1. Threads
   a) Under what circumstances does a multi-threaded solution using multiple kernel threads provide
      better performance than a single-threaded solution? Assume we are using a processor with a
      single core.

   b) Can a multi-threaded solution using only user-level threads achieve better performance on a
      multiprocessor system than on a single processor system? Explain.

2. Locks:
   a) The following code attempts to implement a lock in the same manner that bakeries serve
      customers. A customer takes a ticket with a number on it when they enter, then approaches the
      counter to be helped when their number is called. Does the following code properly implement
      a lock primitive? If yes, please explain how it meets our 3 criteria for critical sections. If no,
      please demonstrate a case where it fails. Assume that the critical section always finishes and
      that all calls to acquire and release are properly paired.

```
struct lock
{
    int next_ticket=0;
    int now_serving=0;
};

acquire(lock)
{
    int my_num = lock->next_ticket++;
    while( my_num != lock->now_serving );
}

release(lock)
{
    lock->now_serving++;
}
```

(Hint: Review our threading assumptions from Lecture 3)
b) Imagine you are given an atomic instruction called Swap:

```c
void swap(int *x, int *y) {
    int temp = *x;
    *x = *y;
    *y = temp;
}
```

Implement a lock using the Swap method. Provide the acquire() and release() methods, as well as any other code/data structures that might be needed. Note that int *x means the address of an integer, and performing the *x dereference operation means the value at the address (i.e., the value of the integer). You may not disable interrupts in your solution.

3. How does the signal() operation associated with condition variables differ from the corresponding operation defined for semaphores?

4. Web servers can be designed to limit the number of open connections. For example, a server may wish to have only N socket connections at any point in time. As soon as N connections are made, the server will not accept another incoming connection until an existing connection is released. Explain how semaphores can be used by a server to limit the number of concurrent connections.