Assignment 4 – Operating Systems – fa06666

We know that when there are more than one processes running at the same time different threads can share access to the same data, in our case static global variables, that are constantly changing throughout the program. We need to synchronize the different threads so that if a value is changed in one thread, it reflects accurately when we access the same value through another thread. We can do this by using Mutex Locks to protect the shared resources which ensures that at a time, only one process has access to the data at a time. I have done this in our code by creating one Mutex Lock “mutexlock” in line 23 and then locked the thread every time a value is being changed. I have used a single lock in the entire code so that I can avoid deadlock. This is being done every time a thread accesses any data. I have used condition variables long with these mutex locks since we can use these variables to allow the threads to wait until the condition becomes true. These condition variables wait inside the mutex locks until their condition is fulfilled. The wait function blocks the thread, releases the lock associated with the condition variables and once the thread the unlocked it reacquires the lock. The signal unblocks the thread. In our assignment, we are required to ensure that not more than 3 cars are present on the road at one time. I have ensured this by incrementing cars\_on\_street every time a car enters the street. If there are more than the ALLOWED\_CARS then the thread is blocked and it waits until the incoming\_leave or outcoming\_leave function sends it a signal. Once there are less than three cars on the street it will continue to let more cars enter the street. We are required to ensure that incoming and outgoing cars do not enter the street at the same time. I have done this by giving a condition where if an outgoing car is on the street an incoming car will not enter and if there is an incoming car on the street then an outgoing car will not enter. To ensure that there aren’t only outgoing cars going constantly or ingoing cars going constantly I have tried to create a signal which changes the direction of incoming and outgoing cars so it does not cause deadlock in any specific direction. After every 7th car leaves the road will need to be repaired. I have done this by creating another condition variables which signals every time the cars on the street and the cars that have left the street add up to the USAGE\_LIMIT. Once this condition becomes true the code signals the wait in street thread function. It will then wait for the cars on the street to exit. I have placed a flag to make sure that no other cars will enter once this happens. Once all the cars have exited then I have called repair\_street(). All of this maximises concurrency and ensures that all the threads are synchronized and work together without deadlock. We can decrease our processing time using this.