Final Summary

In this project, we had a lot of big setbacks. There were two times where we had put in a good amount of work into our new approach, only to find out that there was something fundamentally wrong with the design, and we would be back to square one.

At first, we wanted to create an intersection class that would be composed of three street objects, and we would reference and give each street an object of this class so that we could write functions in street to move Car object from one street to another. However, we soon realized that the Street and Intersection classes were both composed of each other, meaning that an object of either class would require infinite memory. As a result, we proposed a new design.

The next approach we had involved breaking the Streets down into two child classes, ExitStreets and DestinationStreets. It seemed that we were making progress. Having all the classes referencing each other so they could push and pop from each others queues. However, we soon learned that many of the errors we were encountering were due to circular logic. It never occurred to us that we couldn’t have two separate classes referencing each other. So we have to revise the relationships between all the classes with queues.

In order to avoid the previous problems we were experiencing, we decided to do a trickle down approach, where we would start with the building class, and work our way outward to the ExitStreets. This model would allow for the classes to transfer Cars to their appropriate queues without needing two classes to have access to each other. The DestinationStreets would be able to push and pull from the ExitStreets and the Buildings would be able to push and pull from the DestinationStreets.

In the end, our program compiled and would run partially, however, it would crash due to a divide by zero error in our show\_stats() function in Simulation.