Branch : CSE & IT **Batch : Hinglish**

WEEKLY TEST – 03

Subject : Operating System

Topic : Process Synchronization / Coordination

Maximum Marks 15

Q.1 to 5 Carry ONE Mark Each

[NAT]

1. At a particular time of computation, the value of a counting semaphore is 12. Then 12P operation and xV operations were performed on the semaphore. If the final value after all wait and signal operations is 8, then value of x is _______.

[MCQ]

2. Consider the following proposed solution to critical section problem for two process i and j. For process Pi j will be 1-i :

```
do{
      flag[i] = True;
      turn = j;
      while (flag[j] && turn = = j);
            <Critical Section>
      flag[i] = true;
      <remainder section>
    }
while(true);
```

The above solution satisfies

(a) Mutual Exclusion and Progress.

(b) Only Mutual Exclusion.

(c) Mutual Exclusion and Bounded Waiting.

(d) Mutual Exclusion, Progress and Bounded Waiting.

[MCQ]

3. Consider the definition of operation's performed on semaphore:

```
wait(S){                Signal(S){
   while(X);               Z;
   Y                    }
```

Which of the following represents correct value of X, Y and Z respectively?

(a) S < 0; S – –; S + +

(b) S ≥ 0; S + +; S – –

(c) S ≤ 0; S – –; S + +

(d) S > 0; S + +; S – –

[NAT]

4. Assume that 'A' is counting semaphore. Consider the following program segment:

A = 12;

P(A);

V(A);

P(A);

V(A);

V(A);

P(A);

P(A);

P(A);

P(A);

What is the value of 'A' at the end of the above program execution?

[MCQ]

5. Which of the following is the guaranteed solution of avoidance of mutual exclusion?

(a) Semaphore.

(b) Monitors.

(c) Banker's algorithm.

(d) All of these.

Q.6 to 10 Carry TWO Mark Each

Common Data for Question 6 and 7

```
Semaphore mutex = 1;
Semaphore empty = N;
Semaphore full = 0;
void producer (void)
{
    int itemp;
    while(true)
    {
     producer-item (itemp);
    down (empty);
    down (mutex);
    buffer[in] = itemp;
    in = (in + 1) mod N;
    up(mutex);
    up(full);
    }
}
void consumer (void)
{
    int itemc;
    while(true)
    {
    down (full);
    down (mutex);
    itemc = buffer[out];
    out = (out + 1)mod N;
    up(mutex);
    up(empty);
    process-item(itemc);
    }
}
```

[MCQ]

6. What happens if we interchange down (empty), down (mutex) in the producer code ___

(a) No problem, the solution still work correct.

(b) Both consumer and producer will access the buffer at same time.

(c) Some of the item produced by the producer will be lost.

(d) It is possible for deadlock.

[MCQ]

7. What happens if we interchange down (full), down (mutex) in the consumer code ___

(a) No problem, the solution still work correct.

(b) Both consumer and producer will access the buffer at same time.

(c) Some of the item produced by the producer will be lost.

(d) It is possible for deadlock.

[MCQ]

8. Consider two process P_1 and P_2 accessing the shared variable X = 10 and Y = 20 protected by two binary semaphore B_x and B_y respectively, both initialized to 1. P and V denote the usual semaphore operations

P_1	P_2
L_1: P(B_X)	L_3: P(B_X)
L_2: P(B_Y)	L_4: P(B_Y)
X = X + 1;	Y = Y + 1;
Y = Y – 1;	X = Y – 1;
V(B_X)	V(B_Y)
V(B_Y)	V(B_X)

What would be the maximum value of X and Y?

(a) X = 21 and Y = 20

(b) X = 20 and Y = 21

(c) X = 11 and Y = 11

(d) None of these

[MCQ]

9. Consider the following snippet for solution to critical section problem:

```
do{
    acquire lock
        critical section
    release lock
        remainder section
    }
    while(true);
```

Which of the following define acquire lock and release lock correctly?

