CSP 609/CSP 509 - PG SOFTWARE LAB
Lab Assignment 2
Weightage: 2.5%
Submission Deadline: 16-August-2018 12:00 noon.

General Instructions:

- You must use only C, C++, or JAVA for this assignment.
- Prescribed specifications must be strictly followed. Failure to do so may lead to substantial loss of points.
- Make sure your code is well written (self explanatory variable names) and documented. You are likely to lose points if your TA cannot understand your code.
- You are not allowed to use any libraries. You are allowed to use only on the primitive data types in the language such as arrays, structures, classes, etc.
- **Code will be evaluated for its efficiency in terms of memory and time requirements. For example, allocating unnecessary space may lead to loss of points. Similarly, creating duplicate copies of same data (inside the program) may also lead to loss of points.**

Programming Question (50 points):

For this assignment, you are expected to implement an Dijkstra's on directed graphs. You may use the adjacency list code you developed in previous assignment (for this reason weightage of this assignment is kept low).

Description of Input:
The input would consist of the following two files. Your code may assume that these files are present in the same directory as your code.

Nodes.txt -- This file contains information about the nodes of the graph. Data in the file is according to the following scheme

<Node id as an positive integer> <new line>
<Node id as an positive integer> <new line>
….
Following can be considered as a sample Nodes.txt

343234
8223
123121
….

Edges.txt --- This file contains information about the edges of the graph. Data in the file is according to the following scheme

<head Node id>   <tail Node id> <edge weight as a positive integer> <new line>
<head Node id>   <tail Node id> <edge weight as a positive integer> <new line>
….

Following can be considered as a sample Edges.txt

343234 8223 23
123121 8223 89
….

Function to be implemented:
A Dijkstra's routine which takes the input graph, a source and a destination and outputs a shortest path between the two. Your output should include both the path (the NodeIds) and the length of the path. In case the destination is not reachable from the source, your code should print an appropriate message.

Further instructions

- Design your code while considering its scalability. For example, you may note that adjacency matrix representation of the graph is usually not scalable beyond 500,000 nodes.
- **Input Graph may have more than one connected component.**
- **Cover all cases, e.g., some nodes may have zero in-degree or out-degree or both.**
- **You should not use specialized libraries for storing graph and/or running Dijkstra's. You may use basic data structures like structures, arrays, hash_maps and associative arrays.**
- Put your name and roll number in all the source code files.
- For evaluation, the TA would put your code in a directory with some test datasets and run your code.
- Please provide compilation instructions along with the code.