Roll No ....

TCS-502

B. TECH. (CSE)

(FIFTH SEMES'TER) END SEMESTER

EXAMINATION, Jan.; 2023

OPERATING SYSTEM

Time : Three Hours

Maximum Marks : 100

Note : **All questions are compulsory.**

(ii) Answer any two sub-questions among (a), (b) and (c) in each main question.

(iii) Total marks in each main question are twenty.

(iv) Each sub-question carries 10 marks.

1. (a) What is 'system call and its various types ? Alsoiexplain its dual mode of operation.

(COI)

P. T. O.

(b) Explain the following : (C01)

(0 Batch Operating System

(ii) Distributed Operating System

(iii) Multiprocessing Operating system

(iv) Real Time Operating System

(c) Explain -various components of an operating system in detail. (C01)

2.. (a) Explain Shortest Remaining Time First (SRTF) CPU Scheduling algorithm. Consider the processes, CPU burst time and arrival time given below : (CO2)

Processes	CPU Burst Time. (ns)	Arrival Time
P_1	8	2
P2	1	
P3	2	
P4	6	
P5	4	[illegible]
P6.	2	[illegible]

Draw the Gantt chart and calculate the following by using SRTF and Round Robin (Time Quantum = 2 ns) CPU scheduling algorithms: :

(i) Average waiting time

(ii) Average turnaround time

(iii) CPU utilization

(iv). CPU idleness

(b) What is Critical Section Problem ? State Reader-Writer problem and describe its solution using semaphores. (CO2)

(c) Explain process state diagram with the help of a suitable diagram. (CO2)

3. (a) Explain paging scheme using tran:slation look aside buffer (TLB). In a paging scheme using TLB, the TLB access takes 10 ns and a main memorraccess takes 50 ns. What is the effective , access time (in ns) if the TLB hit ratio is 90% and there is no page fault. (CO3)

(b) Consider the following snapshot of a system : (CO3)

	Allocation				Max-				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
Po	2	0	0	1	4	.2	1	2	3	3	2	.1
P_1	3	1	2	1.	5	2	5	2				
P2	2	1	0	3	2	3	1	6				
P3	1	3'	1	2	1	4	2	4				
P4	1	4	3	2	3	6	6	5				

Answer the following questions using the

- Banker's algorithm

(i) Illustrate that the system is in a safe state by demonstrating an order in which the processes may complete.

- (ii) If a request from process P_1 arrives* for (1, 1, 0, 0), can the request be granted immediately ?

(iii) If a request from -process P4 arrives for (0, 0, 2, 0), can_ the request be granted immediately ?

(c) Given free memory partitions of 100 K, 500 K, 200 K; 300 K and 600. K (in order), hoW would each of the First-fit, Best-fit and Worst fit algorithms place processes of 212 K, 417 K; 112 K and 426 K (in order) ? (CO3)

4. (a) Describe the most common schemes for defining the logical structure of a directory. (C04)

(b) Explain contiguous and linked allocation methods of file storage. (C04)

(c) A disk drive has 200 cylinders, numbered 0 to 199.- The drive is currently serving a request at cylinder 53. The queue of pending requests, f in FIFO order, is 98·, 183, 37, 122, 14, 124, 65, 67. Starting from the current head position,' what is the total distance (in 'cylinders) that the disk arm moves to satisfy all the pending

1.SOL- 1. (a) What is a system call and its various types? Also, explain its dual mode of operation.

System Call: A system call is a request made by a program to the operating system's kernel for performing operations that are restricted or unavailable in user space, such as accessing hardware, managing files, or controlling processes. It provides an interface between user programs and the underlying hardware via the operating system.

When a program requires services such as I/O operations, memory management, or process management, it cannot directly communicate with the hardware. Hence, system calls act as a bridge between the user and kernel, allowing safe and controlled access to these services.

Types of System Calls: System calls can be categorized based on their function:

1. **Process Control:**
   - **Used for managing processes.**
   - **Examples: fork(), exit(), wait(), exec()**
   - **These system calls handle process creation, termination, and process synchronization.**
2. **File Management:**
   - **Used for file operations such as opening, reading, writing, and closing files.**
   - **Examples: open(), read(), write(), close(), unlink()**
   - **They help in file creation, deletion, and manipulation.**
3. **Device Management:**

   - **Deals with input/output operations and device management.**
   - **Examples: ioctl(), read(), write()**
   - **These system calls allow programs to interact with hardware devices like printers, hard drives, etc.**

4. **Information Maintenance:**
   - **Used for obtaining and setting information about system resources and processes.**
   - **Examples: getpid(), setpriority(), gettimeofday()**
   - **They deal with information like the process ID, system time, and resource utilization.**

5. **Communication:**

- **Involves communication between processes or between the user and the system.**
- **Examples: pipe(), shmget(), msgget(), recv(), send()**
- **These allow inter-process communication (IPC) through messages, shared memory, or signals.**

Dual Mode of Operation:

The dual mode of operation refers to the mechanism by which the system operates in two distinct modes: user mode and kernel mode.

1. **User Mode:**
    - **In this mode, user programs and applications execute.**

    - **User code cannot directly access hardware or critical OS resources, ensuring that one user's program cannot affect the entire system.**

2. **Kernel Mode:**
    - **In kernel mode, the operating system's kernel executes.**
    - **The kernel has unrestricted access to the hardware, memory, and all system resources.**
    - **When a system call is made, the CPU switches from user mode to kernel mode, allowing the operating system to perform the requested task on behalf of the user program.**

Why Dual Mode?

- **Dual mode ensures that user programs cannot interfere with each other or with the** operating system itself, providing security and stability.

- The switch between these modes is done through a mechanism called a context switch.

1. (b) Explain the following:

(i) Batch Operating System: A batch operating system is a system that executes jobs without interactive user input. It groups similar jobs together and processes them in batches, one after the other. Users prepare their jobs in advance and submit them for processing.

- Advantages:
  - Efficient for processing large amounts of data.
  - Automated job scheduling, which reduces human intervention.
- Disadvantages:

- Lack of interactivity (users cannot modify a running job).
- No immediate feedback or result.

(ii) Distributed Operating System: A distributed operating system manages a collection of independent computers (nodes) that appear as a single system to users. These nodes are connected over a network and share resources such as memory and processing power.

- **Advantages:**
  - **Scalability and fault tolerance.**
  - **Resource sharing across multiple systems.**
- **Disadvantages:**
  - **Complex communication and synchronization between nodes.**

  - **Security and reliability issues in a networked environment.**

(iii) Multiprocessing Operating System: A multiprocessing operating system is designed to support the simultaneous execution of multiple processes across multiple CPUs or processors. It can execute multiple programs or tasks in parallel, improving system performance.

- **Advantages:**
  - **Increased throughput and performance.**
  - **Improved fault tolerance, as tasks can be distributed across processors.**
- **Disadvantages:**
  - **Complex management of resources.**
  - **Synchronization issues between processes.**

(iv) Real-Time Operating System (RTOS): An RTOS is designed for applications that require immediate processing and response within a specified time frame. It ensures that tasks are completed within a strict time limit, which is crucial for time-sensitive applications such as embedded systems, robotics, and air traffic control.

- **Advantages:**
  - **Deterministic response times.**
  - **Predictability in task scheduling.**
- **Disadvantages:**
  - **Limited processing power.**
  - **Complexity in managing time constraints and resource allocation.**

---

1. (c) Explain various components of an operating system in detail.

