Final Exam

Due Date: Thursday, March 23, 2000 at 6:00PM (the end of the reglar final exam period for this class)
If you want to submit this electronically, please email it to me at bishop@cs.ucdavis.edu.

1. Suppose that, instead of a first-come first-served semaphore queueing discipline, queueing is last-come, first-served. However, we still want to implement *safe* mutual exclusion for the critical sections of *n* cyclic processes. *Safe* means that no process can block indefinitely while other processes cycle arbitrarily often. For example, processes p_1 and p_2 can block p_3 forever if p_3 arrives second and p_1 and p_2 cycle alternately through their critical sections.

   Consider a binary tree with *n* leaves, one for each process. Each internal node has an associated semaphore. The processes desiring to enter their critical sections perform *down* operations on successive semaphores from their leaves to the root of the tree, execute their critical section, and perform *up* operations in reverse order back down the tree.

   a. Show this scheme implements safe mutual exclusion.

   b. Discuss how the shape of the binary tree can affect the relative speeds of the processes and enforce a priority scheme.

2. This problem asks you to apply a modification of Havender's ordered resource policy to processes, rather than resources. Given that the mutual exclusion, hold and wait, and no preemption conditions are in place, consider the following strategy: All processes are given unique priorities. When more than one process is waiting for a resource and the resource becomes available, allocate the resource to the waiting process with the highest priority. Either prove this prevents deadlock or give an example in which this strategy does not prevent deadlock.

3. Use Holt's graph-based model to prove that the collective requests policy, in which a process requests and acquires all its needed resources at once, prevents deadlock.

4. Prove that Maekawa's distributed mutual exclusion algorithm prevents deadlock.

5. Show that Byzantine agreement cannot always be reached among four processors when *two* processors are faulty.