Roll No .....

TCS-503

B. TECH. (CSE) (FIFTH SEMESTER)
END SEMESTER EXAMINATION, 2019

OPERATING SYSTEM

Time : Three Hours
Maximum Marks : 100

Note : (i) This question paper contains five questions.

(ii) All questions are compulsory.

(iii) Instructions **on** how to attempt a question are mentioned against it.

(iv) T

otal marks for each main question are **twenty.** 1. Attempt any *two* parts of choice from (a), **(b)** and (c). (10x2=20 Marks)

(a) Answer the following

(i) What is the basic goal of operating system ? Give various criteria and state how they help in getting best performance out of operating system. (5 Marks)

(ii) Discuss the various types of operating systems. List at least *two* specifications of each type. (5 Marks)

(b) An operating system uses the Shortest Remaining Time First (SRTF) process scheduling algorithm. Consider the arrival times and execution times for the following processes :

Process -	Burst Time	Arrival Time
P1	20	0
P2	25	15
P3	10	30
P4	15	45

Calculate following :

(i) Draw a Gantt chart for SRTF. (3 Marks)

(ii) What is the total waiting time for process P2 ? (2 Marks)

(iii) Calculate the average turn-around time. (3 Marks)

(iv) Calculate CPU utilization and throughput. (2 Marks)

(c) (i) Discuss Kernels and System calls. (5 Marks)

(ii) What is Virtual Machine ? Give some advantages of virtual machines. Also draw appropriate diagram in support of your answer. (5 Marks)

2. Attempt any *two* parts of choice from (a), (b) and (c). (10x2=20 Marks)

(a) Write short notes on the following : (5 Marks)

(i) Process state (5 Marks)

(ii) Context switch

(b) What do you understand from the term "Operating **System Structure" ?** .Elaborate its different approaches along with suitable **neat,** clean and labelled diagrams.

(c) Consider the following set of processes, with the length of CPU given in milliseconds :

Process	Burst Time	Priority
P1	8	4
P2	6	1
P3	1	2
P4	9	2
P5	3	3

The processes are assumed to have arrived in the order P1, **P2, P3, P4, P5** all at time 0.

(i) Draw Gantt charts for the Priority (Non-preemptive) and R-R scheduling algorithms. (4 Marks)

(ii) What is the waiting time of each process and average Waiting time for each of these scheduling algorithms ? (3 Marks)

(iii) What is the average turn-around time of each scheduling algorithms ? (3 Marks)

3. Attempt any *two* parts of choice from (a), (b) and (c). (10X2=20 Marks)

(a) Answer the following :

(i) Differentiate between Cooperative and Independent processes. (5 marks)

(ii) Race condition (Explain with the help of an example). (5 marks)

(b) What is Critical Section problem ? State Readers-Writers problem and give a solution. Also show that given solution satisfies all the requirements of a solution for critical section problem.

(c) Consider the following snapshot of a system :

	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P1	1	2	2	1	3	3	2	2	3	1	1	2
P2	1	0	3	3	1	2	3	4				
P3	1	2	1	0	1	3	5	0				

Answer the following questions using Banker's algorithm

(i) What is the content of matrix need ?

(ii) Is the system in safe state ? If no, explain the reason. If yes, give the safe sequence.

(iii) If a process from P1 arrives for (1, 1, 0, 2), can the request be granted immediately.

4. Attempt any *two* parts of choice from (a), (b) and (c). (10x2=20 Marks)

(a) Explain the following terms :

(i) Logical and Physical Address Space (3 Marks)

(ii) Swapping (3 Marks)

(iii) Thrashing (2 Marks)

(iv) Demand segmentation (2 Marks)

(b) Consider the following page reference string :

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

How many page faults will occur, if there are three frames and initially all are empty, using LRU, Optimal and FIFO Page replacement ?

(c) Given five memory partitions of 100 Kb, 500 Kb, 200 Kb, 300 Kb, 600 Kb (in order), how would the first-fit, best-fit and worst-fit algorithms place processes of 222 Kb, 419 Kb, 110 Kb and 420 Kb (in order) ? Which algorithm makes the most efficient use of memory ?

5. Attempt any *two* parts of choice from (a), (b) and (c). (10x2=20 Marks)

(a) Explain all types of directory structures using neat and labelled diagrams.

(b) Suppose that a disk drive has 5000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 135 and the previous request was at cylinder 125. **The queue of pending requests, rn FIFO order is :**

86, 1470, 913, 1774, 948, 1509, 1022, 1730, 130

Calculating from initial request, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms ?

(i) LOOK

(ii) SCAN

(iii) FCFS

(iv) C-LOOK

(c) Why do we need to Allocation methods ? Explain the various types of allocation methods with proper diagrams. Also state which method is better and why ?

SOLUTIONS

1. (a) (i) What is the basic goal of an operating system? Give various criteria and state how they help in getting the best performance out of the operating system. (5 Marks)

The basic goal of an operating system is to manage the hardware resources of a computer efficiently and to provide an environment where users and applications can run effectively. The OS ensures the smooth operation of a system by managing processes, memory, input/output devices, and file systems.