The operating system (OS) is a complex software layer that manages hardware resources and provides essential services to software applications. It has several key components:

1. **Kernel:**
    - **The core component of the OS that operates in kernel mode and has full access to hardware.**
    - **It manages resources like memory, CPU, and input/output devices.**
    - **Handles process scheduling, memory management, and system calls.**
2. **Process Management:**
    - **Manages processes, which are instances of executing programs.**
    - **Includes creating, scheduling, and terminating processes.**

- **Provides mechanisms for process synchronization and inter-process communication (IPC).**

3. **Memory Management:**
   - **Responsible for allocating and deallocating memory for processes.**
   - **Ensures that each process gets its required memory space while preventing access to other processes' memory.**
   - **Includes techniques like paging, segmentation, and virtual memory.**
4. **File System:**
   - **Manages the storage and retrieval of files.**
   - **Organizes data in directories and provides a way to access files using file paths.**

- **Includes file operations such as reading, writing, opening, closing, and deleting files.**

5. **Device Management:**
   - **Manages hardware devices such as printers, disk drives, and displays.**
   - **Provides device drivers that allow user programs to interact with hardware through system calls.**
6. **Security and Access Control:**
   - **Ensures the security of data and resources through user authentication, encryption, and access control mechanisms.**
   - **Protects against unauthorized access and ensures system integrity.**
7. **User Interface:**

- Provides a user interface (UI) for interacting with the system.
- Can be a command-line interface (CLI) or a graphical user interface (GUI), depending on the OS.
- Facilitates user input and system output.

8. **Networking:**
    - Provides services for communication between devices over a network.
    - Manages network connections, protocols, and data transmission.
    - Includes features for client-server communication, resource sharing, and distributed computing.
9. **System Utilities:**

- **A set of tools and utilities that help manage and maintain the operating system.**
- **Examples: file management tools, disk utilities, backup software, and performance monitoring tools.**

Together, these components work in harmony to provide a stable, secure, and efficient environment for running programs and interacting with hardware.

2- (a) Shortest Remaining Time First (SRTF) CPU Scheduling Algorithm

SRTF (Shortest Remaining Time First) is a preemptive scheduling algorithm that is a variant of the Shortest Job First (SJF) algorithm. It selects the process with the shortest remaining CPU burst time to execute next, preempting the currently running process if a new process arrives with a shorter remaining burst time.

Given Processes:

Process	**CPU Burst Time (ns)**	**Arrival Time (ns)**
P1	**8**	**2**
P2	**1**	**7**
P3	**2**	**6**
P4	**6**	**3**
P5	**4**	**5**
P6	**2**	**3**

Step-by-Step Execution using SRTF

We will now calculate the Gantt chart for the SRTF scheduling algorithm based on

the processes' arrival and CPU burst times.

1. **At time 0, the CPU is idle because no processes have arrived yet.**
2. **At time 2, P1 arrives with a burst time of 8 ns. It starts executing.**
3. **At time 3, P4 arrives with a burst time of 6 ns. Since P4's remaining time (6 ns) is shorter than P1's remaining time (8 ns), P4 preempts P1.**
4. **At time 5, P5 arrives with a burst time of 4 ns. Since P5's remaining time (4 ns) is shorter than P4's remaining time (3 ns), P5 preempts P4.**
5. **At time 6, P3 arrives with a burst time of 2 ns. P3 has a shorter remaining burst time than P5, so P3 preempts P5.**

6. **At time 7, P2 arrives with a burst time of 1 ns, and it has the shortest burst time. It preempts P3 and executes next.**
7. **Continue this process, updating which process is running at each time slice, and noting when processes finish.**

Gantt Chart for SRTF:

Calculations for SRTF:

1. **Average Waiting Time (AWT):**
   - **Waiting time is calculated as: Waiting Time=Turnaround Time−CPU Burst Time\text{Waiting Time} = \text{Turnaround Time} - \text{CPU Burst Time}**
   - **We calculate the waiting time for each process and average it.**
2. **Average Turnaround Time (ATAT):**
   - **Turnaround time is calculated as: Turnaround Time=Completion Time−Arrival Time\text{Turnaround Time} = \text{Completion Time} - \text{Arrival Time}**
   - **We calculate the turnaround time for each process and average it.**
3. **CPU Utilization:**

