ECE368
Project 3 Milestone 1
Chul Woo Park

In this project, I am going to implement map routing according to Dijkstra algorithm. Dijkstra algorithm is a well-known algorithm to find the shortest path between two specific nodes in a graph. Firstly, I am going to use a 2-D array for representing the graph and use that as a map. By using the 2-D array, I would have an advantage that indices can be exactly used as vertices (x-coordinate and y-coordinate). However, connection of each node might be hard to implement in 2-D array so that my back-up solution would be using modified linked list. At the beginning of my program, it will initially read the input file and start the process of parsing. For the first line, it will read first value and store it as the number of nodes and second value as the number of edges. Then, the next information is the nodes and respective vertices. Since my program knows the number of nodes, it will read that much of lines from the input file. While reading each line, it will add the node into the 2-D array in respective position (x-coordinate, y-coordinate). Then, the next information is the edges that which node and which node are connected to each other so that path can be made. Again, since my program knows the number of edges, it will also read that much of lines from the input file. In order to include the edge information, each node will have a set, which includes connected nodes, as an attribute. For example, if line says 1 4, it will include node 4 into a set of node 1 and node 1 into a set of node 4. Therefore, my program knows which nodes are connected to which nodes. At this point, all the information about the graph will be in the 2-D array and attributes of each element. My program will read the query in order to know which shortest path to find (between which node and which node). It will read the query input file similar to parsing algorithm I just mentioned above. When my program reads a line from query input, it will know the initial node and the target node that the path should arrive from the initial node. Then, the actual Dijkstra algorithm takes in place. It will start from the initial node and proceed to all possible connected nodes while calculating the distance. The distance between two nodes will be determined by computing the Euclidean distance. Each node will have an attribute of distance from the initial node. Since my program is computing for the shortest distance, distance from the initial node will be initialized as large as possible so that no computed distance can overpass that initialized value. Then, my program is going to use recursion in order to go through all possible paths from initial node to target node. Important feature that needs to be added is a visited flag because my program should know whether such node is visited before or not so that it can choose a new path on the further iterations. Such visited flag will be declared as boolean (true or false). When it reaches the target node, it will compare the distance of the path with the former shortest distance. If it is greater, it will proceed to look for another path and if it is less, former shortest distance will be replaced by it. In this way, my program will finally store the shortest path among all possible paths. This is a brief idea of my algorithm and it could be modified in a more efficient way and taken care of some edge cases.