Yunus Delipinar 26617

PA3

In this assignment I have used both semaphores and barriers because I think it is the best way to do it. Below you can find the pseudo code of my submission. I only wrote the function that threads used as it is the most important part.

Lock all the passengers(threads) using semaphores

While we have a room for a passengers

    Unlock passenger threads one by one until we don't have room

Print the "init" (looking for a car statement)

Use barrier to wait other passenger threads to print "init"

If 5 threads passed the lock

    then find the irrelevant one to manage car according to rules

    lock that one

after finishing the above steps update the variables used in upper part of the code

wait 4 passengers threads to finish using barriers

after 4 passengers has been chosen and they get passed the barrier

    print "mid" (I have found a spot in a car)

print the end only by one passenger thread using if statements and semaphore

unlock the other threads waiting at the beginning as we will have a new ride.

As you see in the code above using semaphores and barriers makes the code powerful and robust. By using these two mechanisms I assured that in every situation my code will work fine. Of course, the code itself is more complicated than it looks on the pseudo code but it is efficient as well. I think the nature of this question is about having a barrier that controls the other threads to run concurrently. To summarize, using both semaphores and barriers can produce the required format described in the assignment.