By: Dhiraj Amarnani, Hassan Naveed, Andre Koppert

Class Design Details:

Section
Sections are the segments that form the lanes and are the superclass of intersections. The section is a class that holds a Boolean called occupied. If occupied is true, a part of the vehicle is located at this section. It also contains a variable called “occupiedBy”, which helps in printing out the simulation. We create sections in each lane.

Lane
The lane class creates the lanes of the intersection as an array of sections. The class also holds the direction and the size of the lane. Additionally, the two intersection sections which are in the middle of each lane are given to this class. We also pass a Traffic Signal to each lane. In main, 4 lanes are created, one in each direction.

Intersection
Intersection is a subclass of section which is located where the 4 lanes meet. An intersection holds the integer timeInUse which specifies how long this intersection will be occupied by a vehicle that is currently passing the intersection and potentially changing direction. 4 instances of the Intersection class will be created for each intersection block.

TrafficSignal
TrafficSignal holds the current state of the traffic signal which is either green, yellow or red. The duration for each state is specified in number of clock ticks. The class also contains the integer timeUntilNext, which tells us when the next change of light occurs. There will be 2 instances of the TrafficSignal, one for the East & West bound lanes and one for the North & South bound lanes.

Vehicle
The vehicle class is used to create the cars, SUVs and trucks. The size parameter in vehicle specifies which of the three a particular vehicle is. The vehicle class also holds parameters that specify which lane a vehicle is coming from and which lane it is going to. Additionally, the relative turn direction is stored for each vehicle instance. It holds an array of sections (specifying what sections in the lane it is currently on). It uses the canProceed and proceed methods to move in its lane or change lanes whenever appropriate.

Testing Correctness Details:
Much of the testing of correctness in our program involved outputting the results or intermediary states of the variables we transformed during the simulation. For many of the classes, our method of testing involved creating the respective objects and printing out the states in the GUI

we developed contained in our main.cpp file. This GUI prints a representation of the lanes at every timestep, indicating the color of the North-South and East-West traffic signal lights as well as their remaining time left and the number of vehicles being generated in our file. Testing the traffic signal, vehicle, lane and section/intersection classes were done by the manner in which the GUI followed the instructions given as our project. Additionally, in developing each of our classes, we also repeatedly printed out various class fields to ensure they were initialized and were changing appropriately as our simulation progressed. For example, we printed out the state of the timeInUse variable for each section to ensure the vehicles entering the intersection were reading the correct intersections and were proceeding with the correct conditions.

Compile & Running Details (with examples):

To compile, use the Makefile. It should compile by typing “make” in the terminal. This will create an executable called run. This can be run simply by using “./run”. To clean the files, use “make clean”. The main method in the program holds everything together. It reads input from a file called input.txt . The numbers are in the following order:

Left Probability
Right Probability
Green Ticks
Yellow Ticks
Red Ticks
Vehicle Appear Probability
Car Probability
SUv Probability

The numbers in the files can be changed to modify the parameters, such as probability of creating cars, the direction they will be given, their size, timing of signals, etc. It then goes on by creating the two traffic lights, the lanes, an empty vector of pointers to vehicles, and has variables for the clock ticks and number of cars produced. Then it has a loop which does most of the work by iterating clock ticks. With each clock tick (or iteration of the loop), vehicles are generated with their given probabilities, the traffic signal are updated and then all the vehicles proceed in the order they were created. At the end of the main, all these vehicles are deleted. Each clock tick is by default one second, but it can be changed.