1. Consider a computer that does not have a TSL instruction does have an instruction to swap the contents of a register and a memory word in a single indivisible action. Can that be used to write a routine enter_region such as the one written with TSL instruction (that is in the book and that we have seen in the class)?

2. What is a race condition?

3. Suppose that we have a message passing system using mailboxes. When sending to a full mailbox or trying to receive from an empty one, a process does not block. Instead it gets an error code back. The process responds to the error code by just trying again, over and over, until it succeeds. Does this scheme leads to race conditions.

4. Measurements of certain systems have shown that the average process runs for a time T before blocking on I/O. A process switch requires a time S, which is effectively wasted (overhead). For round-robin scheduling with quantum Q, give a formula for the CPU efficiency for each of the following:
   a. $Q = \infty$
   b. $Q > T$
   c. $S < Q < T$
   d. $Q = S$
   e. $Q$ nearly 0 (zero)

5. Consider the following program written in pseudo-code.
   a. Determine the proper lower bound and upper bound on the final value of the shared variable x. Assume threads can execute at any relative speed and that a value can only be incremented after it has been loaded into a register by a separate machine instruction.
   b. Suppose now M threads (M > 2) will execute in parallel (instead of just two) under the assumptions of part (a). What effect will this modification have on the range of final values of x?

```c
int n = 50; /* shared but never modified */
int x;  /* shared variable */
void foo() {
    int i;
    for (i = 1; i <= n; ++i) {
        x++;
    }
}
void main() {
    x = 0;
    pthread_create(..., &foo, ...);
    pthread_create(..., &foo, ...);
    wait_until_all_created_threads_terminate();
    print (x);
}
```