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

Name: Mitch Bouma
Login: mbouma
Compile line: gcc -Werror -Wall -O3 -lm shortestpath.c -o shortestpath

```
/**********************************************************************************************************
 *  Explain your overall approach to the problem and a short
 *  general summary of your solution and code.
 **********************************************************************************************************/
```

I first made a matrix that was numVerts down and 2 wide to hold the x,y coordinates of each vertex. Then I made a matrix numVerts down and numVerts wide to hold the distances of neighboring points. If there was a distance value in any column of a specific row, that meant the column and row indexes were in fact neighbors, and the distance between them was the value at that index. In the Dijkstra function, I first made visited and distances arrays to keep track of whether I have visited each vertex and the total distance to each vertex. Then I looped until the end vertex had been visited, finding the min distance in the distances formula, and setting that vertex to visited. Inside that loop, I looped again, but this time from 0 to numVerts, checking if the index was 1.) unvisited, 2.) not 0, and 3.) less than the total distance for that node if added to the distance at the current min. If these conditions were satisfied, I updated the new total distance in the distances array to include this addition and updated the index of the previous array in order to have a path from the end to the start. To print, I followed the previous array back, starting at the end vertex, and following the values that were in each index until arriving at the start.

```
/**********************************************************************************************************
 *  Known bugs / limitations of your program / assumptions made.
 **********************************************************************************************************/
```

There is one main limitation of my program as it stands now. For example, because I chose to make the connections matrix numVerts x numVerts, I had to search through numVerts for each row in order to find all neighbors. I could have chosen to use a linked list for each vertex, therefore allowing variable widths and only having to search through as many nodes as there were. Because of this, paths that take large numbered vertices make my program run somewhat slowly. When finding a path from 0 to 82,000, the program takes close to a minute.

```
/**********************************************************************************************************
 *  List whatever help (if any) that you received.
 **********************************************************************************************************/
```

I was able to utilize multiple Youtube videos explaining the process of Dijkstra's algorithm. In addition, I modeled my algorithm off Wikipedia's pseudocode.

```
/**********************************************************************************************************
 *  Describe any serious problems you encountered.
 **********************************************************************************************************/
```

I ran into 2 major problems. The first was that I was originally calculating the final total distances for each vertex, even if they were higher than the end point. I realized that once I visit my end vertex, I no longer had to keep running the algorithm. Adding this "quick stop" cut my program's run time in half for paths that don't take very high numbered vertices.
The second problem was that my program was often times finding a correct path through the map, but it wasn't necessarily that shortest one. I spent two days debugging it and finally realized that it was becuase some vertices are a distance of 0 away from each other (which doesn't really make sense). I had been using calloc to initialize my matrices which automatically sets the indices to 0. Since one of my conditions in Dijkstra was to check if the value was not 0, it was skipping those neighbors. I fixed it by setting any distance of 0 to -1 instead and treating the -1 values as 0 when adding to the total distances.

```
/**********************************************************************************************************
 *  List any other comments/feedback here (e.g., whether you
 *  enjoyed doing the exercise, it was too easy/tough, etc.).
 **********************************************************************************************************/
```

The project was quite enjoyable for me. It was definitely easier than the second but harder than the first. It helped to have an abundance of online walkthroughs and information about the topic as well. I recommend keeping it in the curriculum.