CS 620 hw 7

Instructions:

1. Please type your answers in order, by question number. If there’s a question you don’t know how to answer, type the question number and leave blank. If the TA has difficulty locating your final answer(s), you will lose credit.
2. Late policy: 1 days late: 1 points off. 2 days late: 2 points off. more than 2 days late: will not be graded.
3. You need to submit ONE PDF file on Canvas. If you have multiple PDF files, convert into one, or 4 points will be taken off.
4. Handwritten assignments will not be graded.

Questions:

1. (3) The semaphore P() operation is a blocking wait function, but it is implemented using the busy-wait hardware atomic function. Is this an inconsistency? Does this lower the efficiency of semaphores?

No, This is not an inconsistency and does not lower the efficiency of semaphores, because if the resource is unavailable, the thread gets added to the semaphore queue instead of repeatedly checking until the resource becomes available.

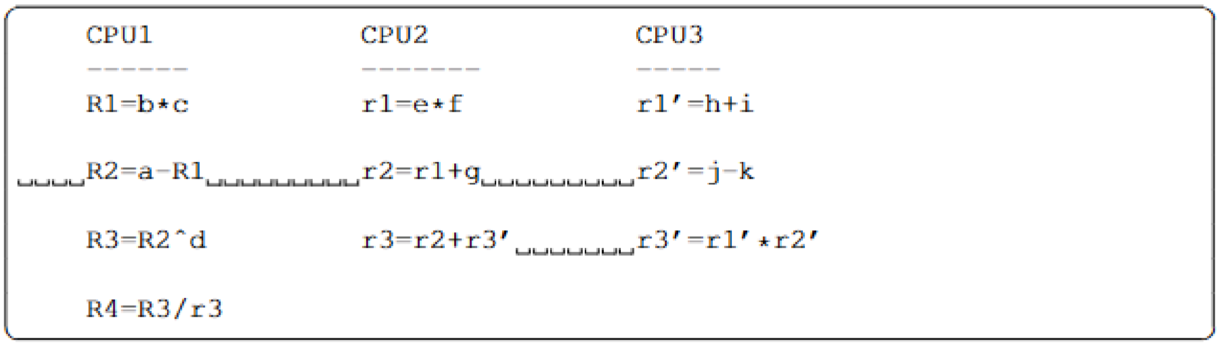
2. (13) Suppose the following computation were to be done in parallel on a set of parallel processors. Each processor is assumed to be able to do only one arithmetic operation per time step. The synchronization primitives P() and V() execute quickly, so assume that they execute in 0 time steps.

$$\frac{(a - b * c)^d}{((e * f) + g) + ((h + i) * (j - k))}$$

(a) (8) Fill in the following table:

# processors	min # of req time steps	min # of req semaphores(for min # of steps)
1	10	0
2	7	1
3	5	2
4	4	3

(b) (5) The following code is designed for 3 CPUs. Please add the minimum number of semaphores, specifying their initial values, and fill in the appropriate P() and V() operations in the code. This will ensure proper synchronization and access control to the critical section for all three threads.



```
     CPU1              CPU2               CPU3
     ------            -------            -----
     R1=b*c            r1=e*f             r1'=h+i
␣␣␣␣R2=a-R1␣␣␣␣␣␣␣␣␣␣r2=r1+g␣␣␣␣␣␣␣␣␣␣r2'=j-k
     R3=R2^d           r3=r2+r3'␣␣␣␣␣␣␣r3'=r1'*r2'
     R4=R3/r3
```

Semaphores a=b=0

CPU 1	CPU2	CPU3
R1=b*c	r1 = e*f	r1' = h+i
R2=a-R1	r2 = r1 + g	r2'=j-k
R3=R2^d	P(b)	r3'=r1'*r2'
P(a)	r3=r2+r3'	V(b)
R4=R3/r3	V(a)	

3. (18) The following is a critical section code. Initially, flag1 = flag2 = false.

```
T1                                      T2
for(;;)                                 for(;;)
{                                       {
   ...                                     ...
 1 flag[1]=true;                         A while (flag[1]==true);
 2 while (flag[2] == true);              B flag[2]=true;
 3 <critical section>                    C <critical section>
 4 flag[1]=false;                        D flag[2]=false;
 5 <non-critical section>                  <non-critical section>
   ...                                     ...
}                                       }
```

† ✆

The codes for the two threads are asymmetric.
An example of a sequence of operations is: 1, 2, A, B, C, 3, 4, 5, D, E, 1, 2

(a) (4) Can mutual exclusion of the critical section be violated?
If your answer is YES, show a sequence of operations that shows the violation.
If your answer is NO, just state No.

