CSE 430 – Operating Systems – Fall 2015
Assignment: #3
Due date: Nov 16, 2015. (hardcopy, hand-in, in class)
(Not all questions will be graded)

Note: You may choose to do this HW in groups of 2 or do it individually. Your choice, but if done in a group submit only 1 copy with two names.

Q1]
Suppose a system has an **atomic** hardware instruction SHIFT, that does the follows:

```
SHIFT ( int *A, *B ) {
 *B =*A; // ATOMICALLY
  *A = 0
 }
```

A] Implement Dijkstra style semaphores with the shift instruction, that is, semaphores which utilize busy waiting.

B] Implement blocking semaphores using the shift instructions.

Q2] // same as a sample exam question

There are 4 processes, executing concurrently. Process P0 is in an infinite loop, incrementing the value of the variable *x* (*x* is initialized to 0). P_0 is the only process that changes the value of *x*.

The rest of the processes P*i* (1<= *i* <=3) monitor the value of *x*. Whenever *x* reaches a value such that it is divisible by *i*, P*i* prints the value of *x*. For example, P3 will print the sequence 3 6 9 12 ….. as the value of *x* reaches 3, 6, 9, 12 and so on.

Write the code for all the 4 processes using semaphores. Note that P1 - P3 should be identical; also P*i* determines whether x is to be printed, and this decision is not made by P_0.

Q3]

A synchronization mechanism consists of 2 *atomic* routines, ENQ(r) and DEQ(r). "r" is a resource variable that has two fields, inuse (boolean) and queue (a queue) which is a queue of processes waiting to acquire the resource. The definitions of ENQ and DEQ are:

```
ENQ(r) : if (r.inuse==1) then begin
             insert current process in r.queue
             block
           end
         else  r.inuse = 1;

DEQ(r) : if r.queue == nil then inuse = false
         else delete a process from r.queue and activate it.
```

Construct an implementation of ENQ/DEQ using semaphores. You can use other variables, etc that you need, but no other atomic code or synchronization constructs other than P or V can be used.

Q4]
Do the reverse of Q3, that is implement Semaphores using ENQ/DEQ.