Question 3

This software uses pthreads for multithreading, and semaphores for organizing them. There are two thread functions: 'left' and 'right'. These stand for cars coming from those directions. Two kinds of semaphores, 'bridgeSemaphore' and 'directionSemaphore', manage the car count on the bridge and ensure cars only come from one side at a time. The 'printMutex' prevents mixed-up output by making sure print commands get done one at a time. The main role starts semaphores and mutex, then builds threads for the cars on both sides. Then it waits for both threads to finish with 'pthread_join'. The software cleans up all resources used afterward, and avoids problems that can come up with multiple things happening at once by using semaphores wisely, protecting important parts with mutex, and making sure all threads get properly synchronized. All in all, the design makes sure the pretend car crossings happen correctly and in order.

Screenshot of a sample output for 4 cars on left and right given below: