MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

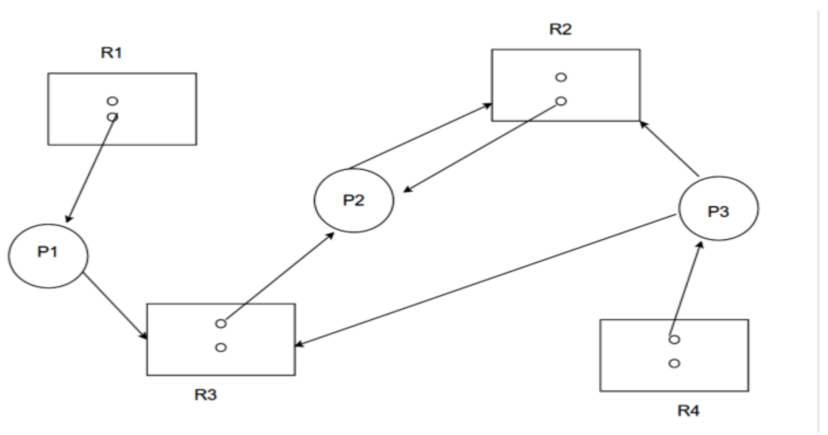
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Q.No.	Question	Marks	CLO/CO	AHEP LO	BTL
1	Which multithreading model provides the best balance between scalability and efficient resource usage? A. One-to-One Model B. Many-to-One Model C. **Many-to-Many Model** D. Single-Level Model	0.5	1	1	2
2	In which multithreading model can a blocking system call by one thread cause all threads in the process to block? A. One-to-One Model B. **Many-to-One Model** C. Many-to-Many Model D. Two-Level Model	0.5	1	1	2
3	What is the primary role of the job queue in process scheduling? A. To store processes waiting for I/O completion B. To store processes that are waiting to be executed C. To store processes that have finished execution D. **To store processes that are waiting to get loaded into memory**	0.5	1	1	2
4	By defining a linear ordering of the resource types can be prevented A. No preemption **B. Circular wait** C. Hold and wait D. Mutual exclusion	0.5	3	2	2
5	Consider a system consisting of 4 resources of the same type shared by 3 processes, each of which needs at most two resources. Which of the following is true? A. The system is not safe **B. The system is deadlock-free** C. The system is in a deadlock. D. Process undergoes starvation	0.5	3	2	2
6	The dispatcher picks the next process to be executed in the CPU from the _____________ A) job queue B) **ready queue** C) execution queue D) process queue	0.5	3	2	2

7	The interval from the time of submission of a process to the time of completion is termed as ____________ A) Waiting time B) **Turnaround time** C) Response time D) Burst time	0.5	3	2	2
8	__________ provide the means for a user program to ask the operating system to perform tasks reserved for the operating system on the user program's behalf. A) **System calls** B) Application programs C) Softwares D) Kernel calls	0.5	1	1	2
9	The operating system is not responsible for the following activities in connection with memory management. A) Keeping track of which parts of memory are currently being used and who is using them B) Deciding which processes (or parts of processes) and data to move into and out of memory C) Allocating and deallocating memory space as needed D) **Mapping files from memory onto secondary storage**	0.5	1	1	2
10	Which of the statements is true in case of MS-DOS A) It is single-tasking B) The command interpreter makes the previous error code available to the user or to the next program. C) It loads the program into memory, writing over most of itself to give the program as much memory as possible **D) All of the above**	0.5	1	1	2

Q.No.	Question	Marks	CLO/CO	AHEP LO	BTL
11	Construct the RAG for the following snapshot of the system Each Resources R1, R2, R3 and R4 have two instances each. There are 3 processes P1, P2 and P3. P1 has R1 and wants R3. P2 has R3, R2 and needs another R2. P3 needs R3, R2 and has R4. Now for the constructed RAG obtain the allocated matrix and need matrix. Find whether the system is in safe state or not. If safe then give the safe sequence else justify why it is not safe.	4	3	2	4

Ans:



1M

Available: 1 1 1 1,

Allocated matrix

	R1	R2	R3	R4
P1	1	0	0	0
P2	0	1	1	0
P3	0	0	0	1

1M

Need Matrix:

	R1	R2	R3	R4
P1	0	0	1	0
P2	0	1	0	0
P3	0	1	1	0

1M

For a process to be in safe sequence its need should be less than equal to available. If the condition is true then add the allocated value to the available value

P1: 0 0 1 0 <1 1 1 1

Available = 1 1 1 1 + 1 0 0 0 = 2 1 1 1

P2: 0 1 0 0 < 2 1 1 1

Available = 2 1 1 1 + 0 1 1 0 = 2 2 2 1

P3: 0 1 1 0 < 2 2 2 1

Available = 2 2 2 1 + 0 0 0 1 = 2 2 2 2

Hence safe sequence is P1->P2->P3 1M

12	Which interprocess communication (IPC) method would be most suitable for this system? Justify your answer with an explanation based on the specific requirements.	3	1	1	5

	Scenario: There is a requirement to develop a real-time system for managing a lab operation. Multiple sensors distributed across the lab are responsible for collecting data about temperature, instrument status (working/not working), and instrument availability. These sensors run as separate processes on the same system. They need to communicate with a central process that aggregates and analyzes the data in real time. The data needs to be transferred between processes efficiently, and the communication must be fast and reliable. Moreover, the system should handle a large volume of data being sent continuously and simultaneously from multiple sensors. Requirements: 1. Low latency: The communication between sensors and the central process must be fast. 2. Simultaneous communication: Multiple sensors may try to send data at the same time. 3. Data synchronization: Ensure that data from the sensors is processed correctly without being lost or misinterpreted. 4. Unidirectional communication: Data flows from sensors to the central process. Answer: Identifying Message queue ----**1M** The most suitable IPC method in this scenario would be **Message Queues** since one central process has to read the data without modifying it in the same location accessed by other processes (no shared memory concept). **0.5M** Further, multiple processes can write to the queue simultaneously which will be queued and can be read by the central process. **0.5M** The data will be sent to the queue and there is a very low possibility of losing the data from the queue. **0.5M** Message queues fit the one-way communication. **0.5M**				
13	Consider a system with three processes **Process A:** currently running, **Process B:** waiting for I/O completion, and **Process C:** waiting for the CPU to become available. Each process is associated with a Process Control Block (PCB) to its information. Process A finishes its time slice, and the CPU is given to Process C. However, Process A has not completed its work. Analyse the given scenario and identify the contents of Process State and Program Counter in the PCB before and after the Process transfers the control to Process C. **Answer:** Before: Process State: 0.5M • Process A: Running • Process B: Waiting (for I/O) • Process C: Ready (to run)	3	1	1	5

	PC: 0.5M • Process A: The current instruction address in its PCB as it is executing. • Process B: The address where it will resume after I/O completion. • Process C: The instruction to be executed once it gets CPU time. After: Process State: 0.5M • Process A: Ready (to run) • Process B: Waiting (for I/O) • Process C: Running PC: 0.5M • Process A: The instruction to be executed once it gets CPU time. • Process B: The address where it will resume after I/O completion. • Process C: The current instruction address in its PCB as it is executing.				
14	Analyze the roles of User Mode and Kernel Mode in an operating system. How does the interaction between these modes facilitate safe system operation? Illustrate your explanation with a real-life scenario, such as saving a document in a word processor. Ans: User Mode operates with restricted privileges, limiting direct access to system hardware and critical resources, while Kernel Mode has unrestricted access, allowing the operating system to manage hardware and execute privileged instructions. This separation enhances system stability and security by preventing user applications from directly interacting with critical resources. For instance, when saving a document in a word processor (which runs in user mode), the application initiates a system call to request the operating system (running in kernel mode) to write the data to the disk. This interaction ensures that the application cannot directly manipulate the hardware, maintaining system integrity by allowing the OS to manage file operations securely and efficiently.	3	1	1	4
15	Analyze the algorithm of Peterson's Solution for mutual exclusion between two processes. Explain how the algorithm ensures mutual exclusion, and discuss any potential drawbacks when implemented on modern multi-core systems. Ans: Peterson's Solution is a classic algorithm used to ensure mutual exclusion between two processes. The algorithm works as follows: Algorithm for Process i:	3	2	2	4

