VIT
Vellore Institute of Technology

Slot:D1+TD1+D2+TD2

School of Information Technology and Engineering

Fall Semester 2023-2024 (2021 B.Tech Students)

Continuous Assessment Test – II

Programme Name & Branch : B.Tech-IT

Course Name & Code: Operating Systems & BITE303L

Class Number (s): VL2023240100136, VL2023240100115, VL2023240100535, VL2023240100121, VL2023240100119, VL2023240100132, VL2023240100173, VL2023240100117, VL2023240100130

Faculty Name (s): KISHORERAJA P C, KRISHNAMOORTHY N, RADHIKA C, SENTHILKUMAR T, ARUL TREESA MATHEW, ARUNKUMAR M, RAJKUMAR M, THANGA PRASATH S

Exam Duration: 90 Min. **Maximum Marks: 50**

General instruction(s):

Q.No.	Questions
1.	You are asked to analyze the performance of a newly designed processor over 4 concurrent processes P1 to P5 in the below given scenario,

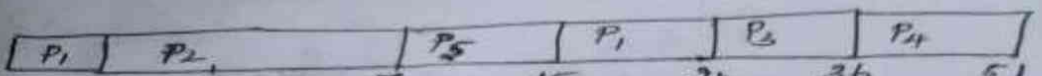
Process	Burst Time (ms)	Arrival Time (ms)	Priority
P1	10	0	4
P2	6	1	2
P3	12	2	1
P4	15	2	3
P5	8	3	5

Illustrate the states of scheduling using GANTT charts and calculate the performance over the average waiting time and average turnaround time metrics to find out the optimal scheduling algorithm from the following,

a) Preemptive SJF(SRTF) [4 Marks]

b)

SRTF [Preemptive SJF]



At time 0,
$P_1 \rightarrow$ Ready

At time 1,
$P_2 \rightarrow$ releases Compare P_1 & P_2
$P_1 = 10$ ms $P_2 = 6$ ms.
$P_2 \rightarrow$ Executes.

At time 2,
P_3 & P_4 released, Compare Burst time
$P_1 = 9$ ms, $P_2 = 5$ ms $P_3 = 12$
$P_4 = 15$ ms.
$P_2 \rightarrow$ continues.

At time 3
$P_5 \rightarrow$ release
Compare, $P_1 = 7$ ms, $P_2 = 4$ ms $P_5 = 12$ ms
$P_4 = 15$ ms

$P_2 \rightarrow$ Continues.
After completion of P_2, Compare B.T
$P_2 \rightarrow P_5 \rightarrow P_1 \rightarrow P_3 \rightarrow P_4$