- CPU utilization is the percentage of time the CPU is actively processing requests.
- CPU Utilization=Time CPU is BusyTotal Time×100\text{CPU Utilization} = \frac{\text{Time CPU is Busy}}{\text{Total Time}} \times 100

4. **CPU Idleness:**

- CPU idleness is the percentage of time the CPU is not executing any process.
- CPU Idleness=100%−CPU Utilization\text{CPU Idleness} = 100\% - \text{CPU Utilization}

Round Robin (Time Quantum = 2 ns)

Now, let's calculate the Gantt chart and other metrics using the Round Robin (RR) scheduling algorithm with a time quantum of 2 ns.

Process Execution:

1. **Processes will execute for 2 ns in each time slice. After 2 ns, they will either finish or be preempted and placed back in the ready queue.**
2. **This continues until all processes are finished.**

Gantt Chart for Round Robin:

Calculations for Round Robin:

Same as the SRTF method, we calculate:

- **Average Waiting Time (AWT)**
- **Average Turnaround Time (ATAT)**
- **CPU Utilization**
- **CPU Idleness**

(b) Critical Section Problem

The Critical Section Problem refers to the challenge of ensuring that multiple processes that need to access shared resources do so in a way that avoids conflicts or inconsistent results. The problem involves coordinating access to shared data to prevent situations where multiple processes attempt to modify the same resource simultaneously.

Reader-Writer Problem:

The Reader-Writer Problem is a specific type of synchronization problem in which:

- **Readers can access the shared resource concurrently, provided no writers are modifying it.**
- **Writers require exclusive access to the resource, and no other readers or writers should be allowed while a write operation is in progress.**

Solution using Semaphores:

We use semaphores to ensure that:

- **Readers can access the resource concurrently as long as no writers are accessing it.**
- **Writers gain exclusive access to the resource.**

Semaphores:

1. **mutex (binary semaphore) is used to ensure mutual exclusion for accessing the shared resource.**

2. readCount (integer semaphore) is used to count the number of readers accessing the resource.
3. writeSemaphore (binary semaphore) is used to control the writer's access to the resource.

Reader Process:

```
wait(mutex);
readCount++;  // Increment the number of readers
if (readCount == 1) {
   wait(writeSemaphore);   // First reader blocks writers
}
signal(mutex);


// Perform reading...
```

```
wait(mutex);
readCount--;   // Decrease the number of readers
if (readCount == 0) {
   signal(writeSemaphore);  // Last reader allows writers
}
signal(mutex);
```

Writer Process:

```
wait(writeSemaphore);    // Writer gains exclusive access to the resource


// Perform writing...


signal(writeSemaphore);    // Release the resource after writing
```

(c) Process State Diagram

A Process State Diagram visually represents the states a process can be in during its lifecycle. The typical process states are:

1. **New: The process is being created.**
2. **Ready: The process is ready to execute but is waiting for CPU time.**
3. **Running: The process is currently being executed by the CPU.**
4. **Waiting (Blocked): The process is waiting for some event, like I/O completion.**
5. **Terminated: The process has finished execution.**

Process State Diagram:

```
+---------+
| New     |
+----+----+
     |
     v
+----+----+
| Ready   |
+----+----+
     |
+----+----+     +---------+
| Running  |<-->| Waiting |
+----+----+     +----+----+
     |               ^
     v               |
+----+----+     +----+----+
| Terminated|<----| Event   |
+----------+     +---------+
```

- A process starts in the New state.

- **It moves to the Ready state once it is loaded into memory and ready for execution.**
- **From the Ready state, it moves to Running when the CPU scheduler picks it.**
- **If the process needs to wait (e.g., for I/O), it transitions to the Waiting state.**
- **When the process completes its execution, it enters the Terminated state.**

This state diagram helps visualize how processes transition between states based on system events.

3. (a) Paging Scheme using Translation Lookaside Buffer (TLB)

In a paging scheme, the virtual memory is divided into fixed-size pages, and the physical memory is divided into fixed-size frames. When a program accesses a memory location, the operating system

maps the virtual page number to a physical frame number using a page table.