	1. Set flag[i] = true (process i intends to enter the critical section). 2. Set turn = j (give the other process priority). 3. Wait while (flag[j] == true && turn == j) (wait if the other process wants to enter and has priority). 4. Critical section (enter when the condition is false). 5. Set flag[i] = false (release the critical section after finishing). This algorithm ensures mutual exclusion by preventing both processes from entering the critical section simultaneously. It uses the flag array to indicate a process's desire to enter and the turn variable to enforce which process should proceed. Potential Drawbacks: Peterson's Solution assumes strict ordering of operations, which can be problematic in modern multi-core systems due to factors like instruction reordering and cache coherence. Additionally, it is primarily effective in single-processor systems and may not scale well to environments with multiple processors or cores, where hardware-based solutions such as test-and-set or atomic operations are often preferred.				
16	Differentiate between logical address space and physical address. Consider the user process size is of 4096 Kb and the data transfer rate is 1Mbps .Calculate the swapping time in milliseconds. Ans: Logical address space is the set of all logical addresses generated by a program Physical address space is the set of all physical addresses generated by a program. Both the addresses differ during the execution time. 1M Time = process size / transfer rate = 4096 / 1024 1M for conversion = 4 seconds = 4000 milliseconds Now taking swap-in and swap-out time, the process will take 8000 milliseconds 1M	3	4	3	3
17	Consider the following snapshot of a system: *Allocation* A B C D — *Max* A B C D — *Available* A B C D *P*0: 0 0 1 2 — 0 0 1 2 — 1 5 2 0 *P*1: 1 0 0 0 — 1 7 5 0 *P*2: 1 3 5 4 — 2 3 5 6 *P*3: 0 6 3 2 — 0 6 5 2 *P*4: 0 0 1 4 — 0 6 5 6	2	3	2	4

Obtain the process sequence if it is in a safe state.
Ans:
Need matrix

	A	B	C	D
P0	0	0	0	0
P1	0	7	5	0
P2	1	0	0	2
P3	0	0	2	0
P4	0	6	4	2

0.5M

	A	B	C	D
Intially	1	5	2	0
After P0	1	5	3	2
After P2	2	8	8	6
After P3	2	14	11	8
After P4	2	14	12	12

With Steps 1M
Safe Sequence P0→P2→P3→P4→P5 0.5M

18	Justify the benefits of multithreaded programming with respect to Two-level multithreading model.	2	1	1	3

Ans: Benefits of thread with respect to Two-level model

1. Responsiveness. Multithreading an interactive application may allow
   a program to continue running even if part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness to the user.
2. Resource sharing.The benefit of sharing code and data is that it allows an application to
   have several different threads of activity within the same address space.
3. Economy Allocating memory and resources for process creation is costly. Because threads share the resources of the process to which they belong, it is more economical to create and context-switch threads
4. Scalability. The benefits of multithreading can be even greater in a multiprocessor architecture, where threads may be running in parallel on different processing cores

19	Consider the set of processes	2	3	2	5

Processes	**Arrival Time(ms)**	**Burst Time (ms)**	**Priority**
P1	3	10	1

P2	0	4	2
P3	1	6	0
P4	2	8	3

Use preemptive priority scheduling. Assume the lowest priority value of the process has a higher priority. Draw Gant chart and Find Turnaround Time of all processes.

Ans:

P2	P3	P1	P2	P4

0 1 7 17 20 28

TAT of p1=17-3=14msec
TAT of P2 = 20-0=20msec
TAT of P3 =7-1=6msec
TAT of P4 =28-2=26msec

Gantt Chart 1M
0.25M for each TAT 0.25x4=1M