Yadav Amit

2301010218

Date __/__/__

~~508~~/508918

Saathi

Operating System Assignment-3

1. Race Condition + Real-World Example + Mutual Exclusion

Ans A race condition occurs when two or more entities try to perform actions simultaneously, and the final outcome depends on the unpredictable timing of their operations.

Real-world example (outside computing):

• Two people withdrawing money from the same ATM account at the same time.

How mutual exclusion solve this?

A lock at the ATM ensures only one person can access the account at a time.

2\.

Ans

Aspect	Peterson's Solⁿ	Semaphores
Implementation complexity	Simple logic but only works for 2 process; difficult to extend	Very flexible; suppor many process, used widely in OS kernel
Hardware Dependency	Pure software, requires strict hardware support	Needs Atomic hardware instruc-tions.

Conclusion: Semaphores are more practical for real OS implementation.

3\.

Ans Monitors automatically provide mutual exclusion, so only one thread can execute critical sections at a time.

Page No.

Date ___/___/______

Advantage:
In multi-core systems, monitors simplify synchronization by eliminating the need for manually controlling semaphores, reducing errors.

4.
Ans How starvation occurs:
If writers are repeatedly waiting but readers keep arriving, the writer may never get access (reader priority → writer starvation).

Prevention Method:
Use writer priority: Once writer arrives, block new readers and allow current readers to finish → writer executes next.

5.
Ans Eliminating Hold and Wait means a process must request all required resources at once before running.
Practical Drawback:
This leads to low resource utilization. Many processes unnecessarily hold resources they don't immediately need, causing delays.

Part B

Q6.
Ans Given Total = (10, 5, 7)

Date ___/___/___

Process	Allocation (A, B, C)	Max (A, B, C)
P_0	0, 1, 0	7, 5, 3
P_1	2, 0, 0	3, 2, 2
P_2	3, 0, 2	9, 0, 2
P_3	2, 1, 1	4, 2, 2
P_4	0, 0, 2	5, 3, 3

(a) Need = Max − Allocation

Process	Need (A, B, C)
P_0	7, 4, 3
P_1	1, 2, 2
P_2	6, 0, 0
P_3	2, 1, 1
P_4	5, 3, 1

Available = Total − Allocation Sum
Allocation Sum = (7, 2, 6) → Available = (3, 3, 1

(b) Safe State check:
Work = (3, 3, 1)

- P_1 can run (Need ≤ Work) → New York = (5, 3, 1)
- P_3 can run → work = (7, 4, 2)
- P_4 can run → work = (7, 4, 4)
- P_0 can run → work - (7, 5, 4)
- P_2 can run → work = (10, 5, 6)

→ Safe sequence = $P_1 \rightarrow P_3 \rightarrow P_4 \rightarrow P_0 \rightarrow P_2$

(c) If P_1 requests (1, 0, 2)
New Need = (0, 2, 0) Request ≤ Available (1, 0, 2) ≤ (3, 3, 1)
cannot be granted immediately.

Page No.

Saathi

Date ___/___/___

7. Dinning Philophers:

→ Deadlock scenario:
each philospher picks up their left chopstick first, all hold one chopstick, and none can pick the right one → deadlock.

Soln: Use semaphore array chopstick [5] = 1 and a mutex for limit

```
wait ( mutex);
wait ( chopstick [i]);
wait ( chopstick [(i+1) % 5);
 eat ();
signal ( chopstick (i]);
signal ( chopstick [(i+1) % 5]);
signal ( mutex);
```

8. I/O system Analysis:
Given:
Interupt time = 5 μs
Transfer Rate = 500 KB/s = 500 × 1024 = 51200 bytes/s
Block = 100 bytes

(a) Interrupts per second = 51200/100 = 5120
CPU Time = 5120 × 5 μs = 25.6 ms per second

(b) Improvement: Use Direct memory access (DMA) to transfer data directly without frequent CPU interupts

Saathi

Date __ / __ / __

9.

Ans (a) Critical Sections :

- Updating shared readar data
- flight Path computation database
- Communication logs

IPC mechanism
Use message queues or real time semaphores for synchronizing ensuring quick suiting and real-time safety.

(b) Deadlock detection & Recovery :

- Use resource allocation graph for detection.
- On ~~[illegible]~~ detection, promt non-critical process (eg: temporarily pause flight path computation).

[illegible]
21/11/25