Devin Sevy
Refactoring

With the simulator assignment, I began by writing the code all in one file. I was continually bouncing around trying to keep my place. I had a giant file with no baring, and was hoping that it all ran accordingly. If I hit a snag, debugging the program became very taxing and I lost track of where I was. Once the program was running correctly, I forgot to enter check points to get statistical data for the program. Since the file was so large, it took awhile to find where to put those points. When I heard our next assignment was to refactor our program, I was excited to take on the task.

First, I separated the two simulations from one program to two. By doing this, I was able to see clearly how each program reacted differently. The program was easier to read, I was able to debug easier, and functions were much closer to the main so I could see how and when the function ran.

Second, I updated the loops in the supermarket simulator. The loops I originally used took awhile for me to figure out. The loops were dealing with objects, checking values, adding, and deleting within the same loop. After everything ran, I had more loops adding, erasing, and changing values of queues and vectors. I looked online and found ways to execute loops better. What I found were iterators and the option fit my program perfectly. I refactored my code to work with the iterators, and it decrease my code from about 20 lines to 4. This was a major refactor, and one I am glad I did.

Finally, the last change I refactored was changing the program to work with classes instead of structs. I wanted to originally run my program with classes, but I started coding and reached a point of no return. I am glad I did, because it cleaned the code up immensely. I was able to track things on a class level instead of relying on one variable for all the data. The code flowed better and I was able to see all the moving components. Debugging was much easier. When I stepped into a class the other variables went away and I was able to focus on the data within the class. Overall both programs worked more efficiently and I was able track of data much better.

I am glad I chose to refactor my program. Refactoring made the program easier to follow and improved the execution. I always thought refactoring was small changes throughout the program that never impacted the results, but this major refactor helped the program perform significantly better. I learned some new things with c++, and will continue to refactor in the future.