OS Assignment 4 – Document

Name: Sajeel Nadeem Alam

ID: sa06840

In order to ensure that all the given conditions i.e no more than 3 cars should be present on the street, either only incoming or only outgoing cars should be travelling at the same time and after every 7th car leaves, the street becomes unusable and has to be repaired, I have used mutexes and condition variables.

No more than 3 cars should be present on the street at any instance:
In order to fulfil this requirement, I have used conditional statements in my code in the street_thread function. For the situations where I am allowing a car to enter the street, I am first checking with the help of conditional statements whether the total number of cars present currently on the street are lesser than 3 or not. If they are then I am letting the code flow to the part where it lets the car in question enter the street with the help of pthread_mutex_lock and pthread_mutex_cond functions. However, if the number of cars on the street are equal to or greater than 3, then I am not letting the car enter the street in that iteration of the while loop and make it wait for the next iteration. If a car enters the street, the total number of cars on the street is incremented in these conditional statements for both the cases of incoming and outgoing depending on the car's direction. Similarly the total number of cars is decremented in the functions incoming_leave (for when an incoming car leaves) and outgoing_leave (for when an outgoing car leaves) and this is done with the help of pthread_mutex_lock and pthread_mutex_unlock.

Only incoming or only outgoing cars travelling at a time:
In order to ensure this I have made sure that an incoming car does not enter the street if an outgoing car is travelling on it and vice versa. Hence before letting an incoming car enter I first check whether there are incoming cars waiting to join the street (in addition to less than 3 cars). If this is true then I check whether the direction of the street is True (which refers to incoming). If this is the case then I let the incoming car enter the street. However, if the direction is not True but the number of cars on the street is zero, I update the direction to True and let the incoming car enter the street. Similarly, prior to letting an outgoing car enter I first check whether there are outgoing cars waiting to join the street (in addition to less than 3

cars). If this is true then I check whether the direction of the street is False (which refers to outgoing). If this is the case then I let the outgoing car enter the street. However, if the direction is not False but the number of cars on the street is zero, I update the direction to False and let the outgoing car enter the street. In both cases, once the car has entered the street, I send a signal with the help of pthread_cond_signal(). Hence a car can only enter the street if the direction of the street matches its direction or if the number of cars on the street is zero which ensures that only incoming or only outgoing cars travel at the same time.

After every 7th car leaves, the street becomes unusable and has to be repaired:
In order to implement this, before checking any other condition I am checking in street_thread inside the while loop if the number of cars since last repair are lesser than 7 or not. If they are then I am letting the code flow as normal however, if they are not then I am additionally checking whether the number of cars on the street are zero and if they are then I am calling repair street and updating cars since repair to zero. Additionally, every time a car enters the street, I am incrementing cars since repair inside mutex locks. Since this is running in a while loop, at every iteration my code checks whether the street needs to be repaired or not.

No deadlock:
In this scenario deadlocks may happen in two cases. Firstly, if multiple cars (threads) are waiting for an event that will never take place. This case is fixed with the help of conditional variables that sent signals to that waiting thread. For example, if a car X is entering the street, no other car can enter at that same time hence they are waiting. When car X has entered the street, with the help of conditional variables such as pthread_cond_signal, we will send a signal to the waiting cars (threads) that they are free to enter the street now. Another case of deadlocks may be if multiple threads are changing certain variables at the same time. In order to prevent this I have made sure that these variables are updated within locks so that only one thread can update them at a time.

Maximum Concurrency:
With the help of conditional statements and mutexes, I have ensured that at all times maximum concurrency is ensured i.e if a car is waiting to enter the street, it can even if there are other cars on the street given that they are travelling in the same direction as the waiting car. At all times, maximum three cars travelling in the same direction can travel on the street

at the same time. Moreover, once the street is empty another car enters in instantly ensuring that no time is wasted. These help to ensure that maximum concurrency is achieved in every possible scenario.