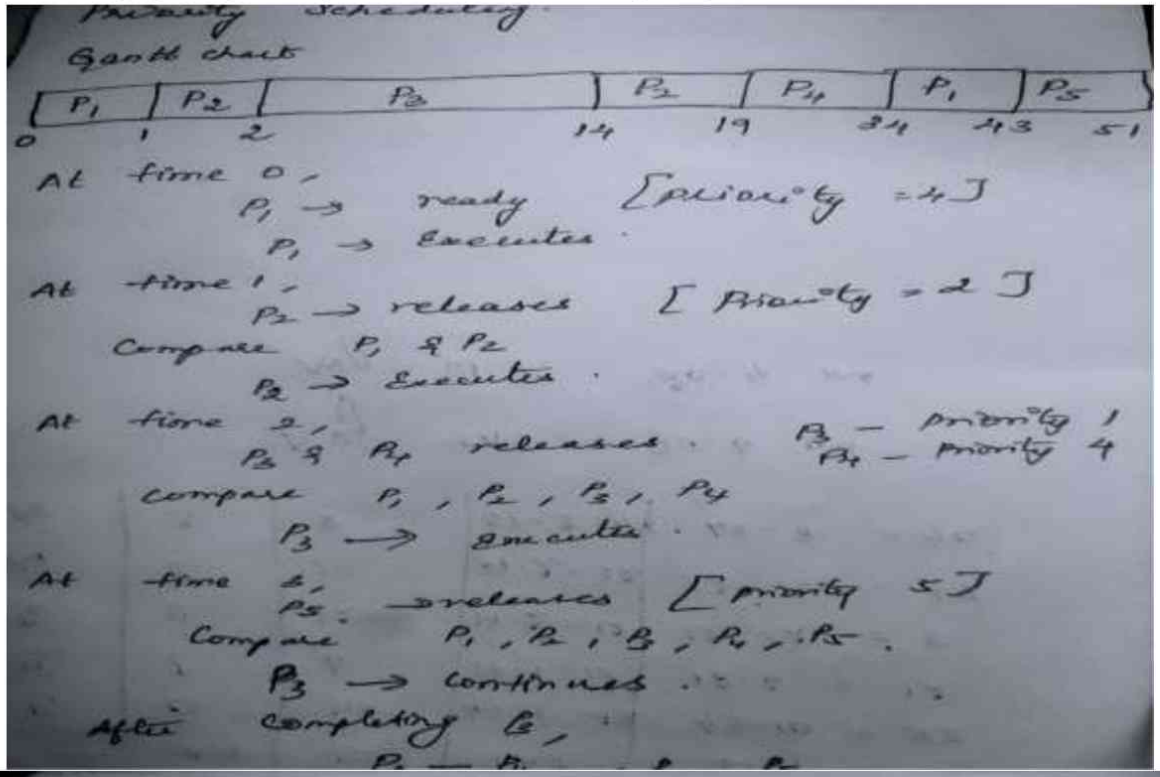
	Arrival	TAT = Completion − Arrival	WT = TAT − B.T	Burst time
P_1	0	24 − 0 = 24	24 − 10 = 14	10
P_2	1	7 − 1 = 6	6 − 6 = 0	6
P_3	2	36 − 2 = 34	34 − 12 = 22	12
P_4	2	51 − 2 = 49	49 − 15 = 34	15
P_5	3	15 − 3 = 12	12 − 8 = 4	8

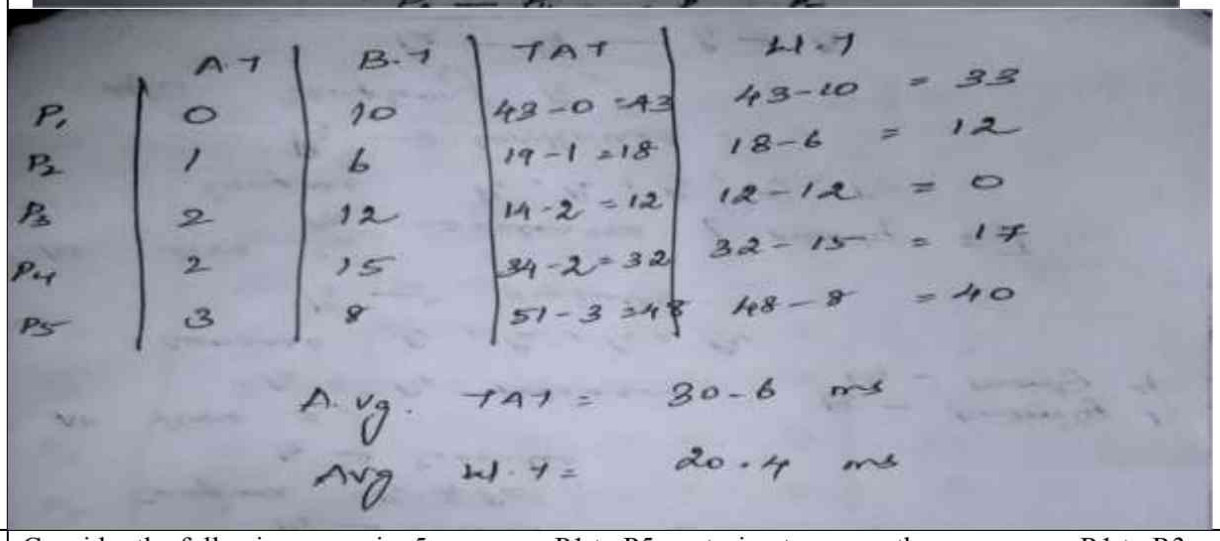
Average TAT = 24 + 6 + 34 + 49 + 12 / 5
= 25 ms

