Name :- Adhway Banerjee
Roll no :- 2301410002
Course :- BTech CSE Cybersecurity
Course Name :- Operating System

Assignment No. 4

Q1→ A race condition happens when two or more people try to do the same task at the same time and the final result becomes unpredictable because their actions overlap.

~~Rat~~ Real World :-
Imagine two people are withdrawing money from the same account at the exact same time using Different ATMs. Both machines check the balance and allow to withdraw ₹1000. The account ends up incorrect balance because the actions happened simultaneously.

How mutual exclusion fixes it :-
Mutual exclusion is only person can access the shared resources at a time. In this example, the bank system would lock the account when first ATM starts the withdrawal process. The second ATM must wait till the lock is released. This prevents overlapping action and keeps balance correct.

Q2→ Peterson's Solution :- Is a software-only algorithm. Simple in concept but tricky to implement correctly, works only for 2 processes and also depends on strict assumptions like atomic read/write operation which may not hold on morden hardware.

Samaphores :- are easier to use at the programming level as they provide built-in operation like 'wait' and 'signal'. They rely on hardware Support for atomic instruction to avoid race conditions internally. Implementation is more complex inside the OS but Simpler for the programmer.

Q3→ using Monitors gives better structure and Safety because Synchronization is built directly into the programming construct.
In a multi-core system, monitors reduce the chance of programmer errors since they automatically handle locking and condition signalling making the producer - consumer process easier and safer to manage compared to mannually using Semaphores.

Q4→ Starvation occours when one group keeps waiting forever because continous incoming readers always get priority. As a result, the writter never gets access to the Shared data.
To prevent this a fair Scheduling method can be used:- giving writters priority once they start writing. This ensures that after current readers finishes, the writer gets access before new readers are allowed in.

Q5→ Eliminating Hold and wait means a process must request all required resources at the start.
A practical drawback is poor resource utilization. Process may hold resources they dont need immediatly causing other processes to wait longer and reducing over all system efficiency.

Q6→

a) Global wait for graphs edges :-

$P1 \rightarrow P_2, P_2 \rightarrow P_5, P_5 \rightarrow P6, P6 \rightarrow P_1$, and $P_3 \rightarrow P4$

b) Yes, there is a dead lock. Cycle :-

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

Process involved $P_1, P_2, P_5, P6$ (P_3 and P_4 are not a part of this cycle)

c) Apply the Chandy - Misra - Haas (probe/edge chasing) algorithms a site with a blocked process sends probe along out going wait for edges; if a probe returns to its orignin a cycle is detected.

Q7→

a) Expected access time
= (0.7 × 5ms) + (0.3 × 25ms)
= 3.5ms + 7.5ms
= 11ms.

b) Using client side caching improves performance because frequently accessed files can be served locally reducing expensive remote accesses and lowering the overall average access time.

Q8→

a) An efficient mix is: One full checkpoint every 10 sec + incremental checkpoints every 1 second.
So in 10 second: 1 full + 9 incremental

b) A full checkpoint gives a complete recovery base while incremental checkpoint ensure the system never loses more than 1 sec of updates satisfying the PRO
This combination keeps overhead low because incremental checkpoints are much cheaper than running frequent full checkpoints.

Q1-> Case Study :-

a) During flash sales the biggest distributed scheduling challenges are sudden overload, uneven traffic accross regions, and keeping response times low while coordinating many servers. A suitable load-balancing approach is the dynamic load balancing algorithm such as ~~last~~ least-Load first which is continuously checks server load and redirects request to the least busy server. This maintain stable performance during traffic spikes -

b) A strong fault-tolerance strategy is georeplication with failover. Data is replicated across multiple regions using asynchronous + periodic synchronous updates. to keep the RPO ~~too~~ low. If one region fails, traffic is automatically shifted to another replica using a hot standby setup which minimizes RTO because services can restart instantly with updated data. This ensures the platform stays available even if an entire ~~at~~ data center goes down.