SRM Institute of Science and Technology
College of Engineering and Technology
School of Computing
SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamilnadu

SET C

Academic Year: 2024-25 (ODD)

Test: FJ2 | **Date: 01.10.2024**
Course Code & Title: 21CSC202J - Operating Systems | **Duration:** 100 Minutes
Year & Sem: II Year / III Sem | **Max. Marks:** 50

Course Articulation Matrix: *(to be placed)*

S.No	Course Outcome	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
1	CO1	3	3	2	2								3			
2	CO2	3	3	3	2								3			
3	CO3	3	3	3	2								3			
4	CO4	3	3	3	2								3			
5	CO5	3	2	3	2								3			

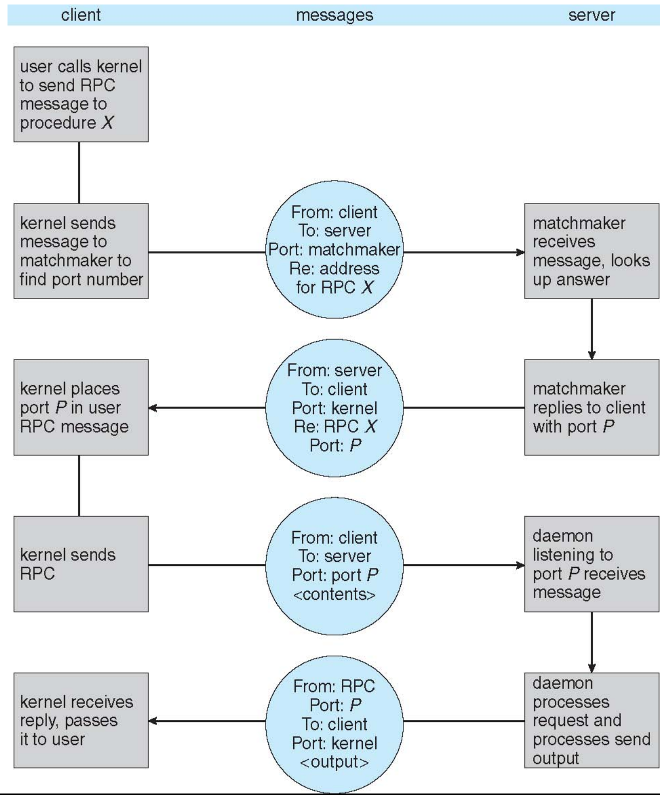
Questions with Answer Key

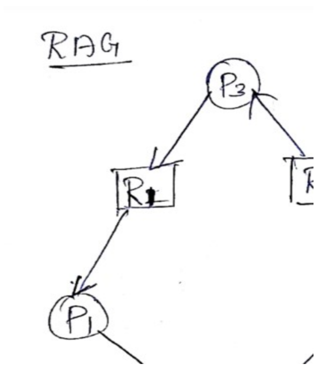
Part – A
(10 x 1 = 10 Marks)
Instructions: Answer all

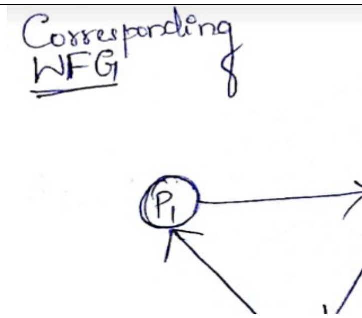
Q. No	Answer with choice variable	Marks	BL	CO	PO	PI Code
1	**A process can be terminated due to __________** a) normal exit b) open files c) signals and signal handlers d) return addresses Answer: a) normal exit	1	L1	2	2	1.2.2
2	**Which of the following schedulers is the fastest?** a. Medium Term Scheduler b. Short Term Scheduler c. Job Scheduler d. Long term Scheduler Answer : b) Short Term Scheduler	1	L1	2	3	1.6.1

3	**--------- unix command system call creates new process.** a) fork b) exec c) ioctl d) longjmp Answer : a) fork	**1**	**L2**	**2**	**2**	**1.6.1**
4	**The connection between two processes, P and Q, for sending and receiving messages is known as __________** a) communication link b) message-passing link c) synchronization link d) network link Answer: a) communication link	**1**	**L2**	**2**	**3**	**1.5.1**
5	**Thread-local storage (TLS) allows each thread to have** its own copy of data a) Thread memory b) Process-local storage c) Thread-local storage d) Deferred cancellation Ansewer: c) Thread-local storage	**1**	**L2**	**2**	**3**	**1.6.1**
6	Regarding the Multilevel Queue Scheduling algorithm, which of the following is true? a) Processes are permanently assigned to a queue b) Each queue has its own scheduling algorithm c) Processes can move between queues d) Both a and b Answer: d) Both a and b	**1**	**2**	**3**	**1**	**1.6.1**