Average WT = 14 + 0 + 22 + 34 + 4 / 5
= 14.8 ms

Round Robin (with quantum = 2ms) [3 Marks] - Attached Separately

c) Priority scheduling that brings out the best performance in the processor. Lowest number denotes high priority and highest number denotes low priority. [3 Marks]





2. Consider the following scenario. 5 processes P1 to P5 are trying to access three resources R1 to R3, where R1 has 7 instances, R2 has 4 instances, and R3 has 6 instances in total. At time t0, P1 is assigned an instance of R2, P2 is assigned two instances of R1, P3 is assigned 3 instances of R1 and two instances of R3, P4 is assigned two instances of R1, an instance each of R2 and R3, and P5 is assigned two instances of R3. The maximum requests for resources (R1, R2, R3) that the processes might request for is given below.

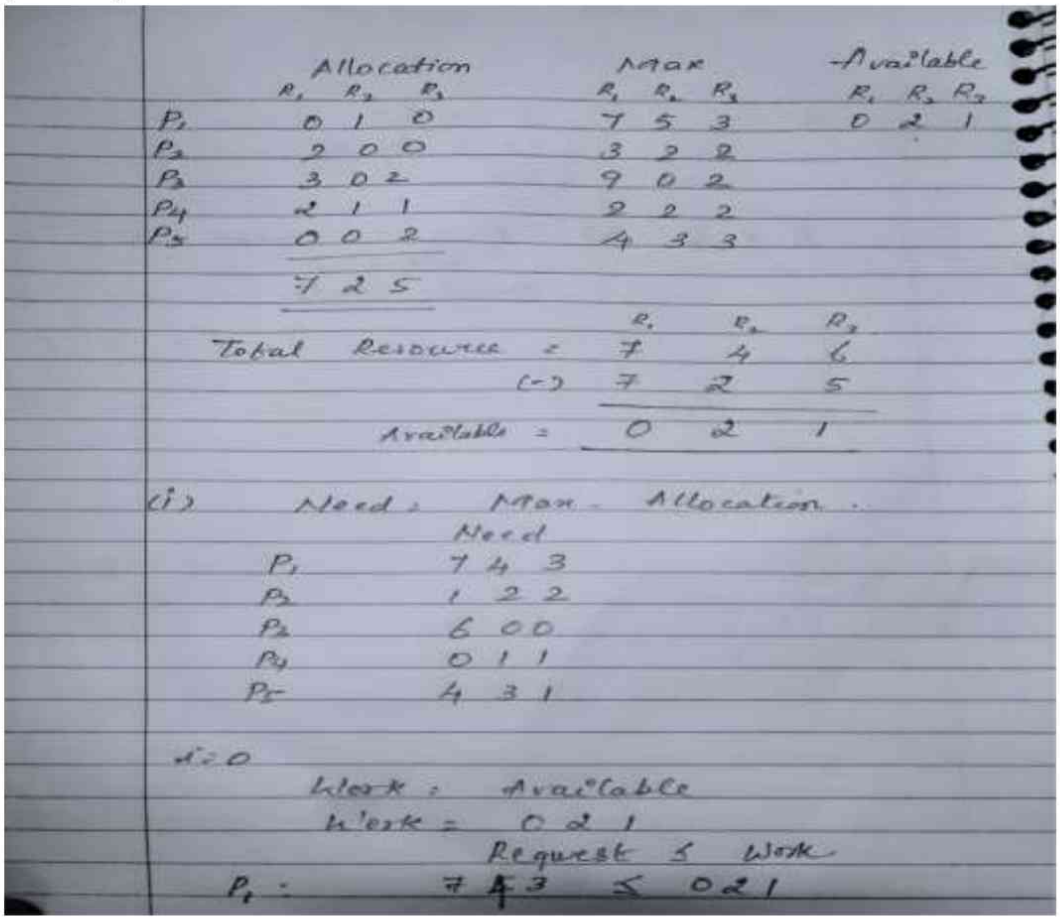
MAX

P1 7 5 3

P2 3 2 2

P3 9 0 2
P4 2 2 2
P5 4 3 3

a) Derive the Need Matrix for the above mentioned scenario [3 Marks]
b) Identify the current state of the system[safe/unsafe] [5 Marks]
c) If a request comes from P1(1,2,1), can the request be granted immediately. Substantiate your answer [2 Marks]



