This report is included with the README and the Stop Light project files. The Stop Light project folder contains main.cpp which contains all the functions and the driver for the project, there are also three other files. The three other files are easy.txt, medium.txt and hard.txt, these contain the sample inputs from canvas. This allows users to test the project and see how the inputs should be formatted. If the input needed to be changed any of these files can be replaced with a file of the same name and file extension.

The goal of the project is to create a multithreaded intersection that reads in a file with key value pairs (Arrival Time, Direction) and creates a car object. The car object will be used to keep the key value pairs together and also to create threads for the lanes and sorting of the cars. A thread should then be created with the car and that thread should check if it’s able to go through the intersection safely. If the car is safe to go through the intersection without conflicts, then it will release the lock and be popped off the queue. If the thread is not able to go through the intersection safely then the thread will wait until the lock is release and try again to see if it’s safe. we need to output when the thread is created and also when the cars have gone through the light and what the thread number was.

This program should run until there is no more cars in the queue and there shouldn’t be any accidents with the cars. We are working with threads to make sure that we understand how they work and how to work with them.

The program currently runs for roughly 43 seconds. The program reads in the file and creates the car objects, after creating the car objects, it creates threads and sorts the cars into lane priority queues. Then it checks if the car is safe to go, then pops the car off the queue and joins the thread. The run time should be around 40 to 46 seconds, because each car should take about a second to arrive and pass through the light plus the time it takes to read the file, store and sort the priority queue. When threads are created “Thread Created: #” is printed, when the thread passes through the intersection “Direction: N ID: # Time: #” is printed and when the thread is joined, then, “Thread joined: #” is printed.