USN □□□□□□□□□□

CELEBRATING 30 YEARS
CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A++ GRADE BY NAAC

Internal Assessment Test 3 – March 2024

SCHEME AND SOLUTION

Sub:	**Operating Systems**				Sub Code:	**BCS303**	Branch:	**CSE**
Date:	**06/03/24**	Duration:	**90 minutes**	Max Marks: **50**	Sem / Sec:	**III / A, B, C**		**OBE**

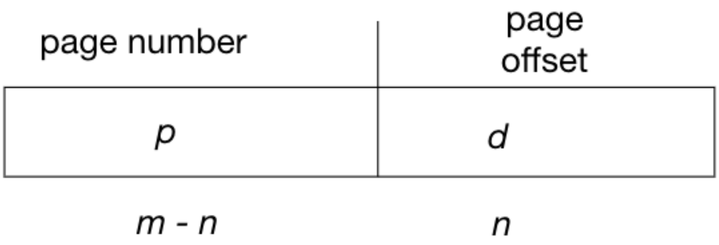
		Answer any FIVE FULL Questions	MARKS	CO	RBT
1	a	**Explain Paging hardware with TLB.(Diagram-2.5+Explaination 2.5)**	[5]	3	L3

Physical address space of a process can be noncontiguous; process is allocated physical memory whenever the latter is available, Divide physical memory into fixed-sized blocks called frames, Size is power of 2, between 512 bytes and 16 Mbytes, Divide logical memory into blocks of same size called pages, Keep track of all free frames, To run a program of size N pages, need to find N free frames and load program, Set up a page table to translate logical to physical addresses, Backing store likewise split into pages, Still have Internal fragmentation

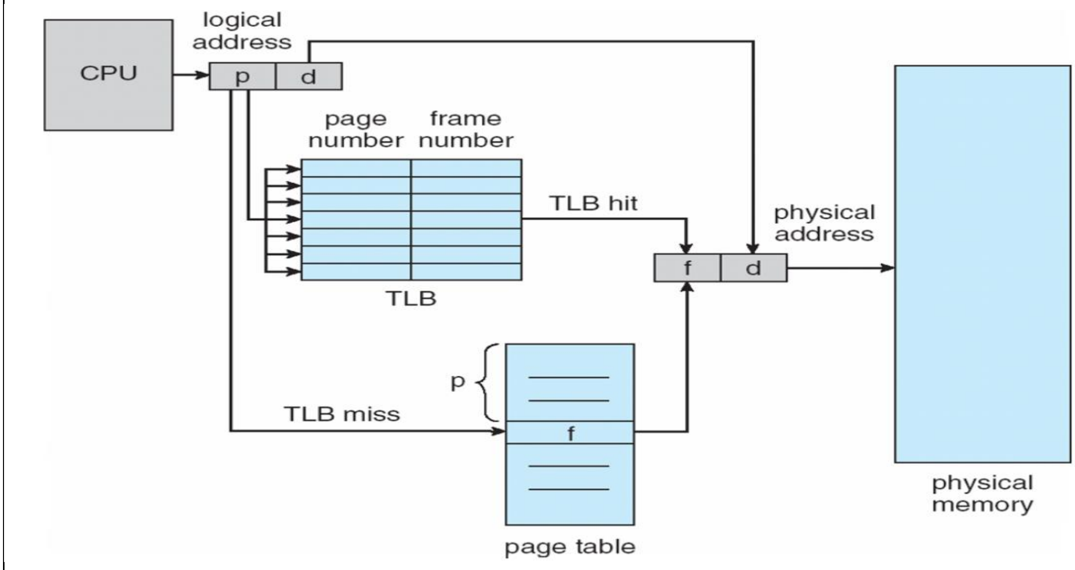
Address generated by CPU is divided into:

Page number (p) – used as an index into a page table which contains base address of each page in physical memory

Page offset (d) – combined with base address to define the physical memory address that is sent to the memory unit

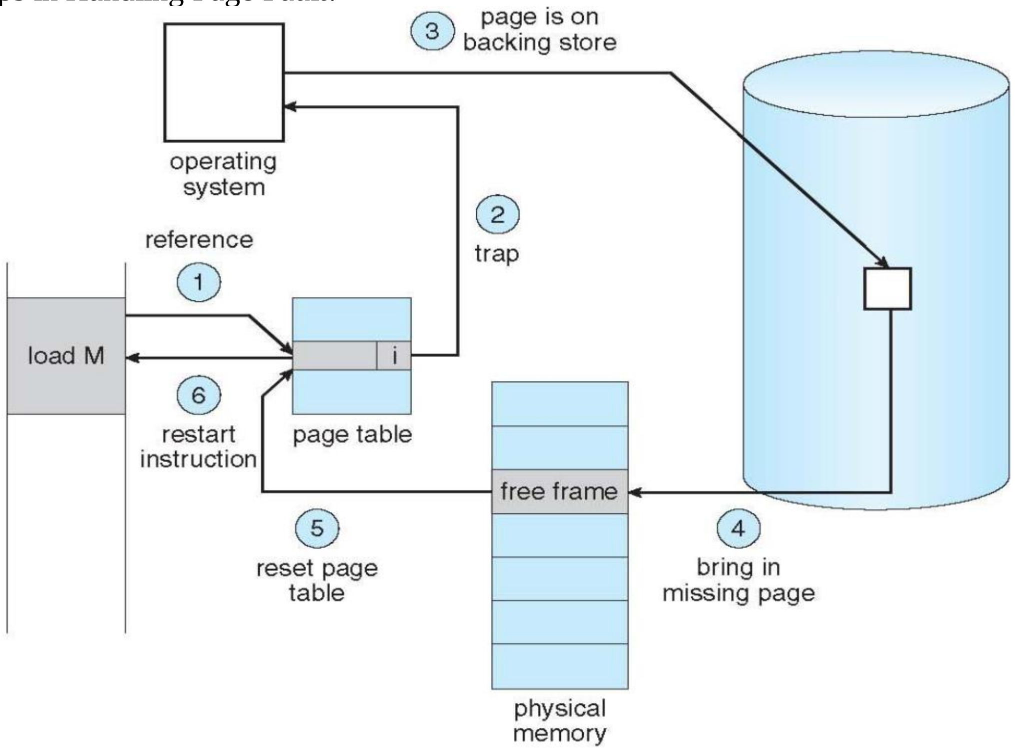


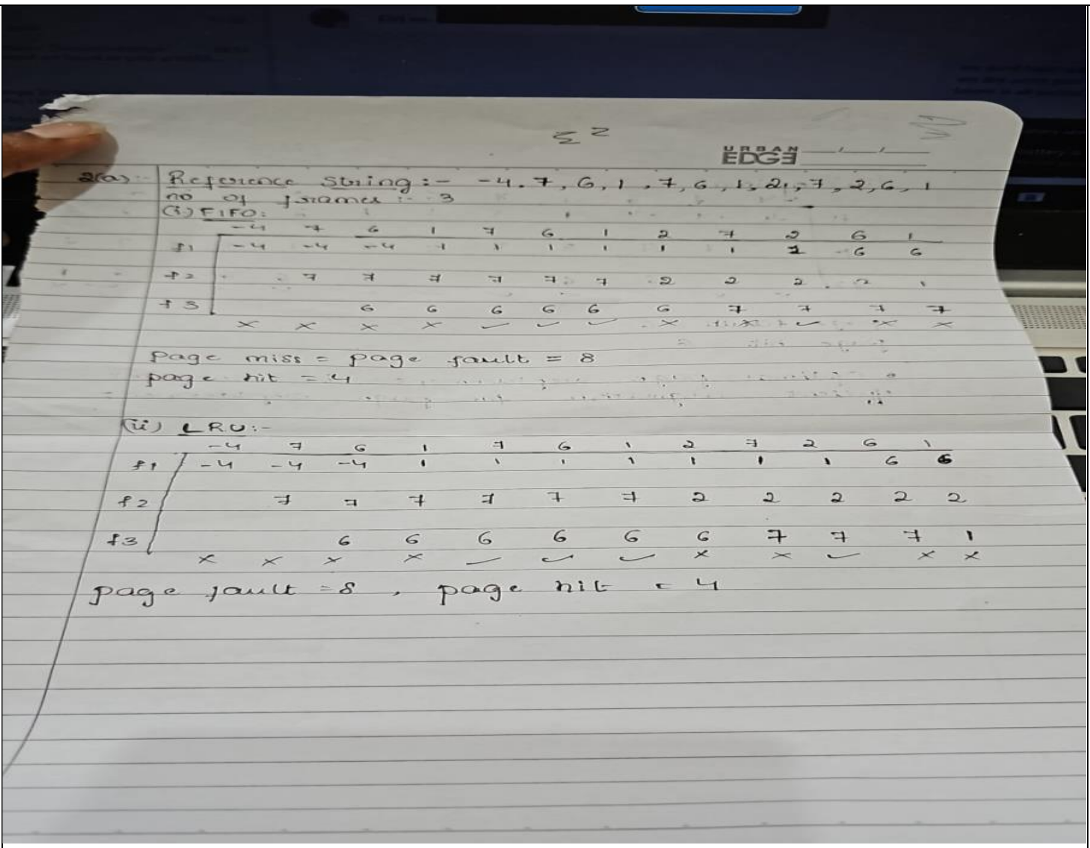
