Mathematics, C++ Assessment 2, 2014-5

This assessment is worth 100% of the total marks available for C++ programming. The hand-in date is 2 April 2015.

Finding the fastest route between two nodes on a graph
You are given a graph where each edge is labelled with a length and a speed. Given a start node A and an end node B, find the fastest route between A and B.

Figure 1 presents an idealized road network. Edges are the roads and the nodes represent road junctions. The distance along each road between junctions is known as is the average speed that vehicles can travel along each section of road. These are given in Table 1. Speed varies considerably depending on, for instance, speed limits and whether the road is urban, a motorway, or some other road type. Speeds also depend on the digits in your ID number. See Table 1. Roads can be traversed in either direction.

You must write a C++ application to find the fastest route from node A to node B. It must output a description of the route and the time taken along that route. After finding the optimal route between A and B on the full network, now suppose that the M6, J2, to M69, J1, link is closed due to road works. Now what is the fastest route?

A graph can be specified, amongst other ways, as an incidence matrix, or as a linked set of nodes and edges. You must reify the graph into a suitable C++ data structure. The graph's reification must enable data to be attached to edges.

Pay attention to the way the specifics of the graph are written into your code so that they are (potentially) easy and intuitive to change. The application must be able to work with any (connected) network.

There are advantages in making the application object oriented (for instance you might consider devising edge, node, and route objects) but it is not required that you should do so. Where possible you should avoiding using C-style arrays. You may wish to consider using recursive function calls.

The emphasis for the assessment is on the quality of your code, not on the sophistication of the graphical algorithm you have implemented. A truncated tree search in which you eliminate patently suboptimal routes, such as those that return to a node already visited, would be acceptable. The implementation should be computationally efficient. It should not perform redundant or duplicated calculation.

You should construct a clear user interface and write code in a clear and maintainable style. The code you submit must to be able to run without modification or addition with the DevCpp installation on the machines in the teaching area.

The report
Your report must include a clear description of the algorithm you implement and an assessment of your results. See the general requirements. You will be assessed on

a) The quality of your code in terms of readability, maintainability, clarity, sophistication, generalizability, and general presentation.
b) The degree to which your code has the required functionality.
c) Your assessment of the efficacy of your implementation.
d) The presentational quality of the assessment as a whole.

Group work is not permitted
You are encouraged to discuss the assessment with other students but the code you submit must be yours alone. Clear similarities in code, in the write-up or in the results, will be taken as evidence of group work (for instance if identical or very similar tables of results are given, or if code is clearly shared.)

Code closely modelled on that from other sources is not permitted
Code to perform components of this exercise is likely be found in various places and might, for instance, be downloadable from the web. Use of code taken from, or cosmetically altered from, such sources is not permitted. You must devise your code from scratch.

Specifically *excluded* from this prohibition is code I have declared that you may use. In particular you may use the library code I have made available, without attribution.

Nick Webber

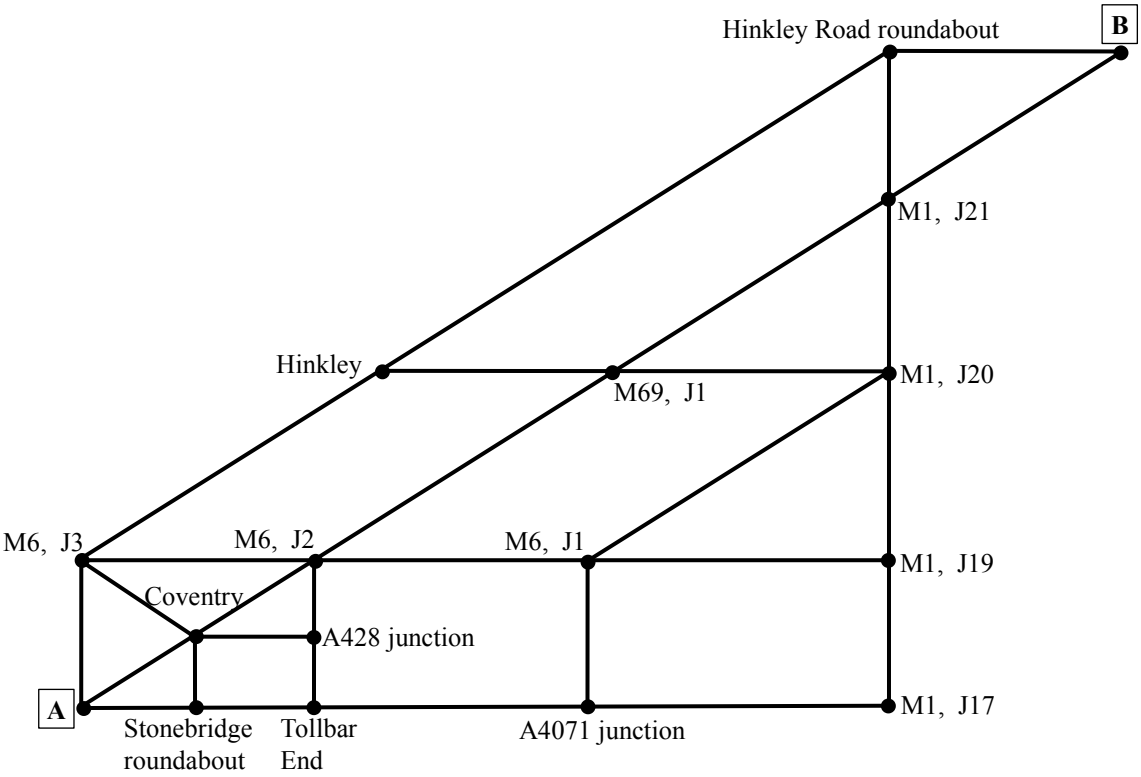

Figure 1 An idealized road network

Table 1 Road distances and speeds [a]

Road	Distance (ml)	Speed (ml/hr)
A to M6, J3	3	30 + x
A to Coventry	3	20 + y
A to Stonebridge roundabout	4	40 – x
Stonebridge roundabout to Coventry	2	30 – y
Stonebridge roundabout to Tollbar End	2	40 + x
Tollbar End to A428 junction	2	40 + y
Tollbar End to A4071 junction	7	60 – x
A4071 junction to M6, J1	6	40 – y
A4071 junction to M1, J17	10	70 + x
M1, J17 to M1, J19	6	70 + y
Coventry to M6, J3	3	30 – x
Coventry to M6, J2	3	30 – y
Coventry to A428 junction	3	20 + x
A428 junction to M6, J2	3	70 + y
M6, J3 to Hinkley	4	60 – x
M6, J3 to M6, J2	3	70 – y
M6, J2 to M69, J1	6	60 + x
M6, J2 to M6, J1	8	70 + y
M6, J1 to M1, J20	3	50 – x
M6, J1 to M1, J19	3	70 – y
M1, J19 to M1, J20	3	70 + x
Hinkley to Hinkley Road roundabout	14	50 + y
Hinkley to M69, J1	3	60 – x
M69, J1 to M1, J21	10	70 – y
M69, J1 to M1, J20	10	50 + x
M1, J20 to M1, J21	11	70 + y
M1, J21 to Hinkley Road roundabout	2	30 – x
M1, J21 to B	2	20 – y
Hinkley Road roundabout to B	1	20 + x

[a] x is the last digit in your ID; y is the penultimate digit.