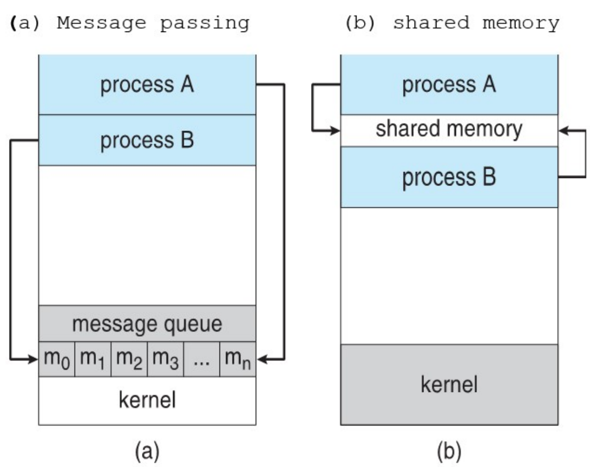
7	Deadlock can be characterized by a circular wait condition. What does this imply? A) Processes can only request resources in a specific order. B) There exists a set of processes {P1, P2, ..., Pn} such that P1 is waiting for a resource held by P2, P2 is waiting for a resource held by P3, ..., and Pn is waiting for a resource held by P1. C) Processes hold resources and wait for additional resources simultaneously. D) Processes can only request resources in a random order Answer: b) There exists a set of processes {P1, P2, ..., Pn} such that P1 is waiting for a resource held by P2, P2 is waiting for a resource held by P3, ..., and Pn is waiting for a resource held by P1	**1**	**2**	**3**	**2**	**2.1.2**
8	**Which scheduling algorithm assigns a higher priority to tasks with shorter periods?** A) Earliest-Deadline-First (EDF) Scheduling B) Round-Robin Scheduling C) Proportional Share Scheduling D) Rate-Monotonic Scheduling **Answer:** D)) Rate-Monotonic Scheduling	**1**	**1**	**3**	**1**	1.6.1
9	Which of the following is not a method for recovering from deadlock? A. Process Termination B. Resource Preemption C. Deadlock Prevention D. Rollback **Answer:** C) Deadlock Prevention	**1**	**1**	**3**	**1**	1.3.1
10	The time from the submission of a request until the first response is produced is called ---------------------- Waiting Time Turnaround Time Response Time Submission Time Answer: c) Response Time	**1**	1	3	1	1.3.1
	Part – B **(4 x 5 = 20 Marks)**					
11	Explain the role of RPC in client server communication with necessary execution diagram. Answer: • Remote procedure call (RPC) abstracts procedure calls between processes on networked systems – Again uses ports for service differentiation • **Stubs** – client-side proxy for the actual procedure on the server • The client-side stub locates the server and **marshalls** the parameters	**4**	**L2**	**2**	**3**	**1.6.1**

	• The server-side stub receives this message, unpacks the marshalled parameters, and performs the procedure on the server • On Windows, stub code compile from specification written in **Microsoft Interface Definition Language** (**MIDL**) 					
12	**Discuss the solution to the critical section problem.** **Answer:** Mutual Exclusion - If process P_i is executing in its critical section, then no other processes can be executing in their critical sections 2. 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 processes that will enter the critical section next cannot be postponed indefinitely 3. 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	**4**	**L2**	**2**	**3**	**2.6.2**
13	Explain the concept of CPU scheduling and describe the criteria used to evaluate different CPU scheduling algorithms. ANS: CPU scheduling is the process of determining which processes in the ready queue are to be allocated to the CPU for execution. The criteria used to evaluate CPU scheduling algorithms include: • CPU Utilization • Throughput	**4**	**2**	**3**	**1**	**1.3.1**

	• Turnaround Time • Waiting Time					
14	Consider a system with three processes, P1, P2, and P3, and three resources, R1, R2, and R3. The resources have the following allocations and requests: • P1 holds R1 and requests R2. • P2 holds R2 and requests R3. • P3 holds R3 and requests R1. Draw the Resource Allocation Graph (RAG) and determine if a deadlock exists. If yes, which processes are involved in the deadlock? **Answer:** **Resource Allocation Graph (RAG)** In a Resource Allocation Graph, we represent processes as circles (P1, P2, P3) and resources as rectangles (R1, R2, R3). Draw edges from processes to resources they are requesting and from resources to processes that currently hold them. • **P1 → R2**: P1 is requesting R2. • **R1 → P1**: P1 holds R1. • **P2 → R3**: P2 is requesting R3. • **R2 → P2**: P2 holds R2. • **P3 → R1**: P3 is requesting R1. • **R3 → P3**: P3 holds R3.  In this graph, each process is holding one resource and waiting for another resource, forming a cycle. The cycle is P1 → R2 → P2 → R3 → P3 → R1 → P1. Hence, there is a deadlock involving processes P1, P2, and P3. **Wait-for Graph (WFG)** A Wait-for Graph is derived from the RAG by collapsing the resources and directly showing which processes are waiting for which other processes. From the RAG:	**4**	**3**	**3**	3	3.2.3

