1. (a) Trace Peterson’s algorithm for all significantly different cases. You can apply symmetry considerations to reduce a lot of work. For example, you should assume that in all cases, the operation interested[process]=TRUE executed by process (thread) 0 is done first. Demonstrate that the algorithm produces
   i. mutual exclusion,
   ii. no deadlocks, and
   iii. access to the critical region for each process that requests it.
   in ALL cases.

(b) Figure out and demonstrate (with a specific interleaving) what goes wrong if Peterson’s solution were modified so the turn=process operation is replaced by turn=other. (In other words, each process sets turn to command the other process to wait.)

(c) Figure out and demonstrate (with a specific interleaving) what goes wrong if Peterson’s solution were modified so the operations interested[process]=TRUE and turn=process were programmed in reverse order in the programs of both processes.

(d) Figure out and demonstrate (with a specific interleaving) what goes wrong if Peterson’s solution were modified so the operations interested[process]=TRUE and turn=process were programmed in reverse order in the programs of the first process (process 0).

2. What goes wrong if the operations down(empty) and down(mutex) were programmed in reverse order in the producer’s function in Figure 2-24? Demonstrate with one specific interleaving that produces trouble.