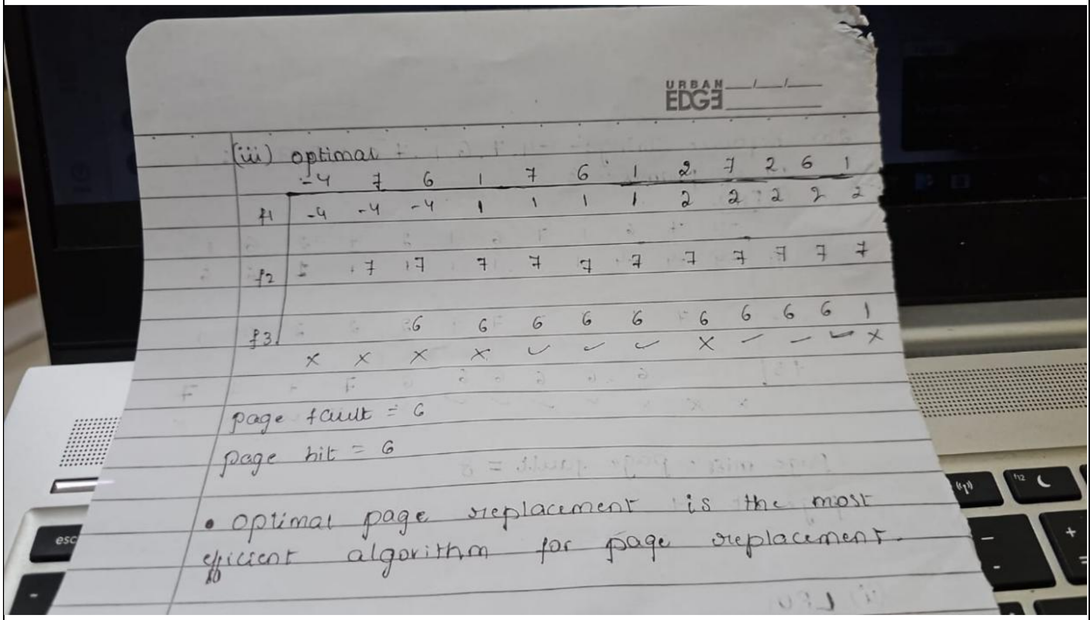
For given logical address space 2^m and page size 2^n