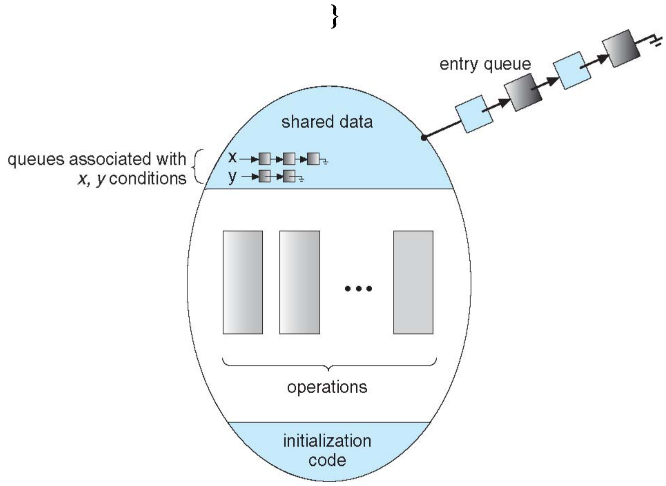
	 • P1 is waiting for R2 which is held by P2 (P1 → P2). • P2 is waiting for R3 which is held by P3 (P2 → P3). P3 is waiting for R1 which is held by P1 (P3 → P1).					
	Part – C **Either OR Choice Questions** **(2 X 10 = 20 Marks)**					

15a	**Write about the various system calls used in process creation and termination.**
	Discuss the two models in IPC with a diagram.
	Answer:
	i) Process Creation:
	• **Parent process create children processes, which, in turn create other processes, forming a tree of processes**
	• **Generally, process identified and managed via a process identifier (pid)**
	• **Resource sharing options**
	  ○ **Parent and children share all resources**
	  ○ **Children share subset of parent's resources**
	  ○ **Parent and child share no resources**
	• **Execution options**
	  a. **Parent and children execute concurrently**
	  b. **Parent waits until children terminate**
	  c. **fork() system call creates new process**
	  d. **exec() system call used after a fork() to replace the process' memory space with a new program**
	ii) Process Termination:
	• **Process executes last statement and then asks the operating system to delete it using the exit() system call.**
	  ○ **Returns status data from child to parent (via wait())**
	  ○ **Process' resources are deallocated by operating system**
	• **Parent may terminate the execution of children processes using the abort() system call. Some reasons for doing so:**
	  a. **Child has exceeded allocated resources**
	  b. **Task assigned to child is no longer required**
		10	**L4**	**2**	**2**	**2.6.3**

	c. The parent is exiting and the operating systems does not allow a child to continue if its parent terminates
	
	Shared Memory :
	• **An area of memory shared among the processes that wish to communicate**
	• **The communication is under the control of the users processes not the operating system.**
	• **Major issues is to provide mechanism that will allow the user processes to synchronize their actions when they access shared memory.**
	Message Passing:
	• **Mechanism for processes to communicate and to synchronize their actions**
	• **Message system – processes communicate with each other without resorting to shared variables**
	• **IPC facility provides two operations:**
	  ○ **send(*message*)**
	  ○ **receive(*message*)**
	• **The *message* size is either fixed or variable**
	
	Communications Models
						

(or)						
15b	Discuss monitors and draw the structure of monitors with condition variables. How monitor addresses the Dining philosopher problem. Answer: • A high-level abstraction that provides a convenient and effective mechanism for process synchronization • *Abstract data type*, internal variables only accessible by code within the procedure • Only one process may be active within the monitor at a time • But not powerful enough to model some synchronization schemes	**10**	**L4**	**2**	**3**	**2.6.1**