Criteria for OS performance:

- *Throughput: The number of processes completed in a given time. Maximizing throughput leads to higher efficiency in utilizing system resources.*
- *Turnaround Time: The time from submitting a process to its completion. Minimizing this time ensures faster execution of tasks.*
- *Waiting Time: The time a process spends waiting in the queue before execution. Reducing waiting time helps improve user experience and system performance.*
- *CPU Utilization: The percentage of time the CPU is actively working. A high CPU utilization indicates that the system is being effectively used.*
- *Fairness: Ensuring each process gets a fair share of resources. Fairness improves the system's responsiveness to all users.*

1. (a) (ii) Discuss the various types of operating systems. List at least two specifications of each type. (5 Marks)

1. *Batch Operating System:*
    - *Executes a series of jobs in a sequential manner without interaction.*
    - *Requires minimal user intervention once the job is submitted.*
2. *Time-Sharing Operating System:*
    - *Allows multiple users to share system resources by providing each user with a small time slice.*
    - *Offers interactive interfaces to users with rapid response times.*

3. *Distributed Operating System:*

- *Manages a group of independent computers and makes them appear as a single system to the user.*
- *Provides resource sharing and coordinated execution across multiple machines.*

4. *Real-Time Operating System (RTOS):*

- *Guarantees specific response times for critical processes.*
- *Used in embedded systems where timing is crucial, such as in robotics or medical devices.*

5. *Network Operating System (NOS):*

- *Manages resources on a network, allowing communication between devices and users.*
- *Provides centralized management of file sharing, printers, and other networked resources.*

---

1. (b) (i) Draw a Gantt chart for SRTF. (3 Marks)

For the processes given (P1, P2, P3, P4), we apply the Shortest Remaining Time First (SRTF) scheduling algorithm.

Processes:

- *P1: Arrival = 0, Burst = 20*
- *P2: Arrival = 15, Burst = 25*
- *P3: Arrival = 30, Burst = 10*
- *P4: Arrival = 45, Burst = 15*

The Gantt chart would look like:

Time P1 P2 P3 P4

0 P1

15 P2 P1

30 P2 P3

40 P4 P2

45 P4

60

---

1. (b) (ii) What is the total waiting time for process P2? (2 Marks)

Waiting Time Formula:

- *Waiting time = Turnaround time - Burst time*
- *Turnaround time = Completion time - Arrival time*

From the Gantt chart, the completion time for P2 is 40. P2 arrived at 15, and its burst time is 25.

- *Turnaround time = 40 - 15 = 25*
- *Waiting time = 25 - 25 = 0*

Thus, the total waiting time for process P2 is 0 ms.

---

1. (b) (iii) Calculate the average turn-around time. (3 Marks)

Turnaround Time Formula:

- *Turnaround time = Completion time - Arrival time*

From the Gantt chart, we calculate the completion times for each process:

- *P1: Completion = 40, Turnaround time = 40 - 0 = 40*
- *P2: Completion = 40, Turnaround time = 40 - 15 = 25*
- *P3: Completion = 50, Turnaround time = 50 - 30 = 20*
- *P4: Completion = 60, Turnaround time = 60 - 45 = 15*

Average Turnaround Time = (40 + 25 + 20 + 15) / 4 = 25 ms

---

1. (b) (iv) Calculate CPU utilization and throughput. (2 Marks)

- *CPU Utilization: The CPU is in use for the entire time span from the start to the end of the last process. From the Gantt chart, the total time the CPU is utilized is 60 units (from time 0 to time 60).*

CPU Utilization = (Total time CPU is used / Total time) × 100 = (60 / 60) × 100 = 100%

- *Throughput: Throughput is the number of processes completed per unit of time. In this case, 4 processes are completed within 60 units of time.*

Throughput = (Number of processes / Total time) = 4 / 60 = 0.0667 processes per unit time

---

1. (c) (i) Discuss Kernels and System Calls. (5 Marks)

- *Kernel:*
  - *The kernel is the core part of an operating system that controls everything in the system.*
  - *It manages hardware resources and provides a secure and stable environment for other programs to run.*
  - *The kernel performs tasks such as memory management, process scheduling, and hardware interaction.*
- *System Calls:*
  - *System calls are interfaces through which user programs interact with the kernel.*
  - *They provide services such as file management, process control, and communication.*
  - *Examples include open(), read(), write(), fork(), and exec().*

