Q.1) Max Marks: 1

If 'm' processes share 'n' resources of the same type, the maximum need of each process does not exceed 'n' and the sum of all their maximum needs is always less than 'm+n'. In this case:

A **Deadlock can never occur** Correct Option

Solution: (A)

Answer:A
Explanation:
To prevent deadlock(maximum resources need -1) will be allocated to all the processes and 1 resource will be given for all to complete individually and release all the resources so it will be = ∑(maximum need -1) +1 <=n (available resources)
= ∑max need -m +1 <=n
=∑max need +1 <= m+n
=∑max need <m+n--this is the condition for deadlock free it is satisfied so ans is A

B **Deadlock may occur**

C **Deadlock has to occur**

D **None of these**

Q.2) Max Marks: 1

There are three processes P1, P2 and P3 sharing a semaphore for synchronizing a variable. Initial value of semaphore is one. Assume that negative value of semaphore tells us how many processes are waiting in queue. Processes access the semaphore in following order :
(a) P2 needs to access
(b) P1 needs to access
(c) P3 needs to access
(d) P2 exits critical section
(e) P1 exits critical section
The final value of semaphore will be :

A **0** Correct Option

Solution: (A)

Answer: (A)
Explanation: Initial value of semaphores S=1

1. P2 needs to access decreases semaphore by 1, new value will be 0 (no one is waiting)
2. P1 needs to access decreases semaphore by 1, new value will be -1 (one process is waiting)
3. P3 needs to access decreases semaphore by 1, new value will be -2 (2 process are waiting)
4. P2 exits critical section increases semaphore by 1, new value will be -1 (one process is waiting)
5. P1 exits critical section increases semaphore by 1, new value will be 0 (no process is waiting)

So, option (A) is correct.

B **1**

C **-1**

D **-2**

Q.3) Max Marks: 1

Consider the following statements:
I. Starvation free imply bounded-waiting
II. Bounded waiting imply starvation free
Which of the following is correct?

A **Only I**

B **Only II**

C I and II

D None of these — Correct Option

Solution: (D)
Answer: D
Solution:
(I). Starvation and Deadlock both are w.r.t time and bounded waiting is w.r.t count of number of processes which can enter critical section .
So, what is starvation freedom ?
It says that process will sometimes enter CS and that time is finite, but is it bounded? Not always. (There is a difference between finite and bounded) . Waiting time may not be bounded, it has no limit .
Hence, first part Answer is No.
(II). Again, same thing, Bounded waiting does not says that one day process will enter CS . It only says that there is a limit within which it will enter . Starvation freedom says, that one day it will enter .
Instead, Progress + Bounded waiting --> Starvation freedom . How?
Consider a deadlock in a 2 process system . We can say progress is not satisfied . But, bounded waiting can still be satisfied in a 2 process system .
So, bounded waiting condition is not violated during a deadlock .
Hence, Progress + Bounded waiting --> Starvation freedom and bounded waiting alone doesn't ensure starvation free.

Q.4) Max Marks: 1

Consider the pseudo code given below and choose the correct option from the following.

```
        turn = 0;

P0 {
        while (turn != 0);
        /*** access shared data ***/
        turn = 1;
}

P1 {
        while (turn != 1);
        /*** access shared data ***/
        turn = 0;
}
```

A Satisfies Mutual Exclusion, Progress and Bounded Waiting.

B Satisfies only Progress and Bounded Waiting.

C Satisfies only progress

D None of these — Correct Option

Solution: (D)
Answer: None of these
Solution: : Strict alternation satisfies mutex – only the process whose turn it is (determined by the turn variable) can access the shared data. It does not satisfy progress, as the approach forces P0 to always run after P1 and vice versa. Should a process want to enter the critical section consecutively, it will have to wait, even if the other process is not interested in the critical section – violating progress. Also, the waiting time could be infinite. Finally, bounded waiting is satisfied because there is a bound on the number of other processes that are allowed to enter the critical section before this process

Q.5) Max Marks: 1

Suppose we want to synchronize two concurrent processes P and Q using binary semaphores S, T and U:

Process P:	Process Q:
P(S)	W:
P(T)	X:
P(U)	Y:
Print 'a'	Print 'a';
Print 'b'	Print 'b';
V(S)	A:
V(T)	B:
V(U)	C:

What should be written at W to avoid deadlock and what should be written at X and Y for this problem?

