Aman Yadav
2301410010
BTech Cybersecurity
O.S.

DATE: __/__/____
PAGE: ______

Assignment - 4

Sol. 1 A race condition occurs when two or more processes try to access or change the same shared resource at the same time, leading to incorrect or unpredic-table results.

- Real-world example:
Imagine 2 people withdrawing money from the same bank account at the same time:
→ Both see the balance as ₹1000
→ Both try to withdraw ₹1000
→ Because their actions overlap, the final balance becomes wrong.
- Mutual exclusion (mutex) ensures that only one person/process uses the shared resource at a time.
In bank example, a mutex would lock the account so:
1. Person A withdraws first.
2. After A finishes, the lock is released.
3. Person B can then withdraw.

Sol. 2 Peterson's Solution is a software method that works only for 2 processes and uses busy waiting. It is simple but depends on hardware behavior, so it may not work on modern systems.
Semaphores are OS-supported variables that can control many processes and avoid busy waiting by blocking processes. They are more flexible and reliable.

Sol. 3 The producer-consumer problem can use either semaphores or monitors for synchronization.

Monitors automatically allow only one thread at a time so they make the producer - consumer problem safer and easier compared to semaphores.

Sol.4 Starvation happens when a reader or writer keep waiting for too long because others always get priority.

- Prevention → Use FIFO order so every reader & writer gets a fair turn.

Sol.5 Removing Hold and wait prevents deadlock, but the drawback is that processes must request all resources at once, which can waste resources and reduce system performance.

Part-B

Sol. 6(a) S1 : $P_1 \rightarrow P_2$, $P_3 \rightarrow P_4$
S2 : $P_2 \rightarrow P5$, $P_5 \rightarrow P_6$
S3 : $P_6 \rightarrow P_1$

Global wait for graph
$\rightarrow$ Combine all edges

$P_1 \rightarrow P_2$
$P_2 \rightarrow P5$
$P_5 \rightarrow P_6$
$P_6 \rightarrow P_1$

but $P_3 \rightarrow P_4$ are separate

$P_1 \rightarrow P_2 \rightarrow P_5 \rightarrow P_6 \rightarrow P_1$ plus $P_3 \rightarrow P_4$

(b) Yes the deadlock exists. P_1, P_2, P_5, P_6 are involved in the deadlock but P_3 & P_4 are not part of it.

(c) We can use centralised collector - every site sends its local wait for edges to a central detector that builds the global wait for graph & runs cycle detection (DFS). It is simple to implement.

Sol. 7 (a) Expected file access time

$E = (1-p) \times T_{local} + p \times T_{remote}$
$E = 0.7 \times 5 + 0.3 \times 25$
$= 3.5 + 7.5$
$= 11$ ms

(b) Caching strategy. Client-side read cache with LRU eviction & validation. Caching reduces the fraction of remote access so typical access time approaches the local time. LRU keeps hot items & validation maintains consistency with acceptable overhead.

Sol. 8 (a) Incremental checkpoint every 1s; Full Checkpoint every 10s.

Overhead (10s):

$1 \times 200 + 10 \times 50 = 200 + 500 = 700\,ms$

(b) Incrementals meet the 1s RPO with low cost; a full every 10s bounds the incremental chain while keeping total overhead small.

Sol. 9 (a) Challenges: Sudden workload spikes, uneven load accross regions, high latency for cross-region requests & session locality issues.

Algorithm: Global load balancer + consistent hashing + autoscaling to distribute traffic based on real-time load while keeping user sessions local.

(b) Use active-active multiregion deployment so service continues even if a datacenter fails.

Critical data replicated synchronously for low RPO; automatic failover + health checks ensure fast recovery & low RTO.