CSCI203/CSCI803 ASSIGNMENT 3 Due 23:55 29/10/2017

This assignment involves extension to the single source - single destination shortest path problem.

The Program

Your program should:

1. Read the name of a text file from the console.
2. Read a graph from the file.
3. Find the shortest path between the start and goal vertices specified in the file.
4. Print out the vertices on the path, in order from start to goal.
5. Print out the length of this path.
6. Devise a strategy for determining the second shortest path between the vertices.
7. Find the second shortest path between the start and goal vertices specified in the file.
8. Print out the vertices on the path, in order from start to goal.
9. Print out the length of this path.

The data files are constructed as follows:

- Two integers: nVertices and nEdges, the number of vertices and edges in the graph.
- nVertices triples consisting of the label and the x- and y-coordinates of each vertex.
- nEdges triples consisting of the labels of the start and end vertices of each edge, along with its weight. Note: the weight associated with an edge will be greater than or equal to the Euclidean distance between its start and end vertices as determined by their coordinates.
- Two labels, the indicating the start and goal vertices for which the paths are required.

A proposed solution to the second shortest path problem is as follows:

```
For each edge $e_i$ on the shortest path:
    Find the shortest path on $(V, E – \{e_i\})$.    // shortest path without edge $e_i$
The shortest of these is the second shortest path.
```

Questions

Is this proposed solution correct?

1. If we require that the second shortest path be longer than the shortest path?
2. If the graph contains cycles?
3. If the graph is undirected?

In each case explain your answer and, if necessary explain how you might modify the proposed algorithm to address any issues you identify.

Note: you may implement either the proposed solution or any modification you develop. You are not required to implement a modified proposal if you do not wish to do so.

Programs must compile and run under gcc (C programs), g++ (C++ programs), java or python. Programs which do not compile and run will receive no marks.

Programs should be appropriately documented with comments.

All coding must be your own work. Standard libraries of data structures and algorithms such as STL or its Java equivalent may not be used, nor may code be sourced from textbooks, the internet, etc.

Marking Guide:
Programs submitted must work! A program which fails, to compile or run will receive a mark of zero for the program.
A program which produces the correct output, no matter how inefficient the code, will receive a minimum of 50% of the mark for the program.
Additional marks beyond this will be awarded for the appropriateness, i.e. efficiency for this problem, of the algorithms and data structures you use.
Programs which lack clarity, both in code and comments, will lose marks.
50% of the marks for this assignment will be assessed from the program, the other 50% will be assessed from your pdf document.

Submission:
Assignments should be typed into a single text file called ass1.*ext* where *ext* is the appropriate file extension for the chosen language. A pdf file describing your solution should also be produced. This file should contain at least:

a) The documentation for your program
   1. A high-level description of the overall solution strategy chosen for the shortest path problem.
   2. A high-level description of the design and overall solution strategy you have developed for the second shortest path problem.
   3. A list of all of the data structures used, where they are used and the reasons for their choice.
   4. A list of any algorithms used, where they are used and why they are used.

b) The answers, explanations and possible changes for each question in the questions section.

Both files should be submitted via the submit program.

```
submit -u user -c csci203 -a 3 ass3.ext ass3.pdf
```

where your unix userid should appear instead of *user*.