The file city.txt(click the link to download the file) contains the weight between two directly connected cities:

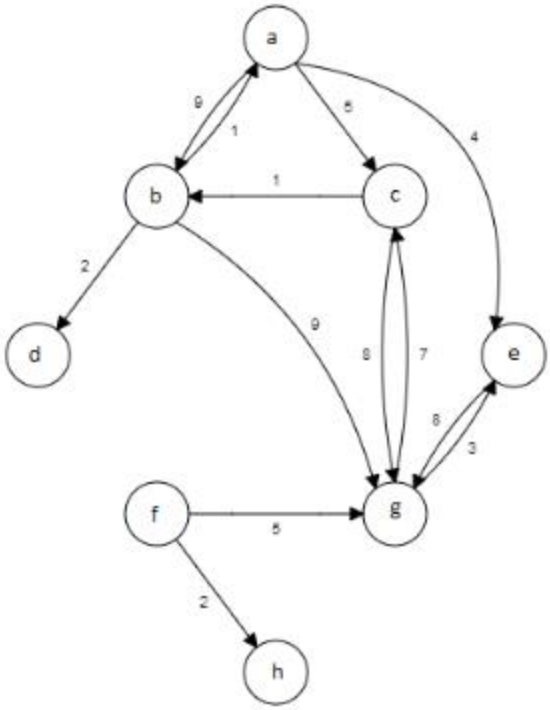
Or if you cannot download from the link above, the contents of city.txt file looks like the left hand side of the picture below:

a b 9...

a c 5...

For the figure below, this is a graph drawing of the city.txt file.

a	b	9
a	c	5
a	e	4
b	a	1
b	d	2
b	g	9
c	b	1
c	g	8
e	g	8
f	g	5
f	h	2
g	c	7
g	e	3



The individual letter is the names of a cities/nodes in the graph. The integer number following a pair of letters represents an undirected edge between two cities with a weight equal to the value of the integer. All the edges will be strictly positive integers greater than zero. The shortest path method takes a single letter to consider as the starting city and produces the following output:

The shortest path method takes a single letter to consider as the starting city and produces the following output:

a origin			b origin		
b	6	acb	a	1	ba
c	5	ac	c	6	bac
d	8	acbd	d	2	bd
e	4	ae	e	5	bae
g	12	aeg	g	9	bg

Each line will represent the letter of the city, the length of the shortest path from the origin to that destination city and the list of letters will be the shortest path from the origin to the listed city.

The output should format as this table. Each line will represent the letter of the possible destination city, the length of the shortest path from the origin to that destination city and the list of letters will be the shortest path.

The program would test the result with command line arguments.

Create a java class with your team name and implement the graph class:

```
class Group01
{
   public static void main(String [] args)
     {
         Group01 mygraph = new Group01(args[0]);
         mygraph.shortestPath(args[1].get(0));
     }
}
```

Consider the driver class would test each of origin by invoking the function mshortestPath(args[1].get(0)).

The main method will create a new graph from the file passed as arg[0].

```
 */
public static void main(String[ ] args) {
   //check for commandline arguments
```

The runtime arguments are stored as an array of strings in the "args" parameter, where the first parameter arg[0].

```
args[0] = "city.txt"; // the graph
args[1] = "a"; // the origin
```