---

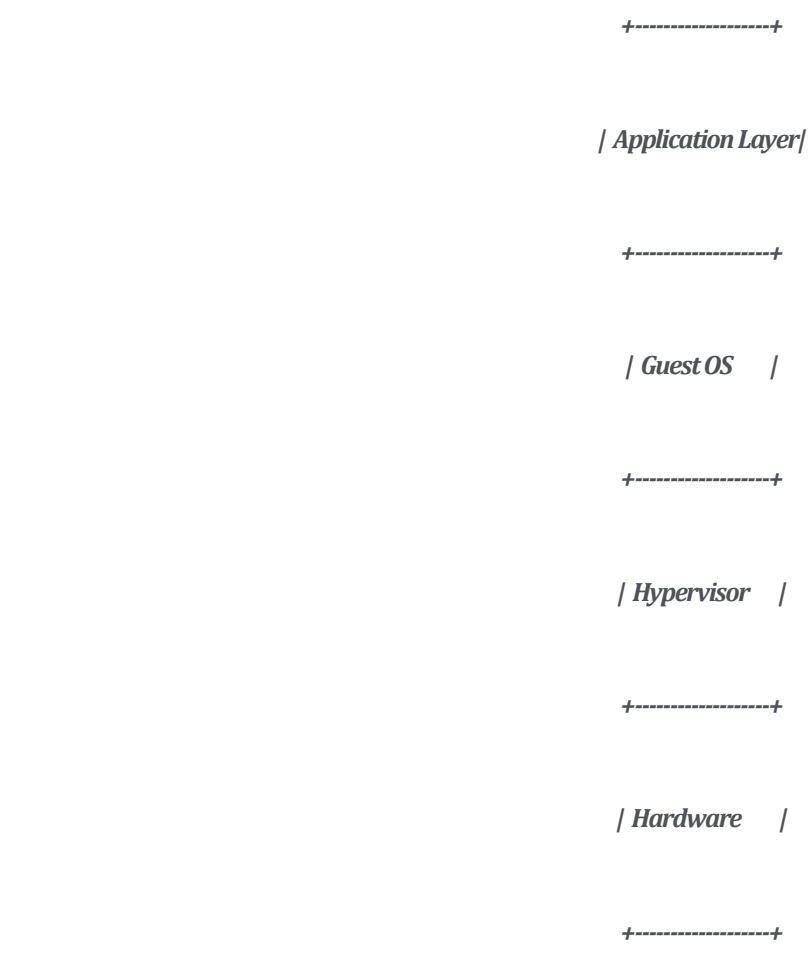
1. (c) (ii) What is Virtual Machine? Give some advantages of virtual machines. Also draw appropriate diagram in support of your answer. (5 Marks)

A Virtual Machine (VM) is a software-based emulation of a physical computer system. It runs an operating system and applications just like a physical computer, but it is isolated from the underlying hardware by a hypervisor.

Advantages of Virtual Machines:

- *Isolation: VMs are isolated from each other, ensuring that problems in one VM do not affect others.*
- *Resource Utilization: Multiple VMs can run on a single physical machine, making better use of available hardware.*
- *Portability: VMs can be easily moved between different physical machines.*
- *Snapshot and Cloning: VMs can take snapshots, enabling easy restoration of system states.*

A basic diagram of a VM structure is as follows:



---

2. (a) (i) Process State. (5 Marks)

A process in an operating system can exist in one of the following states:

1. *New: The process is being created.*
2. *Ready: The process is waiting for CPU time and is ready to execute.*
3. *Running: The process is currently being executed by the CPU.*
4. *Waiting: The process is waiting for some event or resource (e.g., I/O operation).*
5. *Terminated: The process has completed execution.*

Each state reflects the current activity of the process in the lifecycle.

---

2. (a) (ii) Context Switch. (5 Marks)

A context switch occurs when the operating system switches the CPU from one process to another. This involves saving the state of the current process and loading the state of the next process.

Steps in a context switch:

1. *Save the context (CPU registers, program counter, etc.) of the currently running process.*
2. *Update the process control block (PCB) of the current process.*
3. *Load the saved context of the next process.*
4. *Update the PCB of the new process.*
5. *Transfer control to the new process.*

Context switches are essential for multitasking but are costly in terms of time.

---

2. (b) What do you understand from the term "Operating System Structure"? Elaborate its different approaches along with suitable neat, clean, and labeled diagrams. (5 Marks)

Operating System Structure refers to how an operating system is organized and the design of its components. The structure defines how the OS services are provided and how communication occurs between different components of the OS.

Common Approaches:

1. *Monolithic Structure:*
   - *All OS services run in kernel mode and are part of a single large program.*

- *Advantages: High performance, simple design.*
- *Disadvantages: Difficult to maintain and extend.*

2. Layered Structure:

- *The OS is divided into layers, where each layer provides services to the next higher layer.*
- *Advantages: Easier to maintain, modular.*
- *Disadvantages: Slower due to multiple layers of abstraction.*

3. Microkernel:

- *The kernel provides only basic services, and additional services are implemented in user space.*
- *Advantages: More modular and secure.*
- *Disadvantages: Communication overhead between kernel and user space.*

4. Client-Server Architecture:

- *OS components are divided into client and server modules that communicate via messages.*
- *Advantage: Easy to extend and maintain.*
- *Disadvantage: Performance overhead due to communication.*

---

2. (c) Consider the following set of processes, with the length of CPU given in milliseconds:

