ECE 368
Project3
Milestone1
By- Kshitij Jain

This project is based upon graphs, traversing them and to be more particular upon Dijkstra's algorithm. The project's most important part is essentially using Dijkstra's algorithm to traverse a graph and find the shortest path between two given nodes at any point of time. Therefore, I will begin this project by trying to completely and fully understand Dijkstra's algorithm and any of its variations that might be required. Once I understand Dijkstra's algorithm fully, I will then try writing its pseudo code based on the parameters given to us in the project. Once I am sure that I have got the algorithm and the pseudo code right, I will then start working on the other parts of the project.

As far as parsing and reading the input file go, I think that the best way to parse the given input file and store all the relevant information for easy and practical access will be to create a multilinked linked list. In this linked list each node will represent a vertex in the given graph. Each node of this linked list will store the x and y co-ordinates of the corresponding vertex and will contain pointers to all the neighbors of the corresponding vertex. I believe that this infrastructure that I will create will easily integrate with the Dijkstra's algorithm that I will be using.

Now, to talk a little bit about Dijkstra's algorithm and describe it, I'll start by saying that this algorithm seems relatively easy to implement. In this algorithm we start by taking a vertex and assign weights to the distances between the current vertex and all the other vertices, which is infinity when we start. Now, we proceed further by examining the neighbors of the current node and find their distances from the current node. We then move to the one with the shortest distance from the current node and make this node the current node. We will repeat this process until either we finally reach the node that we wanted to or the distance from the current node to our destination node is infinity, in which case we go back to our starting node, choose an alternative neighbor and repeat the whole process.