cs372: Operating Systems
Homework 3
Due Monday, September 28th in class

---

1. Is the following a correct solution to the critical section problem? Argue whether or not the alleged solution ensures mutual exclusion, progress, and bounded waiting.

```
boolean blocked[2] = {false, false};
int turn = 0;

thread t0()
{
  while(true)
  {
    blocked[0] = true;
    while (turn != 0)
    {
      while (blocked[1]) { }
      turn = 0;
    }
    CRITICAL SECTION;
    blocked[0] = false;
    REMAINDER SECTION;
  }
}
```

```
thread t1()
{
  while (true)
  {
    blocked[1] = true;
    while (turn != 1)
    {
      while (blocked[0]) { }
      turn = 1;
    }
    CRITICAL SECTION;
    blocked[1] = false;
    REMAINDER SECTION;
  }
}
```

2. Consider the solution to the critical section problem discussed in class, in which we used a `turn` variable and in which we forced a strict alternation of threads:

```
int turn = 0;

thread t0()
{
  while (true)
  {
    while (turn == 1)
      ;
    CRITICAL SECTION;
    turn = 1;
    REMAINDER SECTION;
  }
}
```

```
thread t1()
{
  while (true)
  {
    while (turn == 0)
      ;
    CRITICAL SECTION;
    turn = 0;
    REMAINDER SECTION;
  }
}
```

Suppose this algorithm is implemented on a two processor system in which a single memory is shared across a system bus. Simultaneous access to the same memory cell is arbitrated in some random fashion, but each access is atomic. Each process is initiated on its own processor, and the processors have unpredictable speeds. Argue whether or not the algorithm still provides mutual exclusion. What about progress and bounded waiting?

3. Suppose you have to implement the Lock/Unlock primitives on a uniprocessor machine which does not support the atomic Test-and-Set and Exchange instructions. Fortunately, there is an atomic instruction called `ijz`, which will increment the contents of a memory location and, if the incremented value is zero, will jump to a specifi

   ed label. If the incremented value is non-zero, the statement following the ijz will be executed. More precisely,

   ```
   ijz(m, label) is defined as: <m = m + 1; if (m == 0) goto label;>
   ```

   Pseudocode surrounded by angle braces $< \ldots >$ is atomic (i.e., done in "memory-interlock" mode: memory accesses within angle braces are serialized by the memory hardware, even in a multiprocessor system). Implement Lock/Unlock on this machine, using `ijz` and any "conventional" C/C++ statements you may need. Argue for the correctness of your implementation. You may not use interrupt disabling as part of your solution.

4. Using only semaphores, give pseudocode to implement a "readers favored" solution to the Readers-Writers problem, and include a description describing how it works. The solution must allow multiple readers into the critical section for readers, but at most one writer may be in the writers' critical section at a time, and must never overlap with any readers.

I am asking for your honor in not going Internet searching (or any other external sources) for solutions to any of these problems.