P_1 → wait

P_2:
122 ≤ 021
P_2 → wait

P_3:
600 ≤ 021
P_3 → wait

P_4:
011 ≤ 021
work = work + Available
= 021 + 211
work = 232

P_5:
131 ≤ 232
P_5 → wait

F	F	F	T	F
P_1	P_2	P_3	P_4	P_5

P_2:
122 ≤ 232
work = 232 + 200
= 432

P_5: 431 ≤ 432
Work: 432 + 002
work = 434

P_1 & P_3 → resource cannot be allocated.
Unsafe state.

F	T	F	T	T

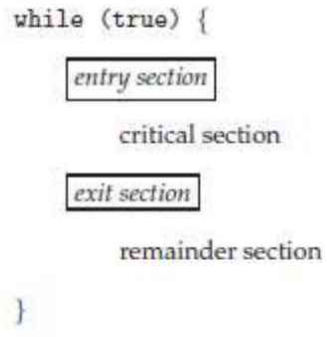
(c) Since the system is already in unsafe state, the request by P_1 cannot be granted immediately.

3.	A gaming console offers 5 play-stations mounted over a round shaped table. A team of 2 players occupies each system. 5 Controls are also arranged in a circular fashion, between the systems, such that a control is shared by two neighboring systems. Each team can access the system only if they get access to both left side and right side controls at the same time. The states of every team alternates between PLAY and IDLE modes. Illustrate the synchronization problem occurring in this scenario with a neat sketch. Quoting necessary programming constructs, also elaborate a solution for this problem using semaphore and using monitors. • N Player teams sit at a round table with a Playstations in the middle. • They spend their lives alternating idle and play. • They do not interact with their neighbors. • Occasionally try to pick up 2 controllers (one at a time) to play • Need both to play, then release both when done Solution using Semaphore The structure of player team i : **while (true){** **wait (controller[i]);** **wait (controller[(i + 1) % 5]);** **/* play for awhile */** **signal (controller[i]);** **signal (controller[(i + 1) % 5]);** **/* idle for a while */** **}** Solution using monitors: **monitor Players** **{** **enum {THINKING; IDLE, PLAY} state [5];** **condition self [5];** **void pickup (int i) {** **state[i] = IDLE;** **test(i);** **if (state[i] != PLAY) self[i].wait;** **}** **void putdown (int i) {** **state[i] = THINKING;** **// test left and right neighbors** **test((i + 4) % 5);** **test((i + 1) % 5);** **}** **void test (int i) {** **if ((state[(i + 4) % 5] != PLAY) &&** **(state[i] == IDLE) &&** **(state[(i + 1) % 5] != PLAY)) {** **state[i] = PLAY ;** **self[i].signal () ;**

```
        }
    }
    initialization_code() {
        for (int i = 0; i < 5; i++)
        state[i] = THINKING;
    }
}
```

4. a) Explain the critical section problem in process synchronization focusing on the minimum requirements to be satisfied to resolve the critical section problem. [5 Marks]

- Consider system of ***n*** processes $\{p_0, p_1, \ldots p_{n-1}\}$
- Each process has **critical section** segment of code
  - Process may be changing common variables, updating table, writing file, etc.
  - When one process in critical section, no other may be in its critical section
- ***Critical section problem*** is to design protocol to solve this
- Each process must ask permission to enter critical section in **entry section**, may follow critical section with **exit section**, then **remainder section**



- **Requirements:**

  Mutual Exclusion - If process P_i is executing in its critical section, then no other processes can be executing in their critical sections

  Progress - If no process is executing in its critical section and there exist some processes that wish to enter their critical section, then the selection of the process that will enter the critical section next cannot be postponed indefinitely

  Bounded Waiting - A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted

b) Illustrate a classic software-based solution for the critical section problem. Identify its difficulties in supporting modern computer architecture. Briefly discuss the possible alternatives to overcome the issue. [5 Marks]

