Assignment-4
Operating Systems

Date / /

Name - Sahil Bhardwaj
Roll - 2301010425

Part-A

Q1

ans A race condition occurs when two people try to use the same shared resource at the same time, causing an incorrect or unexpected result. For example - if two people try to take ₹500 from the same wallet simultaneously, each thinks the money is still there and both take it, even though the wallet only had ₹500 total. Mutual exclusion prevents this by ensuring that only one person can access the wallet at a time, forcing the second person to wait, which guarantees a safe & correct outcome

Q2

ans Peterson's Solution vs Semaphores -

- Implementation Complexity:

Peterson's Solution - More complex to implement correctly because it requires careful use of shared variables and is limited to two processes

Semaphores - Simpler and cleaner to use in programs; support many processes and offer standard operations (wait, signal).

- Hardware Dependency:

Peterson's Solution - Works only under strict assumptions and fails on modern CPUs due to instruction reordering, not hardware-independent in practice.

Semaphores - Require minimal hardware support such as atomic instructions (e.g. test-and-set), make them reliable on modern systems.

Date / /

Q3

ans Monitors automatically handle mutual exclusion so in a multi-core system they reduce the chance of programming errors by ensuring that only one thread can execute the critical section at a time without manually managing semaphores

Q4

ans In the Reader-Writer problem, starvation happens when one type of process (usually writers) keeps getting delayed because the other type (readers) continuously arrives and is always given priority. As a result, a writer may wait indefinitely while readers keep accessing the shared resource.

One method to prevent it -

Use a fair (FIFO) scheduling policy, where readers and writers are queued in the order they arrive. This ensures that each process eventually gets its turn and no reader or writer waits forever.

Q5

ans Eliminating Hold and Wait forces a process to request all needed resources at once before it starts. Drawback: This leads to poor resource utilization, because a process may hold resources it doesn't need yet, keeping them unavailable for others & reducing overall system efficiency.

Date / /

Part-B

Q6
ans Given: $S_1 \Rightarrow P_1 \to P_2, P_3 \to P_4$
$S_2 \Rightarrow P_2 \to P_5, P_5 \to P_6$
$S_3 \Rightarrow P_6 \to P_1$

(a) Global wait-for Graph:
$P_1 \to P_2 \to P_5 \to P_6 \to P_1$
$P_3 \to P_4$

(b) Deadlock Detection:
Cycle exists: $P_1 \to P_2 \to P_5 \to P_6 \to P_1$
Deadlock Processes: P_1, P_2, P_5, P_6

(c) Distributed Detection:
Chandy-Misra-Hass Algorithm for distributed deadlock detection

Q7 Given: local access: 5ms
remote access: 25 ms
remote probability: 0.3

(a) Expected file access time E[T]:
$E[T] = (0.7 \times 5) + (0.3 \times 25)$
$= 3.5 + 7.5 = 11\text{ms}$

(b) Caching strategy
→ Client side caching - store frequently accessed remote file locally

→ justification - reduces repeated remote access latency & network load.

Date / /

Q8
ans Given : Full : 200ms
Incremental : 50 ms
RPO : 1s

(a) Optimal Mix - perform full checkpoint every 1s, followed by incremental checkpoints every 250 ms.

(b) Reasoning - Incremental checkpoints are faster, reducing overhead, full checkpoints ensure complete recovery - Combination meets RPO without blocking the system.

Q9 ans (a) Distributed Scheduling Challenges - Flash sales create sudden load spikes, uneven across regions.
Suitable Algorithm - Weighted Round Robin or Dynamic Load Balancing using least-loaded server.

(b) Fault Tolerance Strategy -
Geo-Redundant Deployment replicate services across multiple data entries.

RTO/RPO - Use synchronous replication for critical services (low RPO) and asynchronous replication for less critical services (acceptable RTO).

Result: High availability even if a region fails.