1)

1	P1	a b c d e	x=4, y=0
	P2	f g h i j	
2	P1	a b c d e	x=2, y=0
	P2	f g h i j	
3	P1	a b c d e	x=2, y=3
	P2	f g h i j	
4	P1	a b c d e	x=2, y=0
	P2	f g h i j	
5	P1	a b c d e	x=0, y=2
	P2	f g h i j	
6	P1	a b c d e	x=2, y=1
	P2	f g h i j	
7	P1	a b c d e	x=2, y=0
	P2	f g h i j	
8	P1	a b c d e	x=2, y=1
	P2	f g h i j	

There exist instances where a & f execute after g & before j or after c & before e, however they lead to the same answers as the others thus were omitted

2) 1 execution path

Thread 1	Thread 2
	if (done == 0)
done = 1	
pthread_cond_signal() → no thread is waiting for signal. signal is lost	pthread_cond_wait() ↑ waiting for signal forever because signal was lost

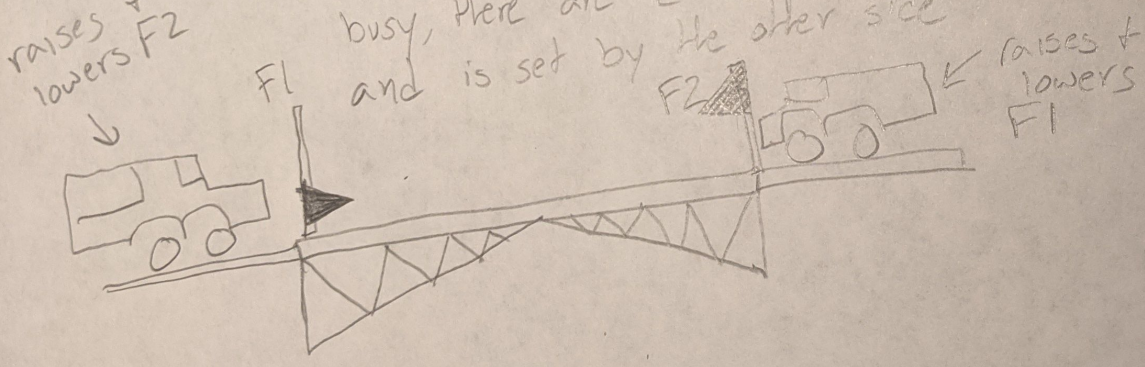
since there is no thread waiting on the conditional variable, the signal is lost. when the thread 2 reaches the wait, its too late as the signal was missed

3) Destroy the bridge, increase taxes, use tax money to hire civil engineer to build a 2-lane bridge that can handle multiple trucks in either direction

Im joking,

When the truck reaches the start of the bride, it turns on a flag on the other side of the bridge, then the driver looks at its own flag on its side of the bridge. if its down then drive across the bridge. once the driver is at the other side, lower the flag

The flag signals the other-side that the bridge is busy, there are 2 flags one each side and is set by the other side

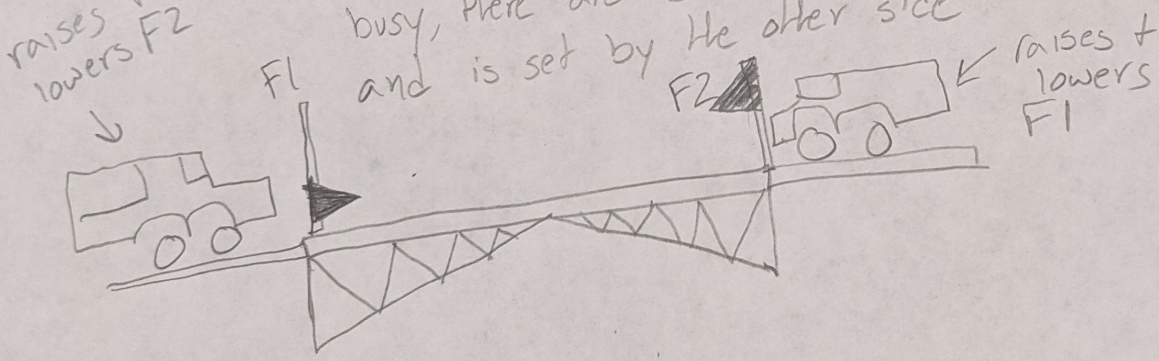


3) Destroy the bridge, increase taxes, use tax money to hire civil engineer to build a 2-lane bridge that can handle multiple trucks in either direction

Im joking,

When the truck reaches the start of the bride, it turns on a flag on the other side of the bridge, then the driver looks at its own flag on its side of the bridge. if its down then drive across the bridge. once the driver is at the other side, lower the flag

The flag signals the other-side that the bridge is busy, there are 2 flags one on each side and is set by the other side



```
num_adults = 0
adult_mutex = new_unlocked_mutex()

num_infants = 0
infant_mutex = new_unlocked_mutex()


def adult_enters():
    violation = false

    lock(adult_mutex)
    num_adults += 1

    if num_adults * 3 < num_infants:
        violation = true

    unlock(adult_mutex)

    if violation:
        # Phone number for chile protection services is listed below
        raise StateLicenseViolation("Calling 704-336-3000")

def adult_leaves():
    violation = false

    lock(adult_mutex)
    # not checking for negative nums cause
    # a non existant adult cant leave a room
    num_adults -= 1

    if num_adults * 3 < num_infants:
        violation = true

    unlock(adult_mutex)

    if violation:
        # Phone number for child protection services is listed below
        raise StateLicenseViolation("Calling 704-336-3000")

def infant_enter():
    violation = false

    lock(infant_mutex)

    num_infants += 1

    if num_adults * 3 < num_infants:
        violation = true

    unlock(infant_mutex)

    if violation:
        raise StateLicenseViolation("GooGoo GaaGaa? I don't \
            know if infants know CPS's number")

def infant_leaves():
    violation = false

    lock(infant_mutex)
    # not checking for negative nums because
    # a nonexistant infant cant leave a room
    num_infants -= 1

    if num_adults * 3 < num_infants:
        violation = true

    unlock(infant_mutex)

    if violation:
        raise StateLicenseViolation("GooGoo GaaGaa? I don't \
            know if infants know CPS's number")
```

5) The first issue is there is no lock acquisition ordering, and could result in a deadlock

If poped item is null, locks aren't released

If stack 2 poped item is null, the stack 1 item is lost (isnt returned to stack)