University of Glasgow | School of Computing Science

Assessed Coursework

Course Name	Algorithmics I (H)			
Coursework Number	1			
Deadline	Time:	16:30	Date:	15/11/2021
% Contribution to final course mark	20%			
Solo or Group ✓	Solo	✓	Group	
Anticipated Hours	20 hours			
Submission Instructions	Moodle (see details in assignment)			
Please Note: This Coursework cannot be Re-Assessed				

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://studentltc.dcs.gla.ac.uk/ for all coursework

Algorithmics I

Assessed Exercise – Text Compression Ratios

Notes for guidance. This is the only assessed practical exercise for Algorithmics I. 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 Plagiarism Policy and Guidelines in the Undergraduate Class Guide (available via moodle).

Deadline for submission. The hand-out date for the exercise is **Friday 15 October**, by which time all the relevant material should have been covered in lectures.

The deadline for submission is **16:30 Monday 15 November**.

The lab sessions in the weeks starting **Monday 1 November** and **Monday 8 November**, during which you will have the opportunity to ask questions on the exercise and discuss your progress with the course coordinator and lab assistant.

Specification. The purpose of the exercise is to write, in *Java*, two programs, each of which will accept as input an arbitrary text file, and will produce as output the compression ratio that would be achieved if that file were to be compressed using (a) in the first case, the `Huffman` algorithm, and (b) in the second case, the `LZW` algorithm. *Note that no actual compressed files are to be produced. Your programs should do the minimum amount of work necessary to determine the compression ratio in each case.*

In the case of the `Huffman` algorithm (based on a frequency count of the characters in the text file to be compressed), you should ignore the fact that, in practice, some representation of the `Huffman tree` would have to be stored with the compressed file (so that the compression ratio that you obtain will be a slight over-estimate of what could actually be achieved).

In the required version of the `LZW` algorithm, compression should begin with codewords of length 8. Whenever the dictionary becomes full the codeword length should be increased by 1, thereby doubling the size of the dictionary. There is to be no upper limit on the length of codewords.

Clarifications.

- The program need work correctly only text files that are limited to the standard ascii character set.
- Assume that each character, including '`\n`' (new line), occupies 1 byte in uncompressed files.
- The name of the file to be compressed should be given as the only program parameter. The only output produced by your programs should take the following form:

```
Original file length in bits = 12688
Compressed file length in bits = 5926
Compression ratio = 0.4671
Elapsed time:  2 milliseconds
```

- The last line of output reports the execution time of the program in milliseconds. 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.
- For the `LZW` program, you should use a `Trie`, appropriately customised, to represent the dictionary. The `Node` and `Trie` java classes and functions from the laboratory exercise can be used, *with appropriate changes that will be necessary if you wish your program to be efficient*.

Submission. The set up files for the exercise are available under Moodle (these include skeleton program files, the `Node` and `Trie` classes used in the laboratory exercise, a template report file and three sample input files). You may wish to create additional small input files for test purposes.

Submit an `.zip` archive of your work through Moodle. The archive should expand into a directory named after your 7-digit matriculation number. You can create such an archive by using the command:

```
zip -r 0123456.tar.gz 0123456/
```

This directory 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 explaining how, in each case, your program obtains the information needed to calculate the compression ratio and justifying any implementation decisions;
  - the output produced by your programs the test data provided.
- Folders `Huffman` and `LZW` 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.

Please make sure you follow the submission instructions - the penalty for non-adherence to Submission Instructions is 2 bands.

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

- `Huffman` implementation, correctness and efficiency: **10 marks**

- `LZW` implementation, correctness and efficiency: **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.