```
/*****************************************************************
 * Project Report Template
 * Project 3 (Map Routing), ECE368
 *****************************************************************/

Name: Sthitapragyan Parida
Login: sparida

/*****************************************************************
 *  Explain your overall approach to the problem and a short
 *  general summary of your solution and code.
 *****************************************************************/
To implement Djisktra's algorithm I focused on first correctly representing the graph in memory. Out of
the various ways to represent a graph, I agreed on the adjacency list represntation as it allowed
modifications to the algorithm itself and was also memory efficient. After that, the algorithm first
recreated the graph from the graph file using linked lists. The edge list of each vertex was a priority
queue which stored the list in increasing order of vertex number. Then for each query a min heap is
created of all the vertices, with the length of the route from the source to each vertex is stored.
Initially source is assigned distnace 0. Then the standard Djisktra's algorithm was implemented using the
min heap. For each query the path was traced using an array and the reversed to print the orginal path.

Sourcefile: shortestpath.c
Compile: gcc -o shortestpath shortestpath.c -lm
Usage: ./shortestpath <graph file> <query file>

/*****************************************************************
 *  Known bugs / limitations of your program / assumptions made.
 *****************************************************************/
Known Bugs:
Incorrect results for certain queries
Limitations:
Max vertex number = 100000
Max value of x and y coordinates = 10000
Does not consider cost of travelling from vertex to destination when calculating best path.
Assumptions Made:
All cooridnates are integers
All vertices are whole numbers and occur continuously with no vertex number missing
It is assumed that the input files are error free and contain values which match the project
requirements, as error checking is not implemented in the code.

/*****************************************************************
 *  List whatever help (if any) that you received.
 *****************************************************************/
I had to read about min heap implementation of Djisktra'a algorithm online and understand the idea of
using a refrence of a min heap node rather than anew node itself to contrcut the min heap. This triggered
the need of a position array to indicate where exactly in the heap was a specific vertex.

/*****************************************************************
 *  Describe any serious problems you encountered.
 *****************************************************************/
After the initial implementation of the program using arrays, the program was extremely slow. Further
research showed that the running time was on the scale of O(V^2) where V is the number of vertices. Then
I read about a min heap approach of finding the shortest edge out of the unvisited vertices. As
instructed in class the min heap were only pointers to the vertices rather than new copies. This demanded
the implementation of a position array to actually store the position of each vertex in the min heap.
Therefore the correct index of a vertex in the min heap was stored in the position array at the location
position[vertex]. After successful implementation the runtime improved significantly and further readig
showe dthat the run time had reduced to O(VlogE), where E is the number of edges.


/*****************************************************************
 *  List any other comments/feedback here (e.g., whether you
 *  enjoyed doing the exercise, it was too easy/tough, etc.).
 *****************************************************************/
I liked completing this exercise and this sparked my interest in pathfinding in general and currently I
am trying to modify my approach to implement the A* algorithm for pathfinding which takes the cost of
travel from a vertex to destination into account when calculating the best possible path.
```