Process Burst Time Priority P1 8 4 P2 6 1 P3 1 2 P4 9 2 P5 3 3

(i) Draw Gantt charts for the Priority (Non-preemptive) and R-R scheduling algorithms. (4 Marks)

Priority (Non-preemptive) Gantt Chart:

- *The process with the highest priority (lowest number) runs first. If processes have the same priority, they are scheduled based on their arrival order.*

Time P2 P3 P5 P4 P1

0 P2

6 P3

7 P5

10 P4

19 P1

---

2. (c) (ii) What is the waiting time of each process and average waiting time for each of these scheduling algorithms? (3 Marks)

Priority (Non-preemptive) Scheduling:

Waiting Time Calculation:

- *P1: Turnaround time = 19 - 0 = 19, Waiting time = Turnaround time - Burst time = 19 - 8 = 11.*
- *P2: Turnaround time = 6 - 0 = 6, Waiting time = Turnaround time - Burst time = 6 - 6 = 0.*
- *P3: Turnaround time = 7 - 0 = 7, Waiting time = Turnaround time - Burst time = 7 - 1 = 6.*
- *P4: Turnaround time = 19 - 0 = 19, Waiting time = Turnaround time - Burst time = 19 - 9 = 10.*
- *P5: Turnaround time = 10 - 0 = 10, Waiting time = Turnaround time - Burst time = 10 - 3 = 7.*

Average Waiting Time = (11 + 0 + 6 + 10 + 7) / 5 = 6.8 ms

Round-Robin (RR) Scheduling (Quantum = 2):

- *P1: Initial burst = 8, needs 4 turns in total (starts first).*
- *P2: Initial burst = 6, needs 3 turns.*
- *P3: Initial burst = 1, needs 1 turn.*
- *P4: Initial burst = 9, needs 5 turns.*
- *P5: Initial burst = 3, needs 2 turns.*

Gantt Chart for RR (Quantum = 2):

Time	*P1*	*P2*	*P3*	*P4*	*P5*
0	*P1*				
2	*P1*	*P2*			
4	*P2*	*P3*			
5	*P3*	*P4*			
7	*P4*	*P5*			
9	*P5*	*P1*			
11	*P1*	*P2*			

Time P1 P2 P3 P4 P5

13 P2 P4

15 P4 P1

17 P1 P2

19 P2 P4

Waiting Time Calculation for Round-Robin:

- *P1: Waiting time = (9 - 0) - 8 = 1.*
- *P2: Waiting time = (13 - 0) - 6 = 7.*
- *P3: Waiting time = (5 - 0) - 1 = 4.*
- *P4: Waiting time = (19 - 0) - 9 = 10.*
- *P5: Waiting time = (7 - 0) - 3 = 4.*

Average Waiting Time for Round-Robin = (1 + 7 + 4 + 10 + 4) / 5 = 5.2 ms.

---

2. (c) (iii) What is the average turn-around time of each scheduling algorithm? (3 Marks)

Priority (Non-preemptive) Scheduling:

Turnaround Time Calculation:

- *P1: Turnaround time = 19 - 0 = 19.*
- *P2: Turnaround time = 6 - 0 = 6.*
- *P3: Turnaround time = 7 - 0 = 7.*
- *P4: Turnaround time = 19 - 0 = 19.*
- *P5: Turnaround time = 10 - 0 = 10.*

Average Turnaround Time = (19 + 6 + 7 + 19 + 10) / 5 = 12.2 ms.

Round-Robin (RR) Scheduling:

Turnaround Time Calculation for Round-Robin:

- *P1: Turnaround time = 17 - 0 = 17.*
- *P2: Turnaround time = 13 - 0 = 13.*
- *P3: Turnaround time = 5 - 0 = 5.*
- *P4: Turnaround time = 19 - 0 = 19.*
- *P5: Turnaround time = 7 - 0 = 7.*

Average Turnaround Time for Round-Robin = (17 + 13 + 5 + 19 + 7) / 5 = 12.2 ms.

---

3. (a) (i) Differentiate between Cooperative and Independent Processes. (5 Marks)

- *Cooperative Processes:*
  - *These processes interact with each other and may share resources like memory or files.*
  - *One process may have to wait for the other to complete before continuing (e.g., producer-consumer model).*
  - *They rely on synchronization mechanisms like semaphores or locks.*
- *Independent Processes:*
  - *These processes do not interact with each other and do not share resources.*
  - *Each process runs independently, and the completion of one does not affect the others.*
  - *They do not require synchronization or mutual exclusion.*

---

3. (a) (ii) Race Condition (Explain with the help of an example). (5 Marks)

A race condition occurs when two or more processes attempt to modify shared data concurrently, and the final outcome depends on the non-deterministic order in which the processes execute.

Example: Consider two processes, A and B, both increment a shared variable x:

- *Initial value of x = 0*
- *Process A: x = x + 1*
- *Process B: x = x + 1*

If both processes execute simultaneously:

1. *Both read the value of x as 0.*

2. *Process A increments x to 1.*

3. *Process B increments x to 1 (instead of 2).*

The final value of x is 1 instead of 2, causing an incorrect result. This is a race condition because the outcome depends on the order of execution.