The Translation Lookaside Buffer (TLB) is a small, fast cache that stores a subset of the most recently used page table entries. The TLB speeds up memory access by reducing the time required for address translation. When a memory address is accessed, the TLB is checked first to see if the translation is already cached.

Access Process:

- **TLB Hit: If the page table entry is found in the TLB, the virtual address can be translated directly without accessing the page table in memory.**
- **TLB Miss: If the page table entry is not found in the TLB, the system has to access the page table in main memory to get the physical frame number.**

In this problem:

- **TLB Access Time = 10 ns**
- **Main Memory Access Time = 50 ns**
- **TLB Hit Ratio = 90% (0.9)**
- **No Page Fault (i.e., all pages are present in memory, so there's no need to load a page from disk)**

Effective Access Time (EAT) Formula:

The Effective Access Time (EAT) is calculated by considering both TLB hits and misses:

EAT=(TLB hit ratio×TLB access time)+(TLB miss ratio×(TLB access time+2×Main memory access time))\text{EAT} = (\text{TLB hit ratio} \times \text{TLB access time}) + (\text{TLB miss ratio} \times (\text{TLB access time} + 2 \times \text{Main memory access time}))

Where:

- **TLB hit ratio = 0.9**
- **TLB miss ratio = 1 - 0.9 = 0.1**
- **TLB access time = 10 ns**
- **Main memory access time = 50 ns**

Calculation:

EAT=(0.9×10)+(0.1×(10+2×50))\text{EAT } = (0.9 \times 10) + (0.1 \times (10 + 2 \times 50)) EAT=9+(0.1×(10+100))\text{EAT} = 9 + (0.1 \times (10 + 100)) EAT=9+(0.1×110)\text{EAT} = 9 + (0.1 \times 110) EAT=9+11=20 ns\text{EAT} = 9 + 11 = 20 \, \text{ns}

So, the Effective Access Time (EAT) is 20 ns.

(b) Banker's Algorithm and Safe State

In this problem, we are given the Allocation, Maximum, and Available resources for each process in the system. Using Banker's Algorithm, we can determine if the system is in a safe state and whether resource requests can be granted immediately.

Given:

Process	**Allocation**	**Maximum**	**Available**
	A B C D	**A B C D**	**A B C D**

Process	Allocation	Maximum	Available
P0	2 0 0 1	4 2 1 2	3 3 2 1
P1	3 1 2 1	5 2 5 2	
P2	2 1 0 3	2 3 1 6	

Process	Allocation	Maximum	Available
P3	1 3 1 2	1 4 2 4	
P4	1 4 3 2	3 6 6 5	

Banker's Algorithm Steps:

1. **Calculate the Need Matrix: The Need matrix is calculated by subtracting**

the Allocation matrix from the Maximum matrix for each process:

Need=Maximum−Allocation\text{Need} = \text{Maximum} - \text{Allocation}

Need Matrix:

Process	**A B C D**
P0	**2 2 1 1**
P1	**2 1 3 1**
P2	**0 2 1 3**
P3	**0 1 1 2**
P4	**2 2 3 3**

2. **Check for Safe State:** **The system is in a safe state if there is a sequence of processes that can complete with the available resources. We use the Available vector and check if it is sufficient for a process to complete and release its resources.**

Initially, Available is: (3, 3, 2, 1).

We can check which process has all its needs less than or equal to the available resources.

