**READ ME Q2**

The code has been made in a way that all the conditions mentioned in the problem are satisfied. First the car loads, then passengers (in random order) board. Number of passengers in each ride do not exceed the car capacity. This is followed by car unloading and passengers un-boarding. The process runs until terminated.

s1, s2, s3: These semaphores are used for synchronization. s1 is initialized to 1, s2 and s3 are initialized to 0.

load(): Simulates the car loading. It prints "Car Loading..." and sleeps for 1 second.

unload(): Simulates the car unloading. It prints "Car Unloading..." and sleeps for 1 second.

pthread\_mutex\_lock() and pthread\_mutex\_unlock(): They lock and unlock a mutex protecting critical section of code.

Main function takes input for number of passengers and capacity of car and creates threads for the passengers and car.

Semaphores are used for synchronization. s2 is used for passengers to wait until the car is ready for boarding, and s3 is used for passengers to wait until the car is ready for un-boarding. This avoids any sort of concurrency bugs that can occur.

Error handling has also been done to ensure user friendly code.