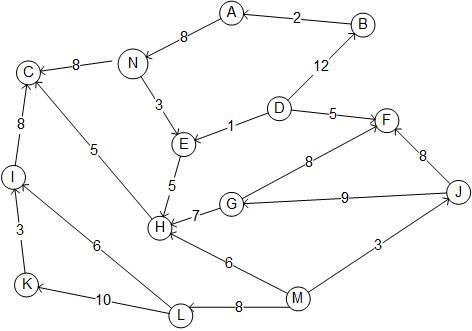
**Program 7 – 20 points CS 2420 – Fall 2014**

**Graph algorithms**

Given the graph shown below, write a C++ program that inputs the graph, stores it as an adjacency list, and produces the desired output. No starter code will be given. You should write your own algorithms. Do not copy from any other source.

Use the file prog7A.txt as input. The format of the file is as follows:

* The first line contains the number of nodes and the number of edges. The names of the nodes are A to N
  + For each edge, the endpoints in order and the weight of the edges
* prog7a.txt contains the description of the following graphing.



1. (5 points) Topological ordering. The output will consist of the node names followed by their topological order number (starting with 1). Note, there are multiple correct orderings. For topological ordering, **ignore edge weights.** If there is no topological ordering, note that fact.
2. (5 points) For the above graph, print the nodes in order of a depth first traversal of the graph. You will have to begin the depth first traversal multiple times (at nodes without predecessors) to get all nodes (as there is not one unique starting point). Use a visited flag at each node so you don't list a node more than once.
3. (5 points) Create an **undirected weighted graph from the input of prog7.txt.**  For this graph, **ignore the direction of the edges.** Use Dijkstra’s algorithm to find the path of shortest distance between the start node (N in our case) and every other node. Your output should look something like:

N:A path is N->A Total Cost 8

N:B path is N->A->B Total Cost 10

N:C path is N->C Total Cost 8

N:D path is N->E->D Total Cost 4

(cont)

1. (5 points) Using the input of prog7.txt (**ignoring direction of edges**), list the edges of a minimal spanning tree using Prim’s algorithm. Output the total weight of the minimal spanning tree.

Repeat each algorithm with graph prog7b.txt.

Hints:

You will use an adjacency list to store the graph. You will want to be able to convert between a nodeID (A-N) and a subscript (0-13). An easy way of doing this is to compute nodeId-'A'. By subtracting the letter 'A' from the character representation, you get the corresponding integer.