**HW4 and report**

Int main(){

Initialize variables, pthread conditionals. Print the menu to the screen with a switch statement allowing the user to select the options.

}

Before touching on options 1-6, ill describe the main vehicle\_routine() function:

Vehicle\_routine(){

Vehicle\_arrival() is called for the respective vehicle object. The first condition checks to see if it’s a car. Mutex lock is enabled, and the conditions for which the car cant move are in the while loop. The cond\_waits are called within if-else statements, checking the directions. Outside of the while loop signifies the cond\_wait is fulfilled (signaled, more on that later). The car is removed from the waiting list, moving car is incremented, and the direction of travel is saved for later use. The car is added to the moving list and the print functions are called to display useful info. The mutex is briefly unlocked to simulate the bridge travel with sleep() only to then be locked again. Now were gonna simulate the car leaving, decrementing the movingcar variable and deleting it from the moving list. Now we call our pthread\_cond\_signals, conditionalizing it for when there are waiting trucks. If there are, we signal the trucks to move. If not, we signal the cars to move. We then unlock the mutex and free up the space. Now we move to the else if statement for when the vehicle passed through is a truck. This is pretty much the same as the previous if statement for cars, with a couple key differences. The conditions for the while loop only include when cars are moving or if a truck is moving, meaning it can only move if the bridge is empty. To alternate the directions north and south for each next truck, I conditionalized the current moving direction and switched it wehenevrr it is called, ensuring the next truck will head in the opposite direction. The rest of the logic is the same for the car if statement, just renamed for trucks.

}

Options 1-6:

These all use for loops to instantiate each vehicle of index j with that respective vehicle id. On each iteration, the vehicle\_type is randomly assigned based on the probability already given to us. The attributes are added to the object (struct). Once the vehicle is created, the pthread is created for each vehicle, calling the vehicle\_routine for each threa. The mutex is then unlocked, and the pthreads are joined. Each of these options are implemented almost identically, with small changes made to meet the testing requirements for each option (sleep if a delay is required, and more for loops to instantiate more objects if requested)

Unfortunately, this code runs differently each time even though nothing changed. Sometimes I will get segmentation faults, sometimes not. The biggest issue is mixed traffic. The trucks will pass through one at a time, alternating as intended, but once its all cars, the program freezes.