- **P0: Needs (2, 2, 1, 1) which is less than or equal to (3, 3, 2, 1), so P0 can complete. After P0 finishes, it releases resources (2, 0, 0, 1), and the new available resources become (5, 3, 2, 2).**
- **P3: Needs (0, 1, 1, 2), which is less than or equal to (5, 3, 2, 2), so P3 can also complete. After P3 finishes, it releases resources (1, 3, 1, 2), and the new available resources become (6, 6, 3, 4).**
- **P1: Needs (2, 1, 3, 1), which is less than or equal to (6, 6, 3, 4), so P1 can complete. After P1 finishes, it releases resources (3, 1, 2, 1), and the new available resources become (9, 7, 5, 5).**

- **P2: Needs (0, 2, 1, 3), which is less than or equal to (9, 7, 5, 5), so P2 can complete. After P2 finishes, it releases resources (2, 1, 0, 3), and the new available resources become (11, 8, 5, 8).**
- **P4: Needs (2, 2, 3, 3), which is less than or equal to (11, 8, 5, 8), so P4 can complete.**

Since all processes can finish in sequence, the system is in a safe state. The safe sequence is: P0 → P3 → P1 → P2 → P4.

(ii) Can the request from Process P1 for (1, 1, 0, 0) be granted immediately?

To determine if the request from Process P1 can be granted, we check if:

- **The requested resources are less than or equal to the available resources.**

- **If granted, we check if the system remains in a safe state.**
- **Requested resources by P1: (1, 1, 0, 0)**
- **Available resources: (3, 3, 2, 1)**

  Since the request is less than or equal to the available resources, it can be granted.

  After granting the request, the new allocation for P1 will be:

- **New Allocation for P1: (3, 2, 2, 1)**

  The new available resources will be:

- **New Available: (2, 2, 2, 1)**

  Now, we need to check if the system remains in a safe state with these updated resources. Using the Banker's algorithm steps, we find that the system still remains in a safe state (safe sequence can still be established).

Thus, the request can be granted immediately.

(iii) Can the request from Process P4 for (0, 0, 2, 0) be granted immediately?

- Requested resources by P4: (0, 0, 2, 0)
- Available resources: (3, 3, 2, 1)

Since the request is less than or equal to the available resources, it can be granted.

After granting the request, the new allocation for P4 will be:

- New Allocation for P4: (1, 4, 5, 2)

The new available resources will be:

- New Available: (3, 3, 0, 1)

Now, we need to check if the system remains in a safe state with these updated resources. By following the Banker's

algorithm, we find that the system is still in a safe state.

Thus, the request can be granted immediately.

Sure! Here's a shorter version of the explanation:

Memory Allocation Using First-fit, Best-fit, and Worst-fit:

Given:

- **Free partitions: 100K, 500K, 200K, 300K, 600K.**
- **Process sizes: 212K, 417K, 112K, 426K.**

---

1. First-fit:

- **P1 (212K) → 500K (Partition 1)**
- **P2 (417K) → 600K (Partition 5)**
- **P3 (112K) → 200K (Partition 3)**

- **P4 (426K) → Cannot fit (no sufficient partition)**

---

2. Best-fit:

- **P1 (212K) → 300K (Partition 4)**
- **P2 (417K) → 500K (Partition 2)**
- **P3 (112K) → 200K (Partition 3)**
- **P4 (426K) → 600K (Partition 5)**

---

3. Worst-fit:

- **P1 (212K) → 600K (Partition 5)**
- **P2 (417K) → 500K (Partition 2)**
- **P3 (112K) → 388K (Partition 5)**
- **P4 (426K) → Cannot fit**

Let me know if you need further details!

4. (a) Common Schemes for Defining the Logical Structure of a Directory:

1. **Single-Level Directory:**
   - **All files are kept in one directory.**
   - **Simple but not scalable for large systems.**
   - **No hierarchical structure, so filenames must be unique.**
2. **Two-Level Directory:**
   - **One directory for each user, and files are listed within each user's directory.**
   - **Solves the issue of unique filenames but lacks flexibility.**
3. **Tree-Structured Directory:**
   - **Hierarchical directories where directories can have subdirectories.**
   - **Widely used, allowing for a clear organization of files and subdirectories.**

4. **Acyclic Graph Directory:**
    - **Allows for shared files, where files can exist in multiple directories.**
    - **Files can be accessed from multiple locations, useful for collaboration and sharing.**