- **Peterson's Solution**

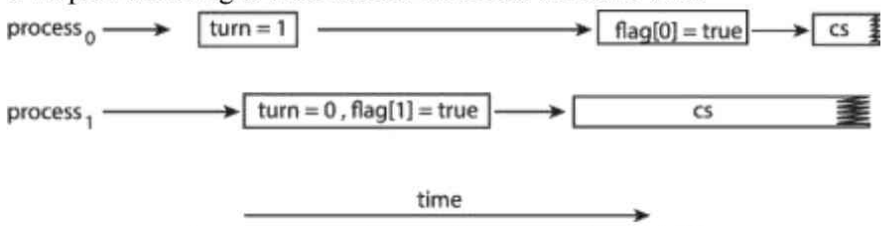
```
while (true){
    flag[i] = true;
    turn = j;
    while (flag[j] && turn = = j)
            ;
      /* critical section */

    flag[i] = false;

    /* remainder section */

}
```

- Peterson's Solution is not guaranteed to work on modern architectures.

	• To improve performance, processors and/or compilers may reorder operations that have no dependencies • Two processes might enter critical section at the same time.  Solution: Memory Barriers, Hardware instructions, Atomic Variables
5.	Given six memory partitions of 300 MB, 170 MB, 40 MB, 205 MB, 100 MB, and 185 MB (in order), illustrate the operation of first-fit, best-fit, and worst-fit algorithms over processes of size 20 MB, 150 MB, 185 MB, 75 MB, 175 MB, and 80 MB (in order)? Indicate which—if any—requests cannot be satisfied. Comment on how efficiently each of the algorithms manages memory. [10 Marks] **Solution** Available memory partitions F1=300MB, F2= 170MB, F3=40MB, F4=205MB, F5=100MB, and F6=185MB. Requesting memory P1=20MB, P2=150MB, P3=185MB, P4=75MB, P5=175MB, and P6=80MB. Three proposed algorithms were used to define memory space and properly labelled processes. Using **First fit algorithm**: P1 will be allocated to F1. With this, F1 will have a remaining space of 280MB from (300 - 20). P2 will be allocated to remaining space of F1. With this, F1 will have a remaining space of 130 MB from (280 - 150). P3 will be allocated F4. With this, F4 will have a remaining space of 20MB from (205 - 185). P4 will be allocated to the remaining space of F1. Since F1 has a remaining space of 130MB, if P4 is assigned there, the remaining space of F1 will be 55MB from (130 - 75). P5 will be allocated to F6. With this, F6 will have a remaining space of 10MB from (185 - 175). P6 will be allocated to F2. With this, F2 will have a remaining space of 90MB from (170 - 80). Using **Best-fit algorithm**: P1 will be allocated to F3. With this, F3 will have a remaining space of 20MB from (40 - 20). P2 will be allocated to F2. With this, F2 will have a remaining space of 20MB from (170-150). P3 will be allocated to F6. With this, F6 will have no remaining space as it is entirely occupied by P3. P4 will be allocated to F5. With this, F5 will have a remaining space of of 25MB from (100 - 75). P5 will be allocated to F4. With this, F4 will have a remaining space of 35MB from (205 - 175).

	P6 will be allocated to the part of the remaining space of F1. Therefore, F1 will have a remaining space of 220MB from (300 - 80). Using **Worst-fit algorithm**: P1 will be allocated to F1. Therefore, F1 will have a remaining space of 280MB from (300 - 20). P2 will be allocated to F1. Therefore, F1 will have a remaining space of 130MB from (280-150). P3 will be allocated to part of F4. Therefore, F4 will have a remaining space of 20MB from (205-185). P4 will be allocated to F6. Therefore, the remaining space of F6 will be 110MB from (185 - 75). P5 will not be allocated to any of the available space because none can contain it. P6 will be allocated to F2. Therefore, F2 will have a remaining space of 90MB from (170 - 80). **Best fit algorithm** is the best as the name suggests, while the **worst fit algorithm** is the worst as **not all memory is allocated**