University of Glasgow | School of Computing Science

Assessed Coursework

Course Name	ANC4			
Coursework Number	1			
Deadline	Time:	4:30pm	Date:	12th March 2018
% Contribution to final course mark	20		This should take this many hours:	
Solo or Group ✓	Solo	✓	Group	
Submission Instructions	See "Deliverables"			
Who Will Mark This? ✓	Lecturer ✓		Tutor	Other
Feedback Type? ✓	Written ✓		Oral	Both
Individual or Generic? ✓	Generic		Individual ✓	Both
Other Feedback Notes				
Discussion in Class? ✓	Yes		No ✓	
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

Marking Criteria
The exercise is worth 20% See below for division of marks

ANC4: Assessed Exercise 2018

Description

Consider a simplified point-point network description consisting of a set of nodes {N1, N2, ….} and a set of links with costs, of the form (N1,N2,C). In addition, assume every node has a routing table with 3 columns: destination, distance and outgoing link.

With a programming language of your choice* implement an application which will read a **text-file** containing a simple human-readable description of such a network and then simulate distance-vector routing with *synchronised* periodic exchanges (i.e. at regular intervals, all nodes exchange vectors simultaneously), updating all node routing tables over any selected number of iterations.

A user should be able to:

- compute routing tables for any preset number of exchanges or until stability is achieved;

- preset any link to change cost or fail after any chosen exchange (you may assume for simplicity that neighbours notice cost changes or unreachable neighbours immediately);

- view the best route between any source and destination after any chosen iteration

- trace the routing tables of any set of nodes for any specified number of iterations in a way that can be easily viewed;

- engage, on request, a *split-horizon* capability to help combat slow convergence

- run the application under Windows 10 x64 (an **easy-to-use** command line interface is OK).

Once your simulator is working, document a walkthrough of **two** simple examples with appropriate input networks to illustrate how some exemplar routes and tables evolve under:

- normal convergence of the distance-vector algorithm;

- slow convergence both **with and without** the split horizon facility switched on.

Describe clearly but briefly, with the help output from your application, what is going on as the routes evolve. **Use diagrams where appropriate to aid your description.**

Deliverables.

Submit the following in a zip or rar file attachment via email to Lewis.Mackenzie@glasgow.ac.uk by Monday 12th March. The subject heading in your email must be: "ANC4 assessed exercise".

1. A short (< 1 page) informal overview of your high-level application design and a short user manual (< 1 page) including instructions to install and run the program*.   (~10%+10%)

2. A well-commented source listing (informative commenting is what counts). Give clear instructions on how to build the executable.   (~10%)

3. A functioning executable file which MUST be derivable from 2. Marks are gained for implementing the required functionalities and for an application which is easy to use. A graphical interface is not necessary (although you may provide one if you wish) but *usability* is important.   (~*30%*)

4. For your two example networks, give input description files and, using words, diagrams and output from your application, a documented "walkthrough" of how some exemplar tables and routes evolve as each iteration completes. Marks here will be awarded for ***clarity and brevity***.   (~*30%*)

5. A (1-page max) status report discussing any limitations of your program and how you would improve it given the time.   (~10%)

All documents must be in ***Acrobat PDF format***. Marks will be awarded for ***clarity and brevity*** of presentation. If you do not provide adequate instructions for building or use of your application, or if your output is hard to interpret, you will be marked down.

Items 1, 2, 4 and 5 should also be submitted in hard copy in the box provided in Lilybank Gardens.

* NOTE: Although you may use any programming language you like, you must ensure your executable will *run* without requiring the user to install any additional <u>commercial</u> software. If ANYTHING else (e.g. a JVM) is needed which would not be present on a freshly installed copy of Windows 7 (or 10) x64, you MUST include instructions about how to get it and install it!