---

4. (b) Contiguous and Linked Allocation Methods:

1. **Contiguous Allocation:**
    - **Files are stored in a single contiguous block of memory.**
    - **Advantages: Easy and fast access, simple management.**
    - **Disadvantages: Wastage of space (external fragmentation), difficulty in finding large contiguous spaces.**
2. **Linked Allocation:**

- **Files are stored in non-contiguous blocks; each block points to the next one.**
- **Advantages: No fragmentation, flexible file sizes.**
- **Disadvantages: Slower access due to multiple disk accesses, requires extra space for pointers.**

---

4. (c) Disk Scheduling Algorithms:

Given:

- **Disk Drive Cylinders: 200 cylinders (0-199).**
- **Current Head Position: 53.**
- **Pending Requests: 98, 183, 37, 122, 14, 124, 65, 67.**

(i) FCFS (First-Come, First-Served):

- **Process requests in the order they arrive.**

- **Request Order: 98, 183, 37, 122, 14, 124, 65, 67.**

1. **From 53 to 98: |98 - 53| = 45**
2. **From 98 to 183: |183 - 98| = 85**
3. **From 183 to 37: |183 - 37| = 146**
4. **From 37 to 122: |122 - 37| = 85**
5. **From 122 to 14: |122 - 14| = 108**
6. **From 14 to 124: |124 - 14| = 110**
7. **From 124 to 65: |124 - 65| = 59**
8. **From 65 to 67: |67 - 65| = 2**

Total Distance: 45 + 85 + 146 + 85 + 108 + 110 + 59 + 2 = 640 cylinders.

(ii) SSTF (Shortest Seek Time First):

- **Select the request that is closest to the current head position.**

1. **From 53, the closest request is 65. |65 - 53| = 12.**

2. **From 65, the closest request is 67. |67 - 65| = 2.**
3. **From 67, the closest request is 37. |67 - 37| = 30.**
4. **From 37, the closest request is 14. |37 - 14| = 23.**
5. **From 14, the closest request is 98. |98 - 14| = 84.**
6. **From 98, the closest request is 122. |122 - 98| = 24.**
7. **From 122, the closest request is 124. |124 - 122| = 2.**
8. **From 124, the closest request is 183. |183 - 124| = 59.**

Total Distance: 12 + 2 + 30 + 23 + 84 + 24 + 2 + 59 = 236 cylinders.

(iii) SCAN (Elevator Algorithm):

- **The head moves in one direction to the end, servicing requests, then reverses direction.**

1. **Start at 53, and move right (to the end) servicing requests.**
    - **Right side requests: 65, 67, 98, 122, 124, 183.**
    - **Move to cylinder 199 (end), servicing right-side requests.**
    - **Distance moved = 199 - 53 = 146.**
2. **Reverse direction and move left, servicing requests.**
    - **Left side requests: 37, 14.**
    - **Distance moved = 53 (reverse) = 53, 14 (reverse) = 23.**

Total Distance: 146 (right) + 53 (reverse) + 23 (reverse) = 222 cylinders.

(iv) C-LOOK (Circular LOOK):

- **Similar to SCAN, but the head only moves to the furthest request in one direction and then wraps around to the other side.**

1. **Start at 53, move right to the end, servicing requests.**
   - **Right-side requests: 65, 67, 98, 122, 124, 183.**
   - **Move to cylinder 199 (end), servicing right-side requests.**
   - **Distance moved = 199 - 53 = 146.**
2. **Jump to the lowest cylinder (14), then move to 37.**
   - **Left side requests: 37, 14.**
   - **Distance moved = 37 - 14 = 23.**

Total Distance: 146 (right) + 23 (left) = 169 cylinders.

Summary of Total Distances:

- **FCFS: 640 cylinders.**
- **SSTF: 236 cylinders.**
- **SCAN: 222 cylinders.**
- **C-LOOK: 169 cylinders.**

5.Q-. (a) What are the various types of files in LINUX operating system? Explain. (CO5)

(b) Explain the command structure used in LINUX. Write a shell script to check the given number is a prime number or not a prime number.