Yes:
1, A, B, C, 2, 3

(b) (4) Can T1 starve T2?
If your answer is YES, then show a sequence of operations that shows starvation. If your answer is NO, just state NO.

No

(c) (4) Can T2 starve T1?
If your answer is YES, then show a sequence of operations that shows starvation. If your answer is NO, just state NO.

Yes:
1, 2, A, B, C, D, 2, A, B, C,D

(d) (4) Can the code deadlock?
If your answer is YES, then show a sequence of operations that shows deadlock. If your answer is NO, just state NO.

No

(e) (2) If both flags are initialized to true, would this solution work?
Just state YES or NO.

No.

4. (10) Consider the following parallel program:

```
Thread x:              Thread y:               Thread z:
for(;;)                for(;;)                 for(;;)
{                      {                       {
 P(x)                   P(y)                    P(z)
 P(x)                                           P(z)
 P(x)
 print "X"              print "Y"               print "Z"
 V(y)                   V(z)                    V(x)
 V(y)                   V(x)                    P(z)
 V(z)
}                      }                       }
```

initially, x = 3; y = z = 0. This code could give different outputs.

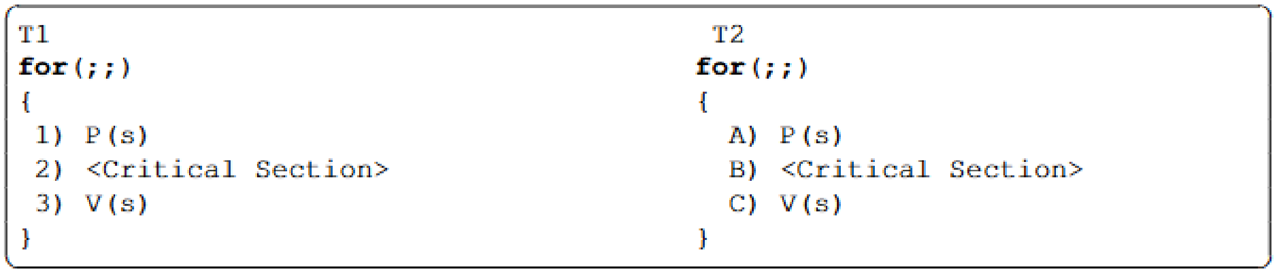
(a) Give one possible output of the first five print statements.

XYZYXYZY… (repeating)

(b) Give a second possible output of the first five print statements.

XYYZXYYZ…(repeating)

5. (6) Consider the following code. Initial value of semaphore s=1. Assume T1 enters the critical section first (at step 2), and before T1 reaches step 3, T2 executes step A. The sequence of actions is: 1, 2, A.



```
T1                                       T2
for(;;)                                 for(;;)
{                                       {
 1) P(s)                                  A) P(s)
 2) <Critical Section>                    B) <Critical Section>
 3) V(s)                                  C) V(s)
}                                       }
```

† ✆

Question: Can T1 re-enter the critical section before T2 does?
Explain your answer with respect to the semaphore queue and the semaphore value. Show the minimum sequence of operations before T1 enters the critical section for the second time.

No, when step 1 is executed the value of semaphore s is decremented from 1 to 0, so T2 is blocked when step A is executed and T2 is put onto the semaphore queue. So when step 3 is executed T2 is woken from the semaphore queue and step B is executed.

Minimum sequence of operations before T1 enters the <CS> for the second time:
1, 2, A, 3, B, C,1, 2