(a) acquire () {   release () {

```
        while(! avail);        avail = true;
        avail = false;       }
    }
```

(b) acquire () {   release () {

```
        avail = false;            avail = true;
    }                         }
```

(c) acquire () {   release () {

```
        avail = true;             while(!avail)
    }                              avail = false;
                              }
```

(d) None of these

[MCQ]

10. Let P and Q be processes and let S and t be semaphores. Initially, both S and t are 1. The two processes execute the steps shown below. Show a sequence of steps that leads to deadlock

P:	**Q:**
Step P_1: Down (s)	Step Q_1: Down (s)
Step P_2: Down (t)	Step Q_2: Down (t)
Step P_3: CS – PA	Step Q_3: CS – QA
Step P_4: UP (s)	Step Q_4: UP(t)
Step P_5: Down (s)	Step Q_5: UP(s)
Step P_6: CS – PB	
Step P_7: UP(s)	
Step P_8: UP(t)	

(a) P_1, P_2, Q_1, Q_2

(b) P_1, P_2, P_3, P_4, Q_1, Q_2, P_5

(c) Q_1, Q_2, Q_3, P_1, P_2

(d) Q_1, Q_2, Q_3, Q_4, Q_5

Answer Key

1. (8)
2. (b)
3. (c)
4. (9)
5. (b)
6. (d)
7. (d)
8. (b)
9. (a)
10. (b)

Hints and Solutions

1. (8)

Initial value of semaphore = 12.

Total wait operation = 12P

Total signal operation = xV

Final Value = 8

So,

8 = 12 – 12P + xV

8 = 12 – 12 + xV

x = 8

2. (b)

The given code is modified Peterson's solution for critical section problem and it satisfies only mutual exclusion.

3. (c)

The definition of wait() and signal() operation performed on semaphore are as follows:

```
wait (S) {
    while(S < = 0);
    S – –;
}
signal (S) {
    S + +;
}
```

Therefore, option (c) is correct.

4. (9)

A = 12

P(A) = 11

V(A) = 12

P(A) = 11

V(A) = 12

V(A) = 13

P(A) = 12

P(A) = 11

P(A) = 10

P(A) = 9

So, the final value of 'A' is 9.

5. (b)

Monitors allows one process to execute in critical section at a time. It always satisfies the mutual exclusion properties.

Improper use of semaphore does not lead to avoidance of mutual exclusion.

Banker's algorithm is not related to mutual exclusion.

6. (d)

Producer	**Consumer**
down(mutex)	down(full)
down(empty)	down(mutex)
C.S	**C.S**
up(mutex)	up(mutex)
up(full)	up(empty)

Buffer full:

mutex = ~~1~~ 0 empty = ~~0~~ –1 ← producer suspended full = 8	mutex = 0 ← consumer suspended empty = –1 full = ~~8~~ 7
leads to deadlock	

7. (d)

The interchange will lead to deadlock.

8. **(b)**

I. X = 10, Y = 20

For the maximum value of X, run P_1 first

P_1: X = X + 1: 10 + 1 = 11

Preempt P_1

P_2: Y = Y + 1: 20 + 1 = 21

X = 21 – 1 = 20

$\therefore$ X = 20

Resume P_1

P_1: Y = Y – 1: 21 – 1 = 20

II. For the maximum value of Y

P_2: Read Y value [i.e. 20]

Preempt P_2

P_1: X = X + 1: 10 + 1 = 11

Y = Y – 1: 20 – 1 = 19

Resume P_2

P_2: Y + 1: 20 + 1= 21

Y = 21

$\therefore$ Y = 21

X = Y – 1: 21 – 1 = 20

Hence, the maximum value of X is 20 and Y is 21

9. **(a)**

The given code is solution to the critical section problem using mutex locks.

A mutex lock has a Boolean variable ‘avail’ whose value indicates if the lock is available or not. If the lock is available, a call to acquire() succeeds and the lock is then considered unavailable. A process that attempts to acquire an unavoidable lock is blocked until the lock is released.

The definition of acquire() and release is as follows:

```
acquire () {
    while(! avail);     /*busy wait*/
    avail = false;
}
release (){
avail = true;
}
```

10. **(b)**

The order of execution P_1, P_2, P_3, P_4, Q_1, Q_2, P_5 leads to deadlock.