In assignment 6, our last assignment we were asked to implement a graph for the sake of using Dijkstra's algorithm to find the shortest path. I had said that assignment 5 was the most enjoyable assignment but I change my mind because this assignment was very cool. All of the functions were easy to implement except for the actual shortest path algorithm. I think that this assignment has some real-life value to it because we could technically use it to actually calculate the shortest paths from one node to another.

In terms of implementation, for this assignment, I was a little confused at first about how exactly to use an adjacency matrix when my nodes and edges are independent objects, not just numbers. I tried using a hashMap to map the vertices to a respective integer but didn't really know what to do with that after. One solution I came up with was to store the actual Road object in the adjacency matrix but then I feel that would defeat the point of using it. Ultimately I scrapped the whole idea of using a matrix and used 2 sets, one of the vertices, and the other of edges.

Other than selecting the data structure to use, I only struggled with one other thing, and that is Dijkstra's shortest path algorithm. Going into this assignment I didn't really even know how it was supposed to be implemented so I had to do a lot of homework to catch up. After watching some youtube videos and reading the slides I figured out the general idea. The coding part came with some issues too, I kept accidentally creating new town objects within my functions and that caused the whole program to break. So my solution was to go back and make sure that all the town objects I used were not deep copies.

One thing I think I did well in this assignment is implementing the basic methods (all except for Dijkstra's and shortest path ArrayList) I managed to pass all of the tests after not

much debugging. I think that it was important that I struggled with functions I did struggle on because now I have a much better understanding of them.