CSC 226 - SPRING 2017
ALGORITHMS AND DATA STRUCTURES II
PROGRAMMING ASSIGNMENT 2
UNIVERSITY OF VICTORIA

1 Programming Assignment

The town of NewVille was a very small town five years ago. There were very few roads that were good enough for cars, but it wasn't a problem because there were very few cars in the town. Now the population has increased four times and there are so many cars that the mayor has to undertake a huge road construction project. His plan is to build on the existing road network, but to minimize cost he is thinking of improving only some of the roads. Of course he has to make sure that you can go from anywhere in the town to anywhere else by car. Given the road network, your job is to calculate the minimum budget required for this construction project.

Input:	An edge-weighted graph of n nodes (representing places in NewVille). The weights on the edges represent the cost of rebuilding that road.
Output:	The minimum cost/budget required for the construction project.

A Java template has been provided containing an empty function `mwst`, which takes an two dimensional integer array G, that represents the graph, and returns the sum of the weights of the edges in a minimum spanning tree integer of G. Your task is to write the body of the `mwst` function. Your code is not required to check for incorrectly formed input data.

You must use the provided Java template as the basis of your submission, and put your implementation inside the `mwst` function in the template. You may not change the name, return type or parameters of the `mwst` function. The main function in the template contains code to help you test your implementation by reading it from a file. 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. Only the contents of the `mwst` function and associated helper functions (if any) will be marked.

2 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.

There are several algorithms for MWST. You can implement Prim's algorithm (lazy or eager) or Kruskal's algorithm. If implemented correctly, the running time of your code should be $O(E \log V)$. 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. The table below shows the expectations associated with different scores.

Score	Description
0 - 15	Submission does not compile or does not conform to the provided template.
15 - 30	The implemented algorithm is not $O(E \log V)$ or is substantially inaccurate on the tested inputs.
30 - 40	The implemented algorithm is $O(E \log V)$ 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.