University of Glasgow | School of Computing Science

Assessed Coursework

Course Name	Algorithmics 3			
Coursework Number	1			
Deadline	Time:	16:30	Date:	November 18, 2013
% Contribution to final course mark	20%		This should take this many hours:	15
Solo or Group ✓	Solo	✓	Group	
Submission Instructions	Moodle (see details in assignment)			
Marking Criteria	Marking scheme is included			
Please Note: This Coursework cannot be Re-Done				

Code of Assessment Rules for Coursework Submission

Deadlines for the submission of coursework which is to be formally assessed will be published in course documentation, and work which is submitted later than the deadline will be subject to penalty as set out below. The primary grade and secondary band awarded for coursework which is submitted after the published deadline will be calculated as follows:

(i) in respect of work submitted not more than five working days after the deadline
  a. the work will be assessed in the usual way;
  b. the primary grade and secondary band so determined will then be reduced by two secondary bands for each working day (or part of a working day) the work was submitted late.

(ii) work submitted more than five working days after the deadline will be awarded Grade H.

Penalties for late submission of coursework will not be imposed if good cause is established for the late submission. You should submit documents supporting good cause via MyCampus.

Penalty for non-adherence to Submission Instructions is 2 bands

You must complete an "Own Work" form via https://webapps.dcs.gla.ac.uk/ETHICS for all coursework UNLESS submitted via Moodle

Algorithmics 3

Assessed Exercise - Word Ladder - 2013

Notes for guidance. This is the only assessed practical exercise for Algorithmics 3. It carries 20% of the total assessment for the course. As a rough guide, it is intended that an average Level 3 honours student should be able to obtain a B grade by putting in about 15 hours work, and you are advised not to spend significantly more time than this on the exercise.

The exercise is to be done *individually.* Some discussion of the exercise among members of the class is to be expected, but close working together, or copying of code, in any form, is strictly forbidden refer to the Department's Plagiarism Policy and Guidelines in the Level 3 Class Guide (available via moodle).

Deadline for submission. The hand-out date for the exercise is **Friday 18 October**, by which time all the relevant material should have been covered in lectures.

The deadline for submission is **16:30 Monday 18 November**.

There will be lab sessions in the weeks starting **Monday 4 November** and **Monday 11 November**, during which you will have the opportunity to ask questions on the exercise, and discuss your progress with the course coordinator.

Specification. The purpose of the exercise is to write, in Java, programs to investigate word ladders composed of five letter words. A word ladder is a sequence of words, each member of the sequence differing from its predecessor in exactly one position. For example, the following ladder, of length 6, transforms' the word *flour* into the word *bread*:

flour → *floor* → *flood* → *blood* → *brood* → *broad* → *bread*

A dictionary file `words5.txt` will be provided, which contains a set of nearly 2000 five-letter words that should be used to construct ladders.

The first program should read in a dictionary file, together with two more five letter words, i.e. the program should take 3 command-line arguments:

1. a dictionary file;
2. a start word;
3. an end word.

The program should produce on the standard output channel the length of the shortest path and a path/ladder of shortest length that transforms the start word into the end word, or should report that no ladder is possible. The final line of output should report the execution time of the program in seconds. (The code to generate this output is included in the skeleton programs provided. Note that it represents elapsed time, so may not be an accurate reflection of actual running time depending on other processes that may be executing on the computer.)

The second program considers a weighted version of the word ladder problem where the weight of a transformation (i.e. edge of the corresponding graph) is the absolute difference in the positions of the alphabet of the non-matching letter. For example, the weight of the edge between *angel* and *anger* equals the position of r minus the position of l which is 6.

The program should implement Dijkstra's algorithm for finding the shortest paths. Similarly to the first case, the program should read in a dictionary file, together with two more five letter words the program and report on the standard output channel the minimum distance between the words together with a corresponding path, or should report that no ladder is possible. As for the first program, the final line of output should report the execution time of the program in seconds.

Clarifications.

- the dictionary file (`words5.txt`) contains only words of 5 letters, all in lower-case, one word per line;
- no data validation is needed: you can assume that input to the first program is provided in the appropriate format, with all words in lower case;
- you can assume also that each word that is input is actually present in the given word file;
- a graph representation of the dictionary is the key to an efficient solution;
- graphs should be represented using adjacency lists and the program from the warm up laboratory exercise provides a very good basis for you programs.

Submission. The set up files for the exercise are available under Moodle (these include skeleton program files, a template report file, dictionary file of words and test data). (As a starting point, it would make sense to copy across the files for the classes `AdjListNode`, `Vertex` and `Graph` developed in the laboratory exercise.) You may wish to create additional small input files for test purposes.

Submit an archive of your work through Moodle, the archive should contain the following:

- A pdf file `report.pdf` generated from the `report.tex` file, containing:

  - a status report, which should state whether you believe that your programs work correctly, and if not what happens when the program is compiled (in the case of compile-time errors) or run (in the case of run-time errors or incorrect output);
  - a written discussion justifying any implementation decisions;
  - the output produced by your programs the test data provided.

- Listings of all the relevant Java source code, with appropriate comments;
- Folders `WordLadder` and `Dijkstra` containing all your `.java` files for the two programs.
- In each folder there should be a class `Main.java` containing your main method; apart from this there can be any number of other `.java` files and other folders corresponding to packages if you wish. But ensure that any redundant files are removed.

Marking scheme. The exercise carries 30 marks (then mapped to a band), distributed as follows:

- Word ladder implementation (program 1): **10 marks**
- Dijkstra's shortest path algorithm (program 2): **10 marks**
- Report, quality of submitted code, overall presentation: **8 marks**
- Outputs from test data: **2 marks**

This is primarily an Algorithmics exercise, rather than a Software Engineering exercise, but you may be penalised, under the third heading above, for poor software engineering practice.