

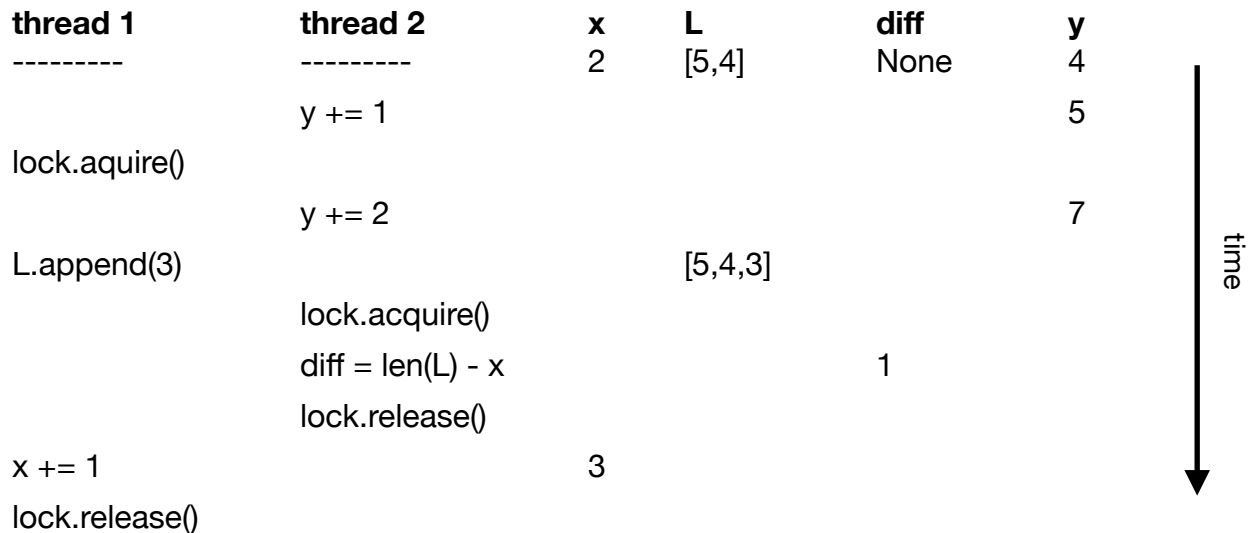
CS 544, Locks Worksheet

```

thread 1
lock.acquire()
L.append(3)
x += 1
lock.release()
    
```

```

thread 2
y += 1
y += 2
lock.acquire()
diff = len(L) - x
lock.release()
    
```



Problem 1: which statement executions above **are not possible** in a correct locking system? Which statements would cause exceptions? If the locking system behaved correctly, what would be possible values for diff at the end?

```

thread 1
if q != 0:
    lock.acquire() #A
    lock.acquire() #B
    r = 1/q #C
    lock.release() #D
    
```

```

thread 2
lock.acquire() #X
q = 0 #Y
lock.release() #Z
    
```

Problem 2: assume q is 2 before the threads start running. Write out an interleaving (for example, something like A, B, C, ...) that leads to an ZeroDivisionError.

```

lock = threading.Lock()
x = 1

def task():
    global x
    with lock:
        x = 2

t = threading.Thread(target=task)
a = x
t.start()
with lock:
    b = x
t.join()
c = x

```

a = _____
b = _____
c = _____

Problem 3: how do a, b, and c end? Write "?" if it is impossible to know.

```

thread 1
lockA.acquire()
lockB.acquire()
A += 1
B -= 1
lockA.release()
lockB.release()

```

```

thread 2
lockB.acquire()
lockA.acquire()
B += 2
A -= 2
lockB.release()
lockA.release()

```

thread 1

thread 2

A
30

B
40



Problem 4: write an interleaving that leads to "deadlock" (both threads blocked).