# Branch : CSE & IT Batch : Hinglish

# Operating Systems

## Process Synchronization/Coordination

DPP 07

**[MCQ]**

**1.** What is a compulsory step before using semaphore?

(a) Deciding final value of semaphore
(b) Initialization of semaphore
(c) Defining number of operations to be performed
(d) All of the above.

**[MCQ]**

**2.** Consider the following statements, which of the following is correct?

(a) Semaphore are atomic in nature and implemented in user's mode.
(b) Semaphore are atomic in nature and implemented in kernel mode.
(c) Semaphore are non-atomic in nature and implemented in user's mode.
(d) Semaphore are non-atomic in nature and implemented in kernel mode.

**[MCQ]**

**3.** If a semaphore's value is "– 3", then what does magnitude of "– ve" value indicate _____?

(a) Number of successful up operation.
(b) Number of successful down operation.
(c) Number of blocked processes.
(d) Number of unblocked processes.

**[MCQ]**

**4.** Processes $x_1$ and $x_2$ uses flag_critical in the following function to achieve mutual exclusion. Assuming flag_critical is initialized FALSE initially.

```
get_access
{
if (flag_critical = = FALSE)
      {
      flag_critical = TRUE;
      Critical_section();
      flag_critical = FALSE;
      }
}
```

Consider the following statement:

(i) The above routine may lead to deadlock.
(ii) It is possible for processes $x_1$ and $x_2$ to access critical section concurrently.

(a) (i) is true and (ii) is false.
(b) (ii) is true (i) is false.
(c) Both (i) and (ii) are true.
(d) Both (i) and (ii) are false.

**[MCQ]**

**5.** Consider the code given below, used by the processes $x_1$ and $x_2$ to access critical section.

The initial value of shared Boolean variable P and Q are false

| $x_1$ | $x_2$ |
|---|---|
| while (P = = Q); | while (P! = Q); |
| <critical section> | <critical section> |
| P = !(Q); | P = Q; |

Select the true statements from the following:

(i) Process $x_2$ can go into critical section just after one entry by process $x_1$ into its critical section.
(ii) Mutual exclusion is not ensured.
(iii) Process $x_1$ can go into critical section many times without single entry of $x_2$ into its critical section.
(iv) None of the above
(a) (i) & (ii)
(b) (ii) & (iii)
(c) Only (i)
(d) (i), (ii), & (iii)

**[NAT]**

**6.** Consider the two function $P_i$ and $P_j$ that share a variable Q with an initial value '3' execute concurrently:

| $P_i()$ | $P_j()$ |
|---|---|
| { | { |
| R = Q * 2: | S = Q + 1: |
| Q = R; | Q = S; |
| } | } |

What are the different possible value for variable Q at the end of execution of both process $P_i$ and $P_j$?

**[MCQ]**

**7.** Match the following statements

| List I | List II |
|---|---|
| A. Critical section | 1. Ensuring that only one process can execute C.S. |
| B. Synchronization | 2. atomic operation are used to ensure co-operation between processes. |
| C. Mutual exclusion | 3. Section of code that only one process can access at once. |

**Matches:**

| | A | B | C |
|---|---|---|---|
| (a) | 1 | 2 | 3 |
| (b) | 3 | 2 | 1 |
| (c) | 2 | 3 | 1 |
| (d) | 1 | 3 | 2 |

**[NAT]**

**8.** Let S be a binary semaphore variable. Let S = 1 initially.
Assume that no blocked processes exist in the system.
The following operations are performed on semaphore S.
6 P, 8 V, 12 P, 11 V, 19 P
The number of blocked processes after executing these operations are _____.

# Answer Key

1. **(b)**
2. **(b)**
3. **(c)**
4. **(b)**
5. **(c)**
6. **(4)**
7. **(b)**
8. **(19)**

# Hints & Solutions

**1. (b)**

Semaphore initialization is compulsory step before using it. Without initializing the semaphore's value it is not possible to perform further operations on it.

**2. (b)**

Semaphore are atomic and are implemented in system's kernel. The semaphore values are kept in a table stored in kernel memory. A semaphore is identified by a number corresponding to a position is this table.

**3. (c)**

Magnitude of "–ve" value indicates the number of blocked processes.

**4. (b)**

(ii) is true because, both the processes $x_1$ and $x_2$ can access critical region concurrently because of

if(flag_critical = = FALSE)

$x_1$, $x_2$ can execute the above condition simultaneously and can enter C.S without leading to deadlock.

**5. (c)**

Process $x_1$ cannot go into critical section multiple times without entry of $x_2$.

**6. (4)**

**I.** It is given the process $P_i()$ and process $P_j()$ is executing concurrently. So, assign the unique number to operations of both the processes.

$P_i()$: $I_1 \Rightarrow R = Q * 2$; $I_2 \Rightarrow Q = R$

$P_j()$: $I_3 \Rightarrow R = Q * 2$; $I_4 \Rightarrow Q = R$

**II.** Now, perform the operation to find the distinct values of Q

1. $I_l$, $I_2$, $I_3$, $l_4 = 7$
2. $I_1$, $I_3$, $I_2$, $I_4 = 4$
3. $I_3$, $I_1$, $I_4$, $I_2 = 6$
4. $I_3$, $I_4$, $I_1$, $I_2 = 8$
5. $I_3$, $I_1$, $I_2$, $I_4 = 4$
6. $I_1$, $I_3$, $I_4$, $I_2 = 6$

Therefore, we have total 4 distinct values of Q: {7, 4, 6, 8}

**7. (b)**

- Synchronization uses p() and v() operation.
- Critical section is a section of code that only one process can access at a time.
- Mutual exclusion ensures that only one process can execute CS at any time.

**8. (19)**

Initially S = 1

6 P: 5 blocked processes

8 V: S = 1 & 0 blocked process

(As all six blocked processes will be resumed and $7^{th}$ V will make S = 1 and $7^{th}$ V will continue to keep S = 1)

12 P: 11 blocked processes

11V : S = 0 and 0 blocked process

19 P : S = 0 and 19 blocked processes