CSC 226 - SPRING 2017
ALGORITHMS AND DATA STRUCTURES II
PROGRAMMING ASSIGNMENT 3
UNIVERSITY OF VICTORIA

1 Programming Assignment

The road network of NewVille has been recently upgraded, thanks to your help. Now the Firefighters can respond to emergencies more quickly than before. But the traffic condition is not always good and the fire vehicles must find the shortest paths to their destinations. Your job is to help them find the shortest paths to all the places of the town from their headquarter.

Input: An edge-weighted graph of n nodes (representing places in NewVille) and the position of the Firefighters' headquarter (the source vertex). The weight on edge (u, v) represents the time required to go from u to v or v to u at a certain time.

Output: The shortest paths to all the places (including the headquarter) from the headquarter. and the total time required to get there.

A Java template has been provided containing two empty functions. The function `ShortestPath` takes a two dimensional integer array G, that represents the graph, and an integer representing the source vertex, and calculates and stores the shortest path to all the vertices from the source vertex. The function `PrintPaths` takes the source vertex as an argument and prints out the paths in a specific format. Your task is to write both the functions. The input might be disconnected, but you can assume that the graphs are undirected.

You must use the provided Java template as the basis of your submission, and put your implementation inside the `ShortestPath` and `PrintPaths` functions in the template. You may not change the name, return type or parameters of those functions. The main function in the template contains code to help you test your implementation by reading it from a file or from the console. A sample file is also provided. You may modify the main function or any other function, because your submission will be tested using a different main function. You can use any helper methods or any helper classes. You can use any built-in class or write your own classes and data structures. We advise you to put all the classes you write in the same file, but no other class except the provided one should be declared as a public class.

2 Examples

The input format is exactly like Assignment 2. The first line denotes the number of vertices n in the graph and the following n lines represent the $n \times n$ adjacency matrix. A 0 at row i and column j in the adjacency matrix means that there is no edge between vertices i and j, otherwise that number denotes the edge weight between i and j. You can assume that the edge weights are between 1 and 1000.

Sample input:

```
 3
 0   5   6
 5   0   7
 6   7   0
 6
 0   7   9   0  0  14
 7   0  10  15  0   0
 9  10   0  11  0   2
 0  15  11   0  6   0
 0   0   0   6  0   9
14   0   2   0  9   0
```

Sample output:

```
Reading graph 1
The path from 0 to 0 is: 0 and the total distance is : 0
The path from 0 to 1 is: 0 --> 1 and the total distance is : 5
The path from 0 to 2 is: 0 --> 2 and the total distance is : 6
Reading graph 2
The path from 0 to 0 is: 0 and the total distance is : 0
The path from 0 to 1 is: 0 --> 1 and the total distance is : 7
The path from 0 to 2 is: 0 --> 2 and the total distance is : 9
The path from 0 to 3 is: 0 --> 2 --> 3 and the total distance is : 20
The path from 0 to 4 is: 0 --> 2 --> 5 --> 4 and the total distance is : 20
The path from 0 to 5 is: 0 --> 2 --> 5 and the total distance is : 11
Processed 2 graphs.
Average Time (seconds): 0.00
```

3 Evaluation Criteria

The programming assignment will be marked out of 40, based on a combination of automated testing (using large test arrays) and human inspection.

You are advised to implement Dijkstra's algorithm. The running time of your code should be at most $O(V^2)$. The mark for each submission will be based on both the asymptotic worst case running time and the ability of the algorithm to handle inputs of different sizes.

Score	**Description**
0 - 15	Submission does not compile or does not conform to the provided template.
15 - 30	The implemented algorithm is substantially inaccurate on the tested inputs.
30 - 40	The implemented algorithm is $O(V^2)$ and gives the correct answer on all tested inputs.

To be properly tested, every submission must compile correctly as submitted, and must be based on the provided template. If your submission does not compile for any reason (even trivial mistakes like typos), or was not based on the template, it will receive at most 15 out of 40. The best way to make sure your submission is correct is to download it from conneX after submitting and test it. You are not permitted to revise your submission after the due date, and late submissions will not be accepted, so you should ensure that you have submitted the correct version of your code before the due date. conneX will allow you to change your submission before the due date if you notice a mistake. After submitting your assignment, conneX will automatically send you a confirmation email. If you do not receive such an email, your submission was not received. If you have problems with the submission process, send an email to the instructor before the due date.