---

3. (b) What is Critical Section Problem? State Readers-Writers Problem and give a solution. Also show that the given solution satisfies all the requirements of a solution for the Critical Section Problem. (5 Marks)

- *Critical Section Problem: The critical section problem occurs when multiple processes need to access shared resources (like memory, file system) and require synchronization to avoid conflicts (such as race conditions) when accessing or modifying shared data.*

Requirements for a solution to the Critical Section Problem:

1. *Mutual Exclusion: Only one process can be in the critical section at a time.*

2. *Progress: If no process is in the critical section, one of the waiting processes must be able to enter.*

3. *Bounded Waiting: A process should not have to wait forever to enter the critical section.*

- *Readers-Writers Problem: In the readers-writers problem, we have two types of processes:*

  - *Readers: Processes that read from a shared resource.*

  - *Writers: Processes that write to a shared resource.*

The problem is to allow multiple readers to access the resource simultaneously but to prevent writers from writing when a reader is accessing it.

Solution (using semaphores):

  - *Use a read semaphore and a write semaphore.*

- *Readers acquire the read semaphore and access the resource. Multiple readers can do this simultaneously.*
- *Writers acquire the write semaphore to ensure that only one writer can write at a time and no reader is reading.*

This solution ensures:

- *Mutual Exclusion: Ensures that only one writer can write at a time.*
- *Progress: If no writer is writing, multiple readers can read simultaneously.*
- *Bounded Waiting: Writers may have to wait, but the waiting time is bounded.*

---

3. (c) Consider the following snapshot of a system:

	Allocation	*Max*	*Available*
P1:	*(1,2)*	*(2,1)*	*(3,3)*
P2:	*(1,0)*	*(3,3)*	*(2,2)*
P3:	*(1,2)*	*(1,0)*	*(3,1)*

(i) What is the content of matrix Need?

The Need matrix is calculated by subtracting the Allocation matrix from the Max matrix.
So, Need = Max - Allocation.

P1	*P2*	*P3*
(1,0)	*(2,3)*	*(0,0)*

(ii) Is the system in a safe state? If no, explain the reason. If yes, give the safe sequence.

To check if the system is in a safe state, we can apply the Banker's Algorithm.
The Available resources are (3,3).

After checking various combinations, we find that we can safely complete all processes in this order:
Safe Sequence = P3 → P1 → P2.

(iii) If a process from P1 arrives for (1, 1, 0, 2), can the request be granted immediately?

To check if the request can be granted immediately, we compare the requested resources with the available resources.

- *The request is (1,1,0,2) from P1.*

- *Available resources = (3,3).*

4. Attempt any two parts of choice from (a), (b) and (c). (10x2=20 Marks)

(a) Explain the following terms:

(i) Logical and Physical Address Space (3 Marks)

- *Logical Address Space: This refers to the address space generated by the CPU during a program's execution. It is the set of addresses used by a program for accessing memory, often called virtual address space.*

- *Physical Address Space: This refers to the actual memory locations (RAM) in the computer's hardware. It represents the real physical memory installed in the system and is managed by the operating system.*

The key difference is that logical addresses are generated by programs, while physical addresses correspond to actual locations in memory.

(ii) Swapping (3 Marks)

- *Swapping is a memory management technique where processes are moved between main memory (RAM) and secondary storage (like a hard disk) to ensure that the CPU always has enough processes to execute. The process is swapped out of memory when it is not being executed and swapped back into memory when it is required again.*

Swapping allows better utilization of memory resources and enables the execution of more processes than the physical memory can accommodate.

(iii) Thrashing (2 Marks)

- *Thrashing occurs when the operating system spends the majority of its time swapping processes in and out of memory, rather than executing processes. This happens when the system is overburdened with processes and doesn't have enough memory to run them efficiently, resulting in a significant degradation of performance.*

- *In thrashing, the system's CPU utilization is high, but it fails to make meaningful progress on executing processes because too much time is spent on swapping.*

(iv) Demand Segmentation (2 Marks)

- *Demand Segmentation is a memory management technique where the operating system loads only the necessary segments (like code, data, etc.) of a program into memory when they are required during execution. This approach helps in reducing the memory load and allows the execution of large programs even if there isn't enough physical memory to load the entire program at once.*

(b) Consider the following page reference string:

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1
How many page faults will occur, if there are three frames and initially all are empty, using LRU, Optimal, and FIFO Page replacement? (5 Marks)

We have 3 page frames and we will calculate the number of page faults for LRU (Least Recently Used), Optimal, and FIFO (First In First Out) page replacement algorithms.

