Kaan ATMACA 28239

PA-3 REPORT

In this PA I used pthread_t, sem_t and pthread_mutex_t for the program work with no synchronization errors. First starting in the main I created two arrays containing threads. One for teamA and the other for teamB. Afterwards I created them one by one in two for loops. TeamA was calling findCarA function meanwhile teamB was calling findCarB function. Both functions are similar the only difference is one prints team a the other prints team b. So looking at findCarA function we firstly lock the mutex. Next print the pid and which team for looking for a car. Next we increament the variables total number of team members in a car. Then there are three different conditions:

- The first one is we have 4 team a players in the current car if so then we decrease the number and sem_post(&sema) //which allows us to make the waiting semaphore a's to not wait and find the correct car they are supposed to be in. // We sem_post three times since we have at least three members for the car and are waiting for us.
- Next condition is if we have more then 4 people in the car where at least 2 is from team a and two is from team b then we again decrement the numbers of the people and post semaphores. We will this time post one semaphore for team a (sema) and two for team b (semb). //if we were in teamb function then two sema and one semb is posted.
- Last condition is where we don't have enough peope yet for the car so we unlock the mutex and start waiting for the semaphore we want sema //if we were in teamb function then we would wait for semb. After we wait and get the posts we lock again and continue

Nextly once we get the posts and stop waiting we print that we spotted a car for the current pid. Nextly if the person for the car is the forth person we say that he/she is the captain and will drive the car. And lastly we unlock the mutex.

This is a correct implementation to complete the homework. For 4 spotted people in a car we have 1 captain. There is always 2-2, 4-0 or 0-4 from each team in the car. And also we do not have deadlocks. Hence the algorithm is correct.