Name → Ayushi kumari
Roll no. → 2301410035
Course → B. tech cse (cyber security)

Assignment 4. Operating Systems

PART-B

1.) Race condition → Imagine two people writing on the same whiteboard at the same time. If Person A erases a part while Person B is writing there, the final result becomes unpredictable. This is a race condition b/w their actions.

Mutual Exclusion Fix:

only one person enters the "whiteboard room" at a time (lock the door). This ensures that while on is writing, the other must wait → preventing conflicts.

2.)

Peterson's Solution	Semaphores
• Requires careful algorithmic logic using two shared variables (turn + flag); tricky but doable in software.	• simple operation like wait() and signal().
• Hardware dependency, purely software no special hardware needed	• Needs atomic hardware instructions (e.g. test-and-set) for safe implotation.

3.) Advantages → Monitors provide implicit mutual exclusion - lock acquisition and release are handled automatically. In a multi-core system, this reduces programming errors and improves thread-safety, since every method in the monitor is executed by only one thread at a time.

4.) Starvation → If writes are waiting but new readers keep arriving, readers may continuously get priority → write waits forever.

Prevention:- use write priority: once a writer is waiting, no new readers is allowed to enter. This ensures writers eventually get the lock.

5.) To eliminate Hold-and-wait, a process must request all resources at once, before execution.

Drawback → This leads to low resource utilization.

Part B

6.) Given fragments:-

- S1: P1 → P_2, P3 → P4
- S2: P2 → P5, P5 → P6
- S3: P6 → P1

a) Global wait-for Graph

Combine all:-

P1 → P_2 P3 → P4 (Independent chain)

P2 → P5

P5 → P6

P6 → P_1

b.) Yes, there is a deadlock.
cycle P1 → P2 → P5 → P6 → P1

c) ~~[illegible]~~ Chandy - Misra - Hass algorithm for distributed deadlock detection.

7.) Given • Local access = 5ms
• Remote access = 25ms
• P(remote) = 0.3 → P(local) = 0.7

a.) $E = (0.7 \times 5) + (0.3 \times 25)$
$E = 3.5 + 7.5 = 11ms$

b.) Client - side caching with Least Recently Used (LRU). Because, most frequently accessed files stay local and reduces remote accesses, easy to implement.

8.) • Full checkpoint = 200ms.
• Incremental = 50 ms
• Must have a checkpoint ≤ 1 sec old at any time.

a.) Do 1 full checkpoint every 4 seconds + incremental every one second.
Ex schedule in (sec)

0s – full
1s → Incremental
2s, 3 s → Incremental
4s → Full
5s, 6s, 7s → Incremental
8 s → Full
9s → Incremental
10s → end.

b.) • Incremental checkpoints are cheap.
• Full checkpoints reduces dependency chain.
• Ensures RPO ≤ 1s.

9.) a) Challengs:-

- Sudden huge traffic spikes
- Uneven regional load
- Hotspots on specific microservices (cart, ~~checkpoint~~) checkout)
- Maintaing low latency

Suitable algo:-

→ Dynamic load Balancing using the work-Stealing Algo.

b.) Use Geo-Redundent Replication and Failover:-

- active-Active multi-region deployment.
- Data replicated via synchronous replication for low ~~RB~~ RPO
- Automated failover via health checks to ensure low RTO.

why?

→ If one region fails, traffic immediately shifts to healthy region.

→ Ensures continous service availability and minimizes data loss.