

## Quiz 4

1. (2pts) Why would you choose to use threads instead of processes when writing an application?

**Threads are more lightweight than processes, sharing the same set of process resources and memory space. They would be a good choice for parallelizable tasks that need access to the same memory space.**

2. (2pts) What is the difference between kernel and user thread managers? Which one does the python threading system resemble?

**Kernel thread managers are implemented in the operating system. Since they have access to hardware interrupts, the scheduler of the kernel thread manager can preempt a running thread. Threads managed by a user thread managers will need to yield if they wish to give up the CPU. The python threading system is a blend of both, since threads are run for a fixed number of virtual machine instructions rather than some timesplice or needing to yield.**

3. (4pts) Describe the problem with the code below and fix it to prevent the violation of the first law of thermodynamics while allowing the philosophers to eat as often as possible. (For 0.5pt extra credit, What is the first law of thermodynamics?)

**The code below needs locks to control access to the shared global chopsticks, that maintains the state of the chopsticks in the system. Without the locks, extra chopsticks can be created for use by multiple philosophers.**

**The first law of thermodynamics is the conservation of mass-energy: In any system, mass and energy are neither created nor destroyed.**

```
import threading
chopsticks = []

class DiningPhilosophers():
    def __init__(self, num):
        global chopsticks
        global choplock
        chopsticks = [1] * num
        choplock = Threading.Lock()
        for i in range(num):
            t = DiningPhilosopher(i)
            t.start()
```

```
class DiningPhilosopher(threading.Thread):
    def __init__(self, id):
        self.id = id

    def run(self):
        while True:
            think()
            trytoeat()

    def trytoeat(self):
        global chopsticks
        #check left and right chopsticks
        choplock.acquire()
        if chopsticks[self.id] and chopsticks[self.id-1]:
            #pick up left chopstick
            chopsticks[self.id] -= 1
            #pick up right chopstick
            chopsticks[self.id-1] -= 1
            choplock.release()

            eat()

            choplock.acquire()
            #put down left chopstick
            chopsticks[self.id] += 1
            #put down right chopstick
            chopsticks[self.id-1] += 1
            choplock.release()
```