start the program

initialize the global variables:
car_id = -1; aFan = 0; bFan = 0; driver_c = 0; driver_id;

in main:
if number of inputs in the command line not equal to 2{
  give the error "Usage: ./rideshare.c <A> <B>”}
else{
  if A is divisible by 2 and B is divisible by 2 and A+B is
     divisible by 4:
    initilaze the semaphores a,b, mutex with initial values 0, 0, 1
     respectively
    create the threads
    join all the threads}

give the message "The main terminates"

in fan_thread:
acquire the binary semaphore mutex which is a lock
state that you are looking for a car
if there is a valid combination including this thread{
    be the driver
    reset the fan numbers
    wake the threads that compose the valid combination}
else{
    release the mutex lock
    go to sleep --> (if Team A fan sem_wait(&a) else sem_wait(&b))}

state that you found a spot in a car

if you are the driver{

wait in spin lock until 4 "found a spot" is printed

increase the car id by one

state that you are the driver with the car id

release the mutex lock}

I used the semaphore "a" for Team A fans to wait until a suitable combination is composed for them.

I used the semaphore "b" for Team B fans to wait until a suitable combination is composed for them.

I used the semaphore mutex because when a valid combination occured I wanted the stop "looking for a car" print statements

I used global variables aFan and bFan to hold the number of waiting Team A fans and Team B fans.

I used car_id to hold the current cars id

(if there isn't any it is -1)

I used driver_id to hold the drivers id(the last there that composes the valid combination)

I used a spin lock before printing the "driver" print statement. I used it because I wanted the driver to wait until all of the 4 "found a spot" print statements to printed. after one printed I increased driver_c variable's value by 1. The driver spinned until driver_c becomes 4. after that I printed the "driver" statement and reset the driver_c to 0 before releasing the mutex lock.

That is how the "looking", "found a spot", and "driver" print statements have not interrupted each other in the output of my implementation.