The Need for Synchronization:
Recall, when we ended last week, we had multiple threads counting up -- one by one -- and had various unexpected results when running the code below:

```
int ct = 0;
void *thread_start(void *ptr) {
  int countTo = *((int *)ptr);
  int i;
  for (i = 0; i < countTo; i++) {
    ct = ct + 1;
  }
  return NULL;
}
```

A __________________________ is any code that accesses a shared resource that must be accessed only by a single thread at a given time to function correctly.

Synchronization: Using Locks
The simplest way to protect a region of code from being accessed is through the use of a __________________________:

```
void *thread_start(void *ptr) {
  int countTo = *((int *)ptr);
  int i;
  for (i = 0; i < countTo; i++) {
    pthread_mutex_lock(&lock);
    ct = ct + 1;
    pthread_mutex_unlock(&lock);
  }
  return NULL;
}
```

A __________________________ is any code that accesses a shared resource that must be accessed only by a single thread at a given time to function correctly.

```
int main(int argc, char *argv[]) {
  // Parse Command Line:
  if (argc != 3) {
    printf("Usage: %s <countTo> <thread count>\n", argv[0]);
    return 1;
  }
  const int countTo = atoi(argv[1]);
  if (countTo == 0) { printf("Valid `countTo` is required.\n"); return 1; }
  const int thread_ct = atoi(argv[2]);
  if (thread_ct == 0) { printf("Valid thread count is required.\n"); return 1; }
  // Create Lock:
  pthread_mutex_init(&lock, NULL);
  pthread_mutex_init(&lock, NULL);
  [...code continues the same as last week...]
```

Q: What happens when we run this code now?
Q: What is the performance of this code vs. the code without the lock?
Critical Sections
We know that critical sections require exclusive access to a resource. We also know locking a resource is computationally expensive. However, are there other concerns?

The Dining Philosophers
Imagine five philosophers and five chopsticks at a circular table. Each philosopher has two states: eating and thinking:
- When a philosopher is thinking, she holds no chopsticks.
- When a philosopher starts the process of eating, she must take the chopstick to her left, then her right, and then begin eating.

Q: Using the strategy described above (take left, take right, then eat), what happens over a long period of time?

![Diagram of philosophers and chopsticks]

Q: What happens when we run this thread for all five philosophers?
Deadlock:
- Definition:

- Four necessary conditions of deadlock:
  1)
  2)
  3)
  4)

---

Solution #1:

---

Solution #2:

---

Solution #3:

---

Solution #4: