ABHINAV JHA
2301010211
B.TECH CSE CORE - 4

Assignment - ~~4~~3

Part - A

1. Race condition : Two people try to write different notes on the same sticky note at the same time. If both read current text, append their own & write back without coordination, one person's update can overwrite other's - result depends on timing

2. Implementation complexity: Peterson's solution is simple algorithmically ~~for~~ but limited to two processes and is tricky to extend. Semaphores provides a general high level abstraction suitable for many process.
Hardware dependency: Peterson's relies on sequential consistency and atomic writes to shared variables - it may fail on modern weak memory multiprocessors without memory fences.

3. Monitors provide structured mutual exclusion plus condition variables built in. In a multi-core system, monitors can use efficient per-monitor locks & condition queues, reducing context switch churn & avoiding busy-waiting.

4. Writers starve if readers continously arrive because now readers can keep entering and block waiting writers forever. We can prevent it by using a fair policy so that once a writer arrives it eventually gets priority and waits readers are blocked until writer completes.

5. Eliminating Hold and Wait typically forces process to acquire all required resources at once ~~for~~ (or release held resource before requesting new ones). Practical drawback: poor resource utilization and reduced concurrency process may

hold many ~~any~~ resources while idle, leading to low throughput and possible starvation increased complexity to predict all future needs.

Part - B

6.a) Need matrix = Max Matrix - Allocation Matrix

Process	A	B	C
P_0	7-0=7	5-1=4	3-0=3
P_1	3-2=1	2-0=2	2-0=2
P_2	9-3=6	0-0=0	2-2=0
P_3	4-2=2	2-1=1	2-1=1
P_4	5-0=5	3-0=3	3-2=1

Need matrix - $\begin{bmatrix} 7 & 4 & 3 \\ 1 & 2 & 2 \\ 6 & 0 & 0 \\ 2 & 1 & 1 \\ 5 & 3 & 1 \end{bmatrix}$

b) Safe State Check

Step	Process Chosen	Available Before	Need	Available After
1	P_1	[3, 3, 2]	[1,2,2]	[5,3,2]
2	P_3	[5, 3, 2]	[2,1,1]	[7,4,3]
3	P_0	[7, 4, 3]	[7,4,3]	[7,5,3]
4	P_2	[7, 5, 3]	[6,0,0]	[10,5,5]
5	P_4	[10, 5, 5]	[5,3,1]	[10,5,7]

Safe Sequence : P_1, P_3, P_0, P_2, P_4
System is in a Safe State

c) Request from P_1 : $(1,0,2)$
Request $\le$ Need ?
$[1,0,2] \le [1,2,2]$ = yes
Request $\le$ Available ?
$[1,0,2] \le [3,3,2]$ = yes

Request can be granted safely.

7. All 5 philosophers pick their fork simultaneously → each holds one fork and waits for right fork → circular wait → deadlock.

Semaphore solution: semaphore fork initialized to 1; each philosopher : wait (forks [left]); eat; signal (fork [right]); signal [forks [left]) . Deadlock avoidance: impose an ordering or allow at most 4 philosophers to try picking forks concurrently.

8. a) Interrupts ~~per second~~ = 512.00
CPU time spent handling per second = 0.025600 sec/sec
= 2.5600 % of CPU

[illegible] 20/11/28

b) Use larger block sizes or use DMA (Direct Memory Access) so CPU is integrated less frequently with DMA CPU overhead for transfers is minimal while keeping same throughout.

9 a) Critical sections: shared radar buffer / queue where raw radar frames are written shared flight - state database updated by flight path calculator; communication channel logging and command queue to pilots.

b) Deadlock detection and recovery. Detection: Periodically run a deadlock detection algorithm on resource - graph. Recovery: - preempt low - priority or less - critical process rollback process holding resources to take over critical tasks quickly.