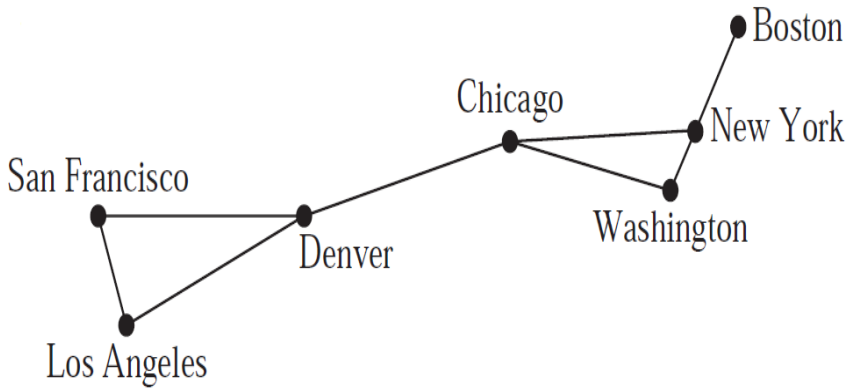
Fundamental Structures Lab 12

(Due Date: 04/25/2016 11:59 pm)

Program 1 - 15 pts - Undirected graph



Consider the above undirected graph and represent the graph using an adjacency matrix. While creating the adjacency matrix, use the state names in alphabetical orders. Instead of using the whole names (e.g. New York), use the first letters (e.g. N) of the states to identify them. Format of adjacency matrix will be (the first row is done for you):

	B	C	D	L	N	S	W
B	0	0	0	0	1	0	0
C							
D							
L							
N							
S							
W							

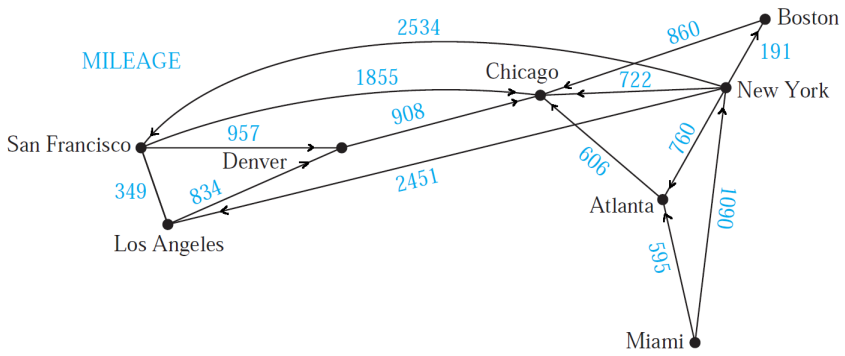
Create a menu driven application which will provide the following options to the users:

1. **Insert:** This option will allow users to add an edge to the existing graph model. To add an edge, take two vertices as input from the user, update the adjacency matrix using that two vertices as index. If an edge already exists, notify the user by saying "edge already exists!"

2. **Delete:** This option will allow users to delete an edge to the existing graph model. To delete an edge, take two vertices as input from the user, update the adjacency matrix using that two vertices as index. If there is no existing edge, notify the user by saying "no edge available!"

3. **Show total number of edges:** Show total number of edges using the adjacency matrix.
4. **Show the matrix:** Show the updated adjacency matrix in the console.
5. **Exit**

Program 2 - 15 pts - Directed graph



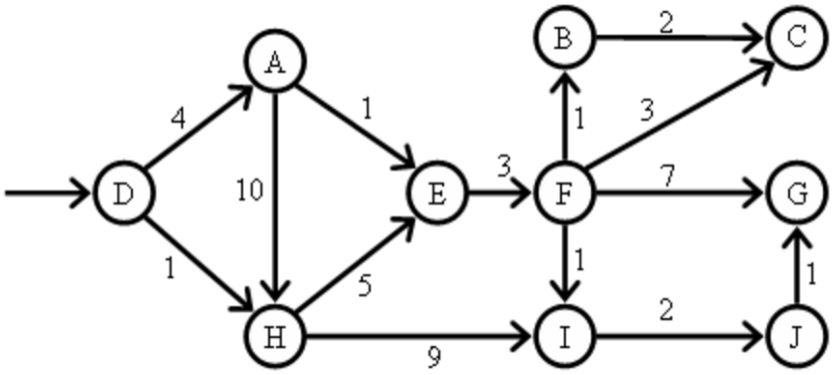
Consider the above directed weighted graph and represent the graph using an adjacency matrix. While creating the adjacency matrix, use the state names in alphabetical orders. Instead of using the whole names (e.g. New York), use the first letters (e.g. N) of the states to identify them. Format of adjacency matrix will be (the first row is done for you):

	A	B	C	D	L	M	N	S
A	0	0	606	0	0	0	0	0
B								
C								
D								
L								
M								
N								
S								

Create a menu driven application which will provide the following options to the users:

1. **Insert:** This option will allow users to add an edge to the existing graph model. To add an edge, take two vertices (the edge is from first to second) and the edge weight as input from the user, update the adjacency matrix using that two vertices as index. If an edge already exists, notify the user by saying "edge already exists!"
2. **Delete:** This option will allow users to delete an edge to the existing graph model. To delete an edge, take two vertices as input from the user, update the adjacency matrix using that two vertices as index. If there is no existing edge, notify the user by saying "no edge available!"
3. **Show total number of edges:** Show total number of edges using the adjacency matrix.
4. **Show the matrix:** Show the updated adjacency matrix in the console.
5. **Exit**

Program 3 - 20 pts - Finding shortest path



Consider the above directed weighted graph. Assume the vertex D as the source. Use can see the pseudo code given in the lecture slide.

1. Show the shortest distance values from source to all other vertices in the given graph.
2. Show the shortest path from source D to destination G.

Submission Example

Extraction of LastnameFirstnameLab01.zip

```
/Documents
    LastnameFirstnameLab01.zip
    /LastnameFirstnameLab01
        /prog1
            prog1.cpp
        /prog2
            prog2.cpp
            A2Output.txt
       /Bonus
           bonus.cpp
```

Important reminder: Minimum penalty of plagiarism is failing (F) grade in the course.