Name: Andrew Gan
ECE 368 Project 3 Milestone 1
Date: 11/12/2019

The solution to the project can be implemented in 3 phases.

The first step is to parse the input. The first line of the input file is the number of vertices and edges. The next few lines hold the absolute positions of each vertex, followed by the next lines containing information about the edges that connect the different vertices in the map. The information may be stored in a struct type object that contains the following fields: two ints representing number of vertices and edges, an array of structs consisting of two ints to hold the coordinates of each vertex, and an array of structs consisting of two ints representing the connections made by each edge. After this is done, a map structure may be created.

The second step is using the information retrieved from the input file to create the map structure. The data structure will consist of nodes with multiple parents and children that are linked together. A node contains information such as node ID, node distance, visited state, and an array of pointers to its children. The starting initial node has distance of zero, while all other nodes will have distance of infinity. As the pointer traverses through the node, the distance values of the nodes are updated. Two sets are created, one for unvisited nodes which will be initialized to contain all nodes, and the other for visited nodes, which will be an empty set.

```
Typedef struct {
   int x, y;
} Vector2;
```

```
Typedef struct {
   int numVert, numEdge;
   Vector2* vertices;
   Vector2* edges ;
} MapInfo;
```

```
Typedef struct _Node{
   int id, distance;
   bool visited;
   _Node** child;
} Node;
```

After the creation of the map structure, the final step is to apply Dijkstra's algorithm. As the program traverses through the nodes, all unvisited children nodes are considered and the one with the shortest relative distance to the current node is selected. If the selected child node has a distance value larger than the sum of current node's distance and the edge distance, the value is overwritten. Else, the smaller value is kept. After all unvisited children nodes are considered, the current node is placed into the visited nodes set. This process repeats until the destination node is marked visited, which indicates that the shortest path has been found, or if the shortest path out of all the unvisited nodes is infinity, which indicates that no path to the end node was found.

Green boxes represent a Node struct type.
Blue lines represent the pointers stored in Node** child.