Answer Set 1

15 FEBRUARY, 1993

1. Extend Dekker's algorithm to negotiate N processes competing for the same resource.

   The "classic" solution is by E. W. Dijkstra and appears in: "Solution of a Problem in Concurrent Programming Control", Communications of the ACM, Vol. 8, No. 9, 1965. This solution is from G. L. Peterson's "Myths About the Mutual Exclusion Problem" appearing in Information Processing Letters, Vol. 12, No. 3, 1981. For n processes, one defines arrays *Q[1..n]* and *TURN[1..n-1]*. The first is initially all 0, the second, all 1.

   ```
   /* my process number is i */
   for j := 1 to n-1 do
   begin
     Q[i] := j ;
     TURN[j] := i ;
     wait until ( for all k <> i, Q[k]<j ) OR TURN[j] <> i
   end ;
   Critical Section ;
   Q[i] := 0
   ```

2. Use P/V Semaphores to implement a Monitor.

   For each monitor with k signals we will use $k+1$ semaphores

   $$M, S_1, \ldots, S_k$$

   and k integer variables

   $$Q_1, \ldots, Q_k.$$

   INITIALIZATION: Let $M = 1$ and $S_i, Q_i = 0$, for $i = 1, \ldots k$.

   ENTRY/EXIT: All monitor code will be guarded by semaphore M. Any user must call $P(M)$ to gain entrance to the monitor and, except for a special *signal* exit, must call $V(M)$ when it exits.

   WAIT/SIGNAL: To Wait on signal i, the variable Q_i is incremented, then $V(M)$ is called, then $P(S_i)$ is called. To signal i, the variable Q_i

is checked for non-zero. If it is zero, the process just exits, performing a $V(M)$. If it is non-zero, the process decrements Q_i, does a $V(S_i)$ and immediate exits the monitor *without* doing a $V(M)$.

DISCUSSION: A signaling process turns over its monitor permission to the signaled process. Therefore, if it signals and no process is waiting, the monitor will become dead. That is why the integer Q_i must be maintained.

3. Use a Monitor to implement a P/V Semaphore.

   A monitor is created for each semaphore. Each monitor has one signal, *sleep*, and an internal integer variable S. This variable is initialized to the same value as the semaphore. The monitor has two entry points: P and V. A call to $P(A)$ will be changed by the operating system into an attempt to enter by P the monitor A.

   P: Check S for non-zero. If it is not zero, decrement S and exit the monitor. Else, do a *Wait(sleep).* When awakened, decrement S and exit the monitor.

   V: Increment S and do a *Signal(sleep).*