(i) LRU Page Replacement:

1. *Initial reference string: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1*

2. For LRU, the least recently used page is replaced when a page fault occurs.

Page fault count:

- *Pages loaded into frames are replaced based on the least recently used page. As we go through the reference string, we get:*
  - *7, 0, 1 → 3 page faults.*
  - *2 → 1 page fault (replace 7).*
  - *0 → 0 page fault (already in memory).*
  - *3 → 1 page fault (replace 1).*
  - *0 → 0 page fault (already in memory).*
  - *4 → 1 page fault (replace 2).*
  - *2 → 1 page fault (replace 3).*
  - *And so on.*

Total page faults for LRU: 15 page faults.

(ii) Optimal Page Replacement:

The Optimal page replacement algorithm replaces the page that will not be used for the longest period in the future.

Page fault count:

- *Pages are replaced based on future references. For example, when the page 7 is referenced and not in memory, it will be loaded into the available frame. The next page replacement decision will always select the page that will not be used again for the longest time.*

Total page faults for Optimal: 12 page faults.

(iii) FIFO (First-In-First-Out) Page Replacement:

The FIFO algorithm replaces the oldest page (the first one that was loaded into memory) when a new page needs to be loaded.

Page fault count:

- *Pages are replaced in the order they are loaded. As the reference string progresses, the first page that was loaded will be replaced when a new page comes in.*

Total page faults for FIFO: 16 page faults.

(c) Given five memory partitions of 100 Kb, 500 Kb, 200 Kb, 300 Kb, 600 Kb (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of 222 Kb, 419 Kb, 110 Kb, and 420 Kb (in order)? Which algorithm makes the most efficient use of memory? (5 Marks)

Let's see how the First-fit, Best-fit, and Worst-fit algorithms would allocate memory.

Available partitions:

- *100 Kb, 500 Kb, 200 Kb, 300 Kb, 600 Kb*

Processes:

- *222 Kb, 419 Kb, 110 Kb, 420 Kb*

(i) First-Fit Allocation:

- *222 Kb: Fits in 500 Kb → allocated to 500 Kb.*

- *419 Kb: Fits in 600 Kb → allocated to 600 Kb.*
- *110 Kb: Fits in 200 Kb → allocated to 200 Kb.*
- *420 Kb: Cannot fit in any remaining partition → cannot be allocated.*

(ii) Best-Fit Allocation:

- *222 Kb: Fits in 300 Kb → allocated to 300 Kb (smallest remaining partition).*
- *419 Kb: Fits in 500 Kb → allocated to 500 Kb (smallest remaining partition).*
- *110 Kb: Fits in 200 Kb → allocated to 200 Kb (smallest remaining partition).*
- *420 Kb: Cannot fit in any remaining partition → cannot be allocated.*

(iii) Worst-Fit Allocation:

- *222 Kb: Fits in 600 Kb → allocated to 600 Kb (largest partition).*
- *419 Kb: Fits in 500 Kb → allocated to 500 Kb (largest remaining partition).*
- *110 Kb: Fits in 300 Kb → allocated to 300 Kb (largest remaining partition).*
- *420 Kb: Cannot fit in any remaining partition → cannot be allocated.*

Most Efficient Memory Use:

- *Best-Fit is the most efficient in terms of memory usage because it tries to fit the processes into the smallest available partition, leaving fewer and more evenly distributed unused spaces.*

---

5. Attempt any two parts of choice from (a), (b) and (c). (10x2=20 Marks)

(a) Explain all types of directory structures using neat and labelled diagrams. (5 Marks)

Types of Directory Structures:

1. *Single-Level Directory:*

   - *A single directory contains all files, and there is no hierarchy.*
   - *Diagram:*
   - *Directory: /*
   - *Files: File1, File2, File3, File4*

2. *Two-Level Directory:*

   - *Each user has a separate directory under the root directory.*
   - *Diagram:*
   - */*
   - *|-- User1*
   - *| |-- File1*

- *| |-- File2*
- *|*
- *|-- User2*
- *|-- File3*
- *|-- File4*

3. *Tree-Structured Directory:*

- *Directories are organized in a hierarchical tree structure, with subdirectories under each directory.*
- *Diagram:*
- */*
- *|-- Home*
- *| |-- User1*
- *| | |-- File1*
- *| |*
- *| |-- User2*
- *| |-- File2*
- *|*
- *|-- Bin*
- *|-- File3*

4. Acyclic-Graph Directory:

- *Allows directories to have shared subdirectories or files, meaning a file or subdirectory can be accessed via multiple paths.*

- *Diagram:*

- */*

- */-- Home*

- */ /-- User1*

- */ /-- User2*

- */*

- */-- Bin*

- */-- File1*

(b) Suppose that a disk drive has 5000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 135, and the previous request was at cylinder 125. The queue of pending requests, in FIFO order, is:

86, 1470, 913, 1774, 948, 1509, 1022, 1730, 130
Calculate the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms:

(i) LOOK
(ii) SCAN
(iii) FCFS
(iv) C-LOOK (10 Marks)

We can calculate the total distance for each algorithm as follows:

Given Disk Queue:
Pending requests = 86, 1470, 913, 1774, 948, 1509, 1022, 1730, 130
Initial position = 135
Previous position = 125

For each of the algorithms, we'll need to calculate how the disk arm moves and track the total distance traveled, taking into account the movement direction and order of processing the requests.