A W = P(T), X = P(U) and Y = P(S)

B W = P(U)

C W = P(S), X= P(T) and Y = P(U) — Correct Option

Solution: (C)
Solution:
S, T and U are binary semaphores shared between two concurrent processes P and Q. So, S, T and U can have a 0 or 1 as their initial value. If they take 0 as their initial value then, P cannot execute (because it performs down on all 3 semaphores), In this case for Process Q to execute W,X and Y have to be UP

operations in some order. (but it's not the case as all options are down operations). So, it means to answer this question Initially S=1,T=1 and U=1.

Process P performs DOWN on S,T,U in order. So if at the same time Q performs DOWN on these binary semaphores in some orders (remember 3∗2∗1=6 ways), Some of these 6 ways may result in Deadlock.

In process P, Down on S is performed. So if W is P(T) or P(U), i.e, down on T or U by Process Q Deadlock happens. (This eliminates 4 ways) and Confirms that W has to be P(S).

And out of remaining 2−ways, one is **W,X,Y = P(S),P(T),P(U)** which is sure shot answer for X and Y respectively. And, it's pretty interesting to trace it out that remaining case **W,X,Y = P(S),P(U),P(T)** is also deadlock free.

D **Not possible to avoid deadlock**

Q.6) Max Marks: 1

Each process P_i , i = 1 to 9 executes the following code :

```
while (TRUE)
{
     P(mutex);
    Critical section ;
     V(mutex);
}
```

The process P_{10} executes the following code :

```
while(TRUE)
{
     V(mutex);
     critical section ;
     P(mutex);
}
```

Initial value of binary semaphore "mutex" = 1.Then what is the maximum number of processes that may be present in the critical section at any instant of time?

A **2**

B **3** Correct Option

Solution: (B)

Solution:

Initially Binary semaphore mutex=1, and we know that a binary semaphore can have only two states 0 (or) 1. So, even if we perform an UP on semaphore s=1, no advantage here. So, the best way here would be proceeding with an alternating sequence of 0 and 1

```
// Code for process 0-9          //Code for process 10
while (TRUE) {                   while(TRUE) {
        P(mutex);                        V(mutex);
        <Critical section>               <Critical section>
        V(mutex);                        P(mutex);
}                                }
```

Suppose We start with process P1, means P1 enters Critical section by performing a DOWN on semaphore mutex. At this point mutex=0, Now P10 executes and performs an UP on mutex and enters Critical section. At this point mutex=1, Now, Any one process out of P2…P9 enters critical section by performing a DOWN on binary semaphore mutex.Finally, mutex=0 and there is no way we could perform an UP on binary semaphore without allowing any process to leave Critical Section.

So, Maximum number of processes which are allowed to enter critical section = 3 (two out of P1…P9 and P10)

C **9**

D **10**

Q.7) Max Marks: 1

Consider a Counting semaphore value as 20 currently, later 11 down operations and X up operations were completed on this semaphore. If the new value of semaphore is 41, Find the value of X _____

Correct Answer

Solution: (32)

Answer: 32

Solution: from the given data we can write it as

$20 - 11 + X = 41 \Rightarrow X = 41 - 9 = 32$

Q.8) Max Marks: 1

Which of the following statement is TRUE with respect to semaphore?

A. synchronize critical resources to prevent deadlock

B. synchronize critical resources to prevent contention — Correct Option

Solution: (B)
Answer: B
Solution:
A semaphore is a way to lock a resource, so that it is guaranteed that while a piece of code has access to that resource, the semaphore keeps two threads from concurrently accessing the resource.

C. are used to do I/O

D. are used for memory management

Q.9) Max Marks: 1

A	B
down(S)	down(Q)
down(Q)	down(S)
print("applied")	print(" course")
up(Q)	up(Q)
up(S)	up(S)

Where S & Q are two semaphores initialized to 1. Which of the following are the possible out comes.

(a) Deadlock may occur
(b) prints "Applied Course"
(c) Deadlock never occurs

A. Only a — Correct Option

Solution: (A)

Answer : Only a.

Solution: Suppose consider a sequence where all the instructions of A will run and then all the instructions of B => prints '"applied course". It doesn't print "Applied Course".

Now, consider a case where instructions are executed as follows
A(down(S)), B(down(Q)), A(down(Q)), B(down(S))
⇒ S = 0 | Q=0 | Process A is in waiting queue| Process B also in waiting|

B. Only b

C. Only b and c

D. Only a and b

Q.10) Max Marks: 1

Consider the methods used by processes P1 and P2 for accessing their critical sections whenever needed, as given below. The initial values of shared boolean variables S1 and S2 are randomly assigned.

Method used by P1	Method used by P2
while (S1==S2);	while (S1I=S2);
Critical Section	Critical Section
S1=S2;	S2 = not(S1);

Statement I : The above two processes are both deadlock free
Statement II : The above two processes are both starvation free.