monitor monitor-name
{
// shared variable declarations
procedure P1 (…) { …. }
procedure Pn (…) {……}
Initialization code (…) { … }
}
}




```
monitor DiningPhilosophers
{
enum { THINKING; HUNGRY, EATING) state [5] ;
   condition self [5];

   void pickup (int i) {
      state[i] = HUNGRY;
      test(i);
      if (state[i] != EATING) 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] != EATING) &&
        (state[i] == HUNGRY) &&
        (state[(i + 1) % 5] != EATING) ) {
            state[i] = EATING ;
         self[i].signal () ;
        }
}

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

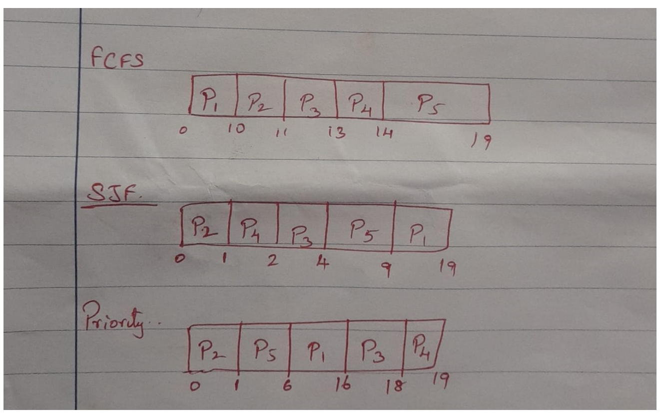
16a	Consider a system with five processes (P0 to P4) and three resource types (A, B, and C). The system has the following allocation and maximum need matrices: **Allocation:** A B C P0 0 1 0 P1 2 0 0 P2 3 0 2 P3 2 1 1 P4 0 0 2 **Maximum:** A B C P0 7 5 3 P1 3 2 2	**10**	3	3	3	3.2.3

	P2 9 0 2 P3 2 2 2 P4 4 3 3 Available resources are A=3, B=3, C=2. Use the Banker's Algorithm to determine if the system is in a safe state. **Answer:** 1. Calculate the Need matrix by subtracting Allocation from Maximum: A B C P0 7 4 3 P1 1 2 2 P2 6 0 0 P3 0 1 1 P4 4 3 1 2. The Available vector is initially A=3, B=3, C=2 3. Find a process whose needs can be met with the available resources: ○ P1: Need(1, 2, 2) ≤ Available(3, 3, 2), so execute P1. ▪ Update Available: Available = Available + Allocation of P1 = (3+2, 3+0, 2+0) = (5, 3, 2). 4. Repeat step 3: ○ P3: Need(0, 1, 1) ≤ Available(5, 3, 2), so execute P3. ▪ Update Available: Available = Available + Allocation of P3 = (5+2, 3+1, 2+1) = (7, 4, 3). 5. Continue the process: ○ P0: Need(7, 4, 3) ≤ Available(7, 4, 3), so execute P0. ▪ Update Available: Available = Available + Allocation of P0 = (7+0, 4+1, 3+0) = (7, 5, 3). ○ P2: Need(6, 0, 0) ≤ Available(7, 5, 3), so execute P2. ▪ Update Available: Available = Available + Allocation of P2 = (7+3, 5+0, 3+2) = (10, 5, 5). ○ P4: Need(4, 3, 1) ≤ Available(10, 5, 5), so execute P4. ▪ Update Available: Available = Available + Allocation of P4 = (10+0, 5+0, 5+2) = (10, 5, 7). Since all processes can be executed, the system is in a safe state.					
(Or)						
16b	Given below are the burst times and priorities of four five processes P1, P2, P3, P4 and P5. The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.	**10**	3	3	3	3.2.3

Process	Priority	Burst tim
P1	3	10
P2	1	1
P3	3	2
P4	4	1

a. Draw four Gantt charts illustrating the execution of these processes using FCFS,
SJF, non preemptive priority (a smaller priority number implies a higher priority), and
RR (quantum = 1) scheduling.
b. What is the turnaround time of each process for each of the scheduling algorithms?
c. What is the waiting time of each process for each of the scheduling algorithms?
d. Which of the schedules in part results in the minimal average waiting time?

Solution:



Round Robin

P1	P2	P3	P4	P5	P1	P3	P5	P1	P5	P1	P5	P1	P5	P1	P1	P1

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

b) Turn Around Time = Completion Time − Arrival Time (0)

Process	FCFS	SJF	Priority	RR
P_1	10	19	16	19
P_2	11	1	1	2
P_3	13	4	18	7
P_4	14	2	19	4
P_5	19	9	6	14

c) Waiting Time = Turn Around Time − BT

Process	BT	FCFS	SJF	Priority	RR
P_1	10	0	9	6	9
P_2	1	10	0	0	1

d) Average Waiting Time

$$FCFS = \frac{0+10+11+13+14}{5} = 9.6$$

$$SJF = \frac{9+0+2+1+4}{5} = 3.2$$

$$Priority = \frac{6+0+16+18+1}{5} = 8.2$$

$$RR = \frac{9+1+5+3+9}{5} = 5.4$$

SJF has minimal average Waiting Time = 3.2

Course Outcome (CO) and Bloom's level (BL) Coverage in Questions