---

i) LOOK Scheduling:

- *LOOK moves the disk arm in the direction of the requested track and processes all requests in that direction, then reverses direction at the end of the request queue.*

- *We'll need to calculate the total movement based on the current direction (to the right of 135) and process the requests in that direction first.*

Let's continue and finish the remaining calculations for disk-scheduling algorithms in question 5(b):

(b) Disk Scheduling Algorithms

We are given the following information:

- *Disk drive with 5000 cylinders (numbered 0 to 4999).*

- *Current position of the disk head: 135.*

- *Pending requests: 86, 1470, 913, 1774, 948, 1509, 1022, 1730, 130.*

- *Previous request: 125.*

Now, we need to calculate the total distance the disk arm moves to satisfy all the pending requests for different disk scheduling algorithms:

---

i) LOOK Scheduling (10 Marks)

In the LOOK scheduling algorithm, the disk arm moves in the direction of the closest request, but once it reaches the end of that direction (either towards the lower-numbered cylinders or the higher-numbered cylinders), it reverses direction.

1. *Sort the pending requests in ascending order:*

   - *Sorted requests: 86, 130, 913, 948, 1022, 1470, 1509, 1730, 1774.*

2. *Direction of the arm: Since the previous request was at cylinder 125 and the current request is at 135, the disk arm will first move to the right (towards higher-numbered cylinders).*

3. *Process the requests: Start from 135 and process the requests in the rightward direction (until we reach the maximum request in that direction) and then reverse direction to process the remaining requests.*

   - *Move from 135 to 1774 (right): Distance = 1774 - 135 = 1639.*

   - *Reverse direction to 86 (left): Distance = 1774 - 86 = 1688.*

4. *Total movement distance for LOOK:*

   - *Total = 1639 (right) + 1688 (left) = 3327 cylinders.*

---

ii) SCAN Scheduling (10 Marks)

In SCAN scheduling, the disk arm moves in one direction (towards the end of the disk) to fulfill requests, and when it reaches the last request, it reverses direction. The arm moves across the entire disk (from the current position to the highest or lowest request), satisfying requests as it goes.

1. *Sort the pending requests in ascending order:*

   - *Sorted requests: 86, 130, 913, 948, 1022, 1470, 1509, 1730, 1774.*

2. *Direction of the arm: The disk arm will move towards the higher-numbered cylinders, as the current position is 135 and the previous request was 125.*

3. *Process the requests:*

- *Move from 135 to 1774 (right) (first the requests towards higher cylinders).*
- *When the arm reaches 1774, it will reverse direction and start serving the requests towards the lower-numbered cylinders.*

4. *Distance:*

- *Move from 135 to 1774: Distance = 1774 - 135 = 1639.*
- *Reverse direction from 1774 to 86: Distance = 1774 - 86 = 1688.*

5. *Total movement distance for SCAN:*

- *Total = 1639 (right) + 1688 (left) = 3327 cylinders.*

---

iii) FCFS (First-Come-First-Served) Scheduling (10 Marks)

In FCFS scheduling, the disk arm moves in the order that the requests are received, without reordering.

1. *Requests in FIFO order: 86, 1470, 913, 1774, 948, 1509, 1022, 1730, 130.*

2. *Distance moved by the disk arm:*

- *Move from 135 to 86: Distance = 135 - 86 = 49.*
- *Move from 86 to 1470: Distance = 1470 - 86 = 1384.*
- *Move from 1470 to 913: Distance = 1470 - 913 = 557.*

- *Move from 913 to 1774: Distance = 1774 - 913 = 861.*
- *Move from 1774 to 948: Distance = 1774 - 948 = 826.*
- *Move from 948 to 1509: Distance = 1509 - 948 = 561.*
- *Move from 1509 to 1022: Distance = 1509 - 1022 = 487.*
- *Move from 1022 to 1730: Distance = 1730 - 1022 = 708.*
- *Move from 1730 to 130: Distance = 1730 - 130 = 1600.*

3. *Total movement distance for FCFS:*

- *Total = 49 + 1384 + 557 + 861 + 826 + 561 + 487 + 708 + 1600 = 7533 cylinders.*

---

iv) C-LOOK Scheduling (10 Marks)

In C-LOOK scheduling, the disk arm moves in one direction (towards the higher-numbered cylinders), but instead of reversing direction when it reaches the last request, it moves back to the first request and continues.