Which of the following is correct?

A **Statement I is True**

B **Statement II is True**

C **Both I and II are True** — Correct Option

Solution: (C)
Solution:
This sequence of statements is both Deadlock free and starvation free.

In strict alternation, bounded waiting ensures *starvation freedom* because a process will then eventually gets to execute CS.

D **Both I and II are False**

Q.11) Max Marks: 2

Consider the following code to solve the critical section problem for two processes P0 and P1. Initially flag[i] contains false for i = 0 and 1

```
while(1)
{
   flag[i] = true;
   while(flag[j]);
   <critical section>
   flag[i] = false;
   <remainder section>
}
```

Assume i refers to the current process Pi and j refers to the other process Pj. If two processes executing above code concurrently then which of the following does not satisfy the above solution?

A **Mutual exclusion and progress**

B **Mutual exclusion and bounded wait**

C **Progress** — Correct Option

Solution: (C)
Answer: C
Solution:
Mutual exclusion and Bounding waiting is satisfied here, progress not.
Mutual exclusion because no two processes can enter critical section at the same time. If one process is in the critical section, then other has to wait.
Bounding waiting because when a process shows the interest to go into the critical section then there is bound on the number of time other processes can go into the critical section and that is 1. (Process is already in the critical section).
Progress is not satisfied here because when both the process execute Flag[i] = true and got preempted then both the processes waiting for each other to release the flag. Which is not going to happen without entering the critical section. So They will wait for indefinite time, No progress. This situation is called "Deadlock".

D **None of the above**

Q.12) Max Marks: 2

Consider the following 6 concurrent processes where semaphore X initialized to zero.

A: down(X); cs; up (X);
B: down(X); cs; up (X);
C: down (X); cs; up(X);
D: up(X); cs; down(X);
E: up(X); cs; down(X);
F: up(X); cs;down(X);

What is the possible Maximum value of X is ?

A **1**

B **3** — Correct Option

Solution: (B)
Answer: 3
Solution: Processes D, E and F can do signal(Up) operation without any intermediate down operation and thus S can go up to value 3.

C **4**

D **6**

Q.13) Max Marks: 2

Consider the following synchronization construct used by processes P1, P2 and P3. The S1, S2 and S3 are counting semaphore variables:

S1 = 3, S2 = 2 and S3 = 1;

P(S1);
P(S2);
P(S3);

Critical Section;

V(S3);
V(S2);
V(S1);

Which of the below statement is true?

A. It satisfies mutual exclusion and progress but not bounded waiting.

B. It satisfies both progress and bounded waiting but not mutual exclusion.

C. It satisfies mutual exclusion and bounded waiting but not progress.

D. It satisfies all the mutual exclusion, progress and bounded waiting. **Correct Option**

Solution: (D)
Answer: D
Solution: It satisfies all the ME, progress and bounded waiting because the order of counting semaphore down operations are accordingly performed.

Q.14) Max Marks: 2

The Dining Philosophers problem with 5 Philosophers has the solution for synchronization without deadlock. Which of the following is correct solution?

A. 4 Philosophers take the left fork and later right fork. Remaining one Philosopher take right fork first then left fork.

B. Any two Philosophers taking left fork first and later right fork. Remaining three Philosopher take right fork first then left fork.

C. Both (a) and (b) are correct solutions. **Correct Option**

Solution: (C)
Answer: C
Solution: To avoid deadlock in Dining Philosophers Problem there are two simple solutions.
Consider there are N philosophers:
(N-1) Philosophers First Take Left Fork then Right Fork, Last Philosopher takes Right and then Left.
All philosopher sitting in ODD position take Left Fork then Right Fork. All philosophers sitting in EVEN position take right fork first then Left Fork .

Assuming option B is telling the point number 2. Hence both should be correct

D. Neither (A) nor (B)

Q.15) Max Marks: 2

A computer has six tape drives with n processes competing for them. Each process may need two drives. What is the maximum value of n for the system to be deadlock free?

A. 5 **Correct Option**

Solution: (A)

Answer: A
Explanation:
n processes each requiring 2 tape drives
total no of tape drives = 6

Process	Process1	Process2	Process3	Process4	Process5	Process6
No. of Tape Drives	1	1	1	1	1	1

This is a deadlock Situation. So anything less than 6 process will be free from Deadlock because at least 1 tape drive will be in hand which can be used for completion of a running process.
Here asked for maximum process with deadlock free Hence 5 only.

B. 4

C. 3

D. 6

close