help3001* please.

Download the Lab 5 folder from the LMS, and complete the method in the *WCsolve* class in the code skeleton. You can run the skeleton via the *main* method in the *Lab5* class. **Submit only the file *WCsolve.java* by the due date above.** If you modify any of the other classes, make sure that *WCsolve.java* works with the original version.

A common form of word puzzle is so-called "word chess". Starting from an English word *W*, and changing only one letter at a time, the puzzle is to derive a second word *W'*. All intermediate stages must also be valid English words. So given e.g. SICK → WELL, one solution would be:

SICK → SI**L**K → SIL**L** → S**E**LL → **W**ELL

Other examples include SOFT → LOUD, WEEK → YEAR, CENT → DIME, and DRY → WET. Note that some of these puzzles require more moves than

the number of letters in the words, but it is easy to spot the minimum possible number of moves.

Your method is required to find an optimal solution to each puzzle in the input file, "optimal" here meaning the shortest possible sequence of words. Many puzzles will have multiple optimal solutions; your method can return any of those solutions. If a puzzle is impossible, your method can return anything – but it should ***never*** go into an infinite loop.

You will need the file *WWF uppercase.txt* to check for valid English words and their conjugations. Note that this is not just a dictionary: it includes e.g. *take*, *takes*, *taking*, *taken*, *took*, *etc*. The code framework reads in this file automatically, and it makes its contents available via a final variable declared in *Lab5.java*.

There is a prize available* for Lab 5, separate to the assessment itself. There are many, many word chess puzzles – a prize is offered for the most interesting (preferably original) puzzle emailed to Lyndon before the lab deadline. "Interesting" here could mean anything – have a go!

*All prizes are awarded or not at Lyndon's sole discretion.