1. *Sort the pending requests in ascending order:*

- *Sorted requests: 86, 130, 913, 948, 1022, 1470, 1509, 1730, 1774.*

2. *Direction of the arm: The disk arm starts at 135 and moves towards the higher-numbered cylinders.*

3. *Process the requests:*

- *Move from 135 to 1774 (right) and satisfy requests.*
- *Once 1774 is reached, the arm jumps back to 86 (the lowest pending request) and moves back rightward.*

4. *Distance:*

- *Move from 135 to 1774: Distance = 1774 - 135 = 1639.*
- *Move from 1774 to 86 (wrap around): Distance = 1774 - 86 = 1688.*

5. *Total movement distance for C-LOOK:*

- *Total = 1639 (right) + 1688 (wrap around left) = 3327 cylinders.*

Summary of Results:

Algorithm	*Total Distance (in cylinders)*
LOOK	*3327*
SCAN	*3327*
FCFS	*7533*
C-LOOK	*3327*

From the table above, we can observe that LOOK, SCAN, and C-LOOK are more efficient than FCFS since they minimize the distance the disk arm travels. Among these, LOOK, SCAN, and C-LOOK have the same performance.

Let's check the questions that are remaining and the ones that are already answered. I'll address the remaining ones.

Remaining Questions

Q2. Attempt any two parts of choice from (a), (b) and (c).

a) Write short notes on the following:

- *(i) Process state*
- *(ii) Context switch*

Q3. Attempt any two parts of choice from (a), (b), and (c).

a) Answer the following:

- *(i) Differentiate between Cooperative and Independent processes.*
- *(ii) Race condition (Explain with the help of an example).*

b) What is Critical Section problem? State the Readers-Writers problem and give a solution. Also, show that the given solution satisfies all the requirements of a solution for the critical section problem.

Let's solve these questions now.

Q2. a) Short Notes

(i) Process State

A process goes through various states during its execution. The common process states are:

- *New: The process is being created.*
- *Ready: The process is loaded into memory and is ready to execute but is waiting for CPU time.*
- *Running: The process is currently being executed on the CPU.*

- *Blocked (Waiting): The process cannot execute because it is waiting for some event (like I/O completion).*
- *Terminated: The process has finished execution and has been removed from the system.*

The state transitions happen in the following order:

- *A process moves from New to Ready when it is loaded into memory.*
- *It moves from Ready to Running when it is assigned CPU time.*
- *If a process needs I/O or is waiting for an event, it moves to the Blocked state.*
- *Finally, it terminates after completing its execution, transitioning to the Terminated state.*

(ii) Context Switch

A context switch occurs when the CPU switches from one process to another. It involves saving the state (context) of the current process (like registers, program counter) and loading the state of the next process to be executed.

Steps involved:

1. *Save the context (program counter, registers, etc.) of the currently running process.*
2. *Load the context of the next process to execute.*
3. *Resume the execution of the new process from where it was last suspended.*

Context switching is a necessary feature in multitasking systems, allowing multiple processes to share CPU time.

Q3. a) Answer the following

(i) Cooperative vs. Independent Processes

- *Cooperative Processes:*
  - *A cooperative process is one that can be affected by or interact with other processes.*
  - *These processes share resources, and their behavior can depend on the actions of other processes.*
  - *Example: Thread synchronization in multithreading where one thread waits for another.*
  - *In cooperative multitasking, processes voluntarily give up control, allowing other processes to run.*
- *Independent Processes:*
  - *An independent process does not share resources or affect other processes.*
  - *These processes do not need to communicate or synchronize with other processes.*
  - *Example: A process that does not need interaction with other processes, like a standalone program.*
  - *In a system with independent processes, they run without causing side effects on each other.*

(ii) Race Condition (Explain with an example)

A race condition occurs when two or more processes or threads attempt to access shared resources concurrently, and the final outcome depends on the sequence or timing of their execution.

Example:

- *Assume two threads T1 and T2, which both increment a shared counter counter.*
- *T1 reads the value of counter (say it's 5), then T2 reads the same value (also 5).*

- *Both increment the value by 1, but without synchronization, both might set counter to 6 instead of 7, causing incorrect behavior.*

Solution to Race Condition:

- *Locks or mutexes are used to ensure only one process can access the shared resource at a time, preventing race conditions.*

Q3. b) Critical Section Problem & Readers-Writers Problem

Critical Section Problem:

- *The Critical Section Problem refers to a situation in concurrent programming where multiple processes or threads try to access shared resources (data or hardware) simultaneously. To avoid inconsistencies, only one process can access the shared resource at a time.*

Requirements for solving Critical Section Problem:

1. *Mutual Exclusion: Only one process can be in the critical section at a time.*
2. *Progress: If no process is in the critical section, and multiple processes want to enter, the selection of the next process should not be postponed indefinitely.*
3. *Bounded Waiting: There must be a limit on the number of times a process is allowed to enter the critical section after requesting.*

Readers-Writers Problem:

- *In the Readers-Writers Problem, there are two types of processes: readers and writers. A reader can access the shared resource (data) simultaneously with other readers, but a writer requires exclusive access.*

Solution: A typical solution uses semaphores to ensure mutual exclusion and priority conditions to avoid starvation of readers or writers.

- *Readers are allowed to access the shared data concurrently, but if a writer is already accessing the data, they must wait.*
- *Similarly, writers must wait if any reader is currently accessing the data, ensuring mutual exclusion.*