Hana Ćatić

Assignment 2

For this assignment, our job was to solve tasks about glass spheres - a new invention used as a mean of public transport, but in reality a representation of a graph. We were presented with 15 locations and a few files - each depicting a new graph.

The first task was the easiest. We had to figure out how to represent a graph. I created a class Graph consisting of a map, with String as its key (the String representation of a node), and Node as its value - a class created for representing the nodes in a graph. The class Node consists of its name and a map of its neighbors and distances from them. The class Node has a method for inserting new neighbors into the map, while the class Graph has methods for inserting nodes and edges into the graph itself.

The second task was implemeting a class for Dijkstra's algorithm. In my opinion, that was the hardest task. It required a lot of searching the internet for different ways of implementing the algorithm. I opted for the method of using a Priority Queue which orders its parts by distance between nodes. I also used a map which stored representations of the nodes and their distances from the starting node. I iterated through every node and its neighbors while the queue wasn't empty and updated the distances. If there is no connection between nodes, the function returns -1.

The third task could be interpreted as an extension of the second task, because it required us to find the optimal paths between each and every node in the graph. The task was not hard to solve. It required iterating through all the nodes in a graph and using Dijkstra's algorithm to find the distance to all the other nodes. However, some files contained codes for locations that don't exist in the places.txt file (like Q or P). I handled those instances just by writing the shortcode onto my files instead of throwing errors.

The fourth task was pretty easy. It incorporated probability. We had to adjust our program so that it takes into consideration the probability of some constrains occuring between specified nodes. I used a random variable which generates a random Double number between 0 and 1 every time the program is run. If the number is less or equal than the number specified in the constraints.txt file, the constraint is applied and the path is blocked. If not, everything works as normal.

This assignment was an interesting one, but challenging as well. It took a lot of patience (and I'm afraid I've run out for now) and research. I think it was smart to incorporate DSA and probability even though the DSA is what made the assignment so challenging.