Full name : Heng Ngounhak

ID : e20170254

Group : I4.gic.A

Note: Please write your answer(s) according to your understanding, no copy from books or internet.
Copy will result in minus scores.

FINAL EXAM
Close Exam
120 minutes

1. Give definitions to these terms: Critical Region, Semaphores, Mutual Exclusion.

- Critical Region: a part of a program that must complete execution before other processes can begin.
- Semaphores: a type of shared data item that may contain either binary or nonnegative integer values and is used to provide mutual exclusion.
- Mutual Exclusion: one of four conditions for deadlock in which only one process is allowed to have access to a resource. It is typically shortened to mutex in algorithms describing synchronization between processes.

2. Three concurrent processes X, Y, and Z execute three different code segments that access and update certain shared variables. Process X executes the P operation (i.e., wait) on semaphores a, b and c; process Y executes the P operation on semaphores b, c and d; process Z executes the P operation on semaphores c, d, and a before entering the respective code segments. After completing the execution of its code segment, each process invokes the V operation (i.e., signal) on its three semaphores. All semaphores are binary semaphores initialized to one. Which one of the following represents a deadlockfree order of invoking the P operations by the processes?
    a. X: P(a)P(b)P(c) Y:P(b)P(c)P(d) Z:P(c)P(d)P(a)
    b. X: P(a)P(b)P(c) Y:P(c)P(b)P(d) Z:P(c)P(d)P(a)
    c. X: P(b)P(a)P(c) Y:P(c)P(b)P(d) Z:P(a)P(c)P(d)
    d. X: P(b)P(a)P(c) Y:P(b)P(c)P(d) Z:P(a)P(c)P(d)

➔ Answer is d
➔ Because a, b, c can cause the deadlock by process need correct acquired semaphores

3. Consider the procedure below for the Producer-Consumer problem which uses semaphores:

```
semaphore n = 0;
semaphore s = 1;
void producer()
{
      while(true)
      {
       produce();
       semWait(s);
       addToBuffer();
       semSignal(s);
       semSignal(n);
      }
}
```

```
void consumer()
{
      while(true)
      {
       semWait(s);
       semWait(n);
       removeFromBuffer();
       semSignal(s);
       consume();
      }
}
```

Which one of the following is TRUE?

a. The producer will be able to add an item to the buffer, but the consumer can never consume it.
b. The consumer will remove no more than one item from the buffer.
c. Deadlock occurs if the consumer succeeds in acquiring semaphore s when the buffer is empty.
d. The starting value for the semaphore n must be 1 and not 0 for deadlock-free operation.

➔ Answer is c
➔ Because Semaphore s=1 and semaphore n=0.semWait(s) decrements the value of semaphore 's' .so s = 0 and semWait(n) decrements the value of semaphore 'n'.Since, the value of semaphore 'n' becomes less than 0 , the control stucks in while loop of function semWait() and a deadlock arises.

4. Consider two processes P1 and P2 accessing the shared variables X and Y protected by two binary semaphores SX and SY respectively, both initialized to 1. P and V denote the usual semaphore operators, where P decrements the semaphore value, and V increments the semaphore value. The pseudo-code of P1 and P2 is as follows :

P1 :

```
While true do {
  L1 : .................
  L2 : .................
  X = X + 1;
  Y = Y - 1;
  V(SX);
  V(SY);
}
```

P2:

```
While true do {
  L3 : .................
  L4 : .................
  Y = Y + 1;
  X = Y - 1;
  V(SY);
  V(SX);
}
```

In order to avoid deadlock, the correct operators at L1, L2, L3 and L4 are respectively

a. P(SY), P(SX); P(SX), P(SY)
b. P(SX), P(SY); P(SX), P(SY)
c. P(SX), P(SX); P(SY), P(SY)
d. P(SX), P(SY); P(SY), P(SX)

➔ Answer is b
➔ Because p1: line1 and p2: line3 block because need sx
➔ p1: line2 and p2: still block
➔ p1: execute cs then up the value of sx
➔ p2 :line 3 line 4 (block need sy)

➔ p1 up the sy
➔ p2: line4 and easily get cs.
➔ So no deadlock

5. The following two functions P1 and P2 that share a variable B with an initial value of 2 execute concurrently.

```
P1()
{
   C = B – 1;
   B = 2*C;
}

P2()
{
   D = 2 * B;
   B = D - 1;
}
```

The number of distinct values that B can possibly take after the execution is

a. 2
b. 3
c. 4
d. 5

➔ Answer is b
➔ Because

C = B – 1; // C = 1
B = 2*C; // B = 2
D = 2 * B; // D = 4
B = D - 1; // B = 3

C = B – 1; // C = 1
D = 2 * B; // D = 4
B = D - 1; // B = 3
B = 2*C; // B = 2

C = B – 1; // C = 1
D = 2 * B; // D = 4
B = 2*C; // B = 2
B = D - 1; // B = 3

D = 2 * B; // D = 4
C = B – 1; // C = 1
B = 2*C; // B = 2
B = D - 1; // B = 3

D = 2 * B; // D = 4

B = D - 1; // B = 3

C = B – 1; // C = 2

B = 2*C; // B = 4

6. Two processes X and Y need to access a critical section. Consider the following synchronization construct used by both

```
Process X
/* other code for process X */
while (true)
{
    varP = true;
    while (varQ = = true)
    {
    /* Critical Section */
        varP = false;
    }
}
/* other code for process X */
```

```
Process Y
/* other code for process Y */
while (true)
{
    varQ = true;
    while (varP = = true)
    {
        /* Critical Section */
            varQ = false;
    }
}
/* other code for process Y */
```

Here, varP and varQ are shared variables and both are initialized to false. Which one of the following statements is true?

a. The proposed solution prevents deadlock but fails to guarantee mutual exclusion.
b. The proposed solution guarantees mutual exclusion but prevent deadlock.
c. The proposed solution guarantees mutual exclusion but fails to prevent deadlock.
d. The proposed solution fails to prevent deadlock and fails to guarantee mutual exclusion.

➔ Answer is c

➔ Because When both processes try to enter critical section simultaneously, both are allowed to do so since both shared variables varP and varQ are false. So, clearly there is mutual exclusion. But deadlock is not prevented because mutual exclusion is not one of the four conditions to be satisfied for deadlock to prevent.