UNIVERSITY of York

Department of Electronic Engineering

ELE00028I Algorithms & Numerical Methods Coursework Assessment 2021/22

SUMMARY DETAILS

This coursework **(Programming Assignment)** contributes **50%** of the assessment for this module.

Clearly indicate your **Exam Number** on every separate piece of work submitted.

Submission is via the VLE module submission point. **The deadline is 12:00 noon on 19 April 2022, Summer Term, Week 1, Tuesday.** Please try and submit early as any late submissions will be penalised. Assessment information including on late penalties is given in the Statement of Assessment.

ACADEMIC INTEGRITY

It is your responsibility to ensure that you understand and comply with the University's policy on academic integrity. If this is your first year of study at the University then you also need to complete the mandatory Academic Integrity Tutorial. Further information is available at http://www.york.ac.uk/integrity/.

In particular please note:

- Unless the coursework specifies a group submission, you should assume that all submissions are individual and should therefore be your own work.
- All assessment submissions are subject to the Department's policy on plagiarism and, wherever possible, will be checked by the Department using Turnitin software.

Department of Electronic Engineering, University of York
ELE00028I: Algorithms and Numerical Methods
Coursework Assessment 2021/2022

Task: Design, implement and test a program that constructs a graph of cities and energy expended in travelling between them, then outputs the paths with minimal energy consumption between pairs of specified locations.

The context of this assessment is the future when electric cars are the norm and so is battery charging on-the-go with the use of wireless chargers under the roads. The chargers can sense the presence of a car above them and charge as the car passes over, resulting in a net positive energy gain (or a net negative loss of energy) of the battery. The "*energy*" data file provided, lists the energy loss incurred when travelling from one city to the other. Some roads have wireless chargers embedded under them, thus resulting in a net negative amount of energy expended (or net energy gained).

The objective of this assessment is to choose the path of least energy expenditure or even maximum energy gain, given a source and a destination city. All the paths connecting source and destination cities are two-way roads i.e. the car can go both ways and will expend the same energy going from source to destination and vice versa. However, in the best path, the car should visit each city only once (to prevent a chance of looping around the same city constantly to gain energy).

You are to write and verify a program in C that implements a graph abstract data type (ADT) suitable for the processing of information about energy spent in travelling between cities, and that returns the path between two cities that leads to least energy loss. It should read and store information from a tab-delimited "*energy-v22-1*" file in which each line consists of two cities and the net loss/gain in energy in travelling between them. A printout of the file is appended. An electronic version (which should be used in the assessment) can be found on the course website at

`wiki.york.ac.uk/display/EE/Algorithms`

– under the "Assessment" section.

The program should read in a second tab-delimited file, "*citypairs*" in which each line consists of two cities. For each pair of cities, it should find the best path between those cities and print out a list of the cities on the route and the overall energy spent. You should run your program and include the output in your report for a file containing the following cities:

Lincoln	Perth
Doncaster	Whitby
Bristol	Blackpool

You must decide the most appropriate data structure and algorithms. As discussed in class a very good implementation will:

- Use a well-designed graph ADT that localizes information
- Use a shortest path algorithm, such as Bellman-Ford algorithm
- Take care of negative cycles in the graph

This assignment counts for 100 % of the total mark of Algorithms and therefore 50 % of the total mark of ELE00028I Algorithms and Numerical Methods.

Academic Conduct

This is an individual assignment – you must work on the program and report on your own. You may consult any sources of information you can find providing you give appropriate reference and attribution to the authors.

Borrowing small reusable pieces of useful code from other sources is acceptable. However, wholesale copying or reworking of a complete program written by someone else is against the rules. If you use fragments of code written by anyone else, you must make it clear in both your report and the code which part is yours and which has been written by the other person.

If the rules on what constitutes correct academic conduct are not clear, please consult the university regulations Academic Misconduct as indicated on the front sheet for this assignment.

Submission

You are to write your program in standard ANSI C using Code::Blocks and the minGW (gcc) C compiler. The program must compile and execute correctly on the university lab machines available via the virtual desktop service (so the "local install" version of this is also acceptable).

Submission will be electronic via the VLE in a zipped folder containing two parts:

1. A PDF report (**no more** than 6 A4 pages in length).
2. The Code::Blocks project folder containing at least
   (a) A `.cbp` Code::Blocks project file
   (b) A `src` folder containing the C source code of your program
   (c) Any data files you create / use as input to your program

The written report should explain your design, your choice of data structures and algorithms used in the program and should include consideration of memory requirements and computational efficiency. It should clearly explain how you tested the program. Include the output from your program for the cases described above.

You should implement the program in the form of one or more C modules, i.e. with a `.h` interface file, a `.c` implementation file and a main `.c` program.

Your program does **not** need an interactive user interface (and certainly not a graphical user interface). It is acceptable, for example, to invoke your program through a command line interface like "*myprogram energy citypairs output*" where *myprogram* is the name of your program, *energy* is the name of the provided text file of energy loss between cities, *citypairs* is the name of the text file saying which particular start and end cities are to be calculated, and *output* is the name of a text file in which to write the output. **But**, appropriate default values for the file names should be assumed by the program.

NOTE: Anonymised marking – use your exam number only

Marking will be done anonymously so **DO NOT** put your name on or in any of the submission parts. Instead, your report, code comments and filenames should identify you using your exam number only.

Marks Allocation

In report:

Design choices and justification.	/ 20
Testing methodology, results and comment.	/ 20

In code:

Correctness – syntax, logic, function.	/ 20
Clarity and maintainability – structure, style, comments.	/ 20

In both:

Program match to report, including verification of output for given city-pairs file and efficiency as claimed.	/ 10
Program match to spec – complete, correct, including correct output for an unseen citypairs file.	/ 10

Data file

```
York        Hull        60
Leeds        Doncaster        -47
Liverpool        Nottingham        161
Manchester        Sheffield        -61
Reading        Oxford        -43
Oxford        Birmingham        103
Birmingham        Leicester        63
Liverpool        Blackpool        79
Carlisle        Newcastle        95
Nottingham        Birmingham        -77
Leeds        York        39
Glasgow        Edinburgh        -74
Moffat        Carlisle        65
Doncaster        Hull        76
Northampton        Birmingham        90
Leicester        Lincoln        82
Sheffield        Birmingham        122
Whitby        Newcastle        116
Glasgow        Perth        93
Lincoln        Doncaster        63
Sheffield        Doncaster        29
Cardiff        Bristol        -71
Bristol        Reading        130
Hull        Nottingham        145
Blackpool        Leeds        116
Birmingham        Bristol        139
Manchester        Leeds        64
Carlisle        Blackpool        140
Birmingham        Liverpool        126
Perth        Edinburgh        83
Leicester        Northampton        61
Newcastle        York        135
Glasgow        Moffat        -28
Leicester        Sheffield        100
Carlisle        Liverpool        -30
Birmingham        Manchester        129
Birmingham        Cardiff        172
Oxford        Bristol        116
Leeds        Hull        89
Edinburgh        Carlisle        154
Nottingham        Sheffield        61
Liverpool        Manchester        56
York        Whitby        78
Carlisle        Glasgow        50
Sheffield        Lincoln        74
York        Doncaster        55
Newcastle        Edinburgh        177
Leeds        Sheffield        53
Northampton        Oxford        68
Manchester        Carlisle        20
```