1. Old Bridge

a. Correctness constraints

   i. At most 3 cars are on the bridge at a time

   ii. All cars on the bridge go in the same direction

   iii. Whenever the bridge is empty and a car is waiting, that car should get on the bridge

   iv. Whenever the bridge is not empty or full and a car is waiting to go the same direction as the cars on the bridge, that car should get on the bridge

   v. Only one thread accesses shared state at a time

b. Cars will be waiting to get on the bridge, but in two directions. Use an array of two condition variables, waitingToGo[2].

c. It will be necessary to know the number of cars on the bridge (cars, initialized to 0), and the direction of these cars if there are any (call it currentdirection). It will also be useful to know the number of cars waiting to go in each direction; use an array waiters[2].

d.
```
ArriveBridge(int direction) {
  lock.acquire();

  // while can't get on the bridge, wait
  while ((cars == 3) ||
        (cars > 0 && currentdirection != direction)) {
    waiters[direction]++;
    waitingToGo[direction].wait();
    waiters[direction]--;
  }

  // get on the bridge
  cars++;
  currentdirection = direction;

  lock.release();
}

ExitBridge() {
  lock.acquire();

  // get off the bridge
  cars--;
  // if anybody wants to go the same direction, wake them
  if (waiters[currentdirection] > 0)
    waitingToGo[currentdirection].signal();
  // else if empty, try to wake somebody going the other way
  else if (cars == 0)
    waitingToGo[1-currentdirection].broadcast();

  lock.release();
}
```