(COS)

(d) Explain various design principles and components of a LINUX operating system.

ANS -(a) Types of Files in LINUX Operating System:

1. **Regular Files:**

   - **The most common type of file, which contains data. These files can be text, binary, or executable files.**
   - **Examples: .txt, .jpg, .exe.**
2. **Directory Files:**
   - **Files that contain information about other files, essentially organizing files into a hierarchical structure (directories).**
   - **Examples: /home/user, /var/log.**
3. **Special Files:**
   - **These represent devices and are used to interact with hardware or virtual devices.**
   - **Types:**
     - **Block Files: Used for devices that transmit data in blocks (e.g., hard drives).**
     - **Character Files: Used for devices that transmit data**

character by character (e.g., keyboards, mice).

- Examples: /dev/sda (block), /dev/tty (character).

4. **FIFO (Named Pipe) Files:**
   - Used for communication between processes. Data is transferred through the pipe in a first-in, first-out manner.
   - Example: /tmp/myfifo.
5. **Socket Files:**
   - Used for inter-process communication (IPC) over networks.
   - Example: /tmp/mysocket.

(b) Command Structure in LINUX:

The structure of a typical LINUX command consists of the following components:

1. **Command:** The name of the command to execute (e.g., ls, cat).
2. **Options:** Modifies the behavior of the command, often prefixed by - (e.g., -l, -a).

3. **Arguments: Specifies the files or data on which the command operates (e.g., file.txt, /home/user).**

   Syntax:

   ```
   command [options] [arguments]
   ```

   Example:

   ```
   ls -l /home/user
   ```

- **ls: Command to list files.**
- **-l: Option to list in long format.**
- **/home/user: Argument specifying the directory.**

Shell Script to Check if a Number is Prime:

```
#!/bin/bash
# Prime number checker script

echo "Enter a number:"
read num

if [ $num -le 1 ]; then
   echo "$num is not a prime number."
   exit
fi
```

```
for ((i = 2; i <= num / 2; i++)); do
    if [ $(($num % $i)) -eq 0 ]; then
        echo "$num is not a prime number."
        exit
    fi
done

echo "$num is a prime number."
```

Explanation:

- **The script reads a number from the user.**
- **It checks if the number is less than or equal to 1; if so, it's not prime.**
- **Then, it checks for divisibility from 2 to the number divided by 2. If it finds a divisor, the number is not prime.**
- **If no divisors are found, it confirms that the number is prime.**

(c) Design Principles and Components of the LINUX Operating System:

1. **Design Principles:**
    - **Modularity: LINUX is designed as a modular system where components like the kernel, libraries, and user interfaces are separate.**
    - **Multitasking: Supports multitasking, allowing multiple processes to run simultaneously.**
    - **Portability: LINUX can run on different hardware platforms, thanks to its portable architecture.**
    - **Security and Privilege: LINUX uses a user-based security model with permissions to control access to resources.**
    - **Open Source: LINUX is open source, allowing users to modify and distribute the system freely.**
    - **Efficiency: Optimized for performance, LINUX can run on a**

wide range of hardware from servers to embedded systems.

2. **Components of the LINUX Operating System:**
    - Kernel: The core of the operating system that manages hardware resources, memory, processes, and input/output.
    - Shell: The user interface for interacting with the kernel. It accepts commands and executes them.
    - File System: Organizes data storage and retrieval. LINUX uses a hierarchical file system with directories and files.
    - System Libraries: A collection of functions and routines that programs can use to perform operations (e.g., the C library).
    - System Utilities: Basic tools and programs that perform system

management tasks, such as managing files, processes, and users.

- **User Space: The area where user applications and processes run, separated from kernel space for security and stability.**

In summary, LINUX is built on a modular and secure foundation with a clear separation between user space and kernel space. It supports multitasking, is highly portable, and follows efficient design principles to meet the needs of a wide variety of computing environments.