```
/*****************************************************************
 * Project Report Template
 * Project 3 (Map Routing), ECE368
 *****************************************************************/
Name:  Max Chi
Login: chi19
/*****************************************************************
 *  Explain your overall approach to the problem and a short
 *  general summary of your solution and code.
 *****************************************************************/
My approach to this assignment was start by first inputting the map file and
making an adjacency matrix to keep track of the connections between different
vertices by storing their x and y coordinates in an array of pointers. I then
read in the query text file, which would provide the number of paths that
wanted to be determined, as well as its initial and final vertex. I set all
the values to infinity (INT_MAX) and set the source's distances to itself as
0 since it's in the exact same spot as itself. The algorithm would then
continually search for the shortest distances to its neighbors, then search
for the node with the smallest distances that was unvisited, calculate the
distance between it and all of it's edges, and then set the vertex to
visited, continuing this processes until the destination had been found.
After finding the shortest path, the program would backtrack by inserting the
previous nodes before it into path array, that would keep track of which
certain path that was taken. The destination would contain the total distance
travelled and would print the path in reverse order starting from the final
vertex to the initial vertex.
/*****************************************************************
 *  Known bugs / limitations of your program / assumptions made.
 *****************************************************************/
Known Bugs: There seems to be a problem with my output is that the shortest
distance is sometimes slightly off for input (maps) that are very large (>
100000 nodes) but would still have the correct path.
Limitations: I used the method of incorporating an adjacency matrix to keep
track of the neighboring nodes as well as their locations. I would
dynamically allocate space for
Assumptions made: 1) any distance that exceeded the maximum limit for an
integer (INT_MAX) would be considered infinity. 2) each vertex would have at
least 2 edges, and every edge would have 2 vertices to connect. 3) If any
nodes share the same coordinates, they would have a distance of 0.
/*****************************************************************
 *  List whatever help (if any) that you received.
 *****************************************************************/
I looked online for explanations of Dijkstra's algorithm and asked some other
students that are familiar with the algorithm to explain it to me. Also
referred to lecture slides for help as well.
/*****************************************************************
 *  Describe any serious problems you encountered.
 *****************************************************************/
Not many problems, mostly just the level of difficulty it required to think
about how to implement the algorithm for larger inputs (over 100000 nodes)
/*****************************************************************
 *  List any other comments/feedback here (e.g., whether you
 *  enjoyed doing the exercise, it was too easy/tough, etc.).
 *****************************************************************/
Relatively tough but fun assignment. Fascinated by the logic behind the steps
of finding shortest path. Would not have thought about implementing the same
way as Dijkstra's logic.
```

Compiled with –lm flag. “gcc –Wall –Werror proj3.c –o proj3 –lm”