Page table is kept in main memory

Page-table base register (PTBR) points to the page table

		Page-table length register (PTLR) indicates size of the page table In this scheme every data/instruction access requires two memory accesses, One for the page table and one for the data / instruction, The two memory access problem can be solved by the use of a special fast-lookup hardware cache called associative memory or translation look-aside buffers (TLBs), Some TLBs store address-space identifiers (ASIDs) in each TLB entry – uniquely identifies each process to provide address-space protection for that process, Otherwise need to flush at every context switch, TLBs typically small (64 to 1,024 entries), On a TLB miss, value is loaded into the TLB for faster access next time, Replacement policies must be considered, Some entries can be wired down for permanent fast access			
	b	**What is Page Fault? With neat diagram explain the steps in handling a Page Fault (Diagram-2.5+Explaination 2.5)** A page fault is a term in computer memory management that occurs when a program or process tries to access data or code that is not currently in physical RAM (Random Access Memory). In a virtual memory system, the operating system uses a combination of physical RAM and disk space to create the illusion of a larger amount of available memory than is physically present. When a program needs to access a specific piece of data or code, the operating system checks if it is in the physical RAM. If the required data or code is not present in the RAM but is stored on the disk, a page fault is triggered. The operating system then needs to retrieve the required page (a fixed-size block of memory) from the disk and load it into the physical RAM before allowing the program to access the requested information. Page faults are a normal part of virtual memory systems and are managed by the operating system to optimize the use of physical memory and provide the illusion of a larger, continuous memory space to applications. Efficient handling of page faults is crucial for maintaining good system performance. Steps in Handling Page Fault:  3 page is on backing store operating system 2 trap reference 1 load M i 6 restart instruction page table free frame 5 reset page table 4 bring in missing page physical memory	[5]	3	L2
2	a	**Consider the following page reference string - 4, 7, 6, 1, 7, 6, 1, 2, 7, 2, 6, 1** **Assuming there are 3 memory frames and all are empty initially, how many page faults would occur in case of the following page replacement algorithms. Which algorithm is efficient?** **(i) FIFO (ii) LRU (iii) Optimal(Diagrams-5+ Explanation 5)**	[10]	3	L3

ᶻ²

URBAN EDGE

2(a) Reference String :- -4, 7, 6, 1, 7, 6, 1, 2, 7, 2, 6, 1

no of frames : 3

(i) FIFO:

	-4	7	6	1	7	6	1	2	7	2	6	1
f1	-4	-4	-4	1	1	1	1	1	1	2	6	6
f2		7	7	7	7	7	7	2	2	2	2	1
f3			6	6	6	6	6	6	7	7	7	7
	×	×	×	×	✓	✓	✓	×	×	✓	×	×

Page miss = page fault = 8

page hit = 4

(ii) LRU:-

	-4	7	6	1	7	6	1	2	7	2	6	1
f1	-4	-4	-4	1	1	1	1	1	1	1	6	6
f2		7	7	7	7	7	7	2	2	2	2	2
f3			6	6	6	6	6	6	7	7	7	1
	×	×	×	×	✓	✓	✓	×	×	✓	×	×

page fault = 8 , page hit = 4

URBAN EDGE

(iii) optimal

	-4	7	6	1	7	6	1	2	7	2	6	1
f1	-4	-4	-4	1	1	1	1	2	2	2	2	2
f2		7	7	7	7	7	7	7	7	7	7	7
f3			6	6	6	6	6	6	6	6	6	1
	×	×	×	×	✓	✓	✓	×	✓	✓	✓	×

page fault = 6

page hit = 6

- optimal page replacement is the most efficient algorithm for page replacement.

3	a	What is Thrashing? Briefly describe the methods in which it can be controlled.**(Diagram-2.5+Explaination 2.5)** Thrashing is a phenomenon in computer systems where excessive paging occurs, leading to a decrease in overall system performance. Paging is a memory management scheme where portions of a program or data are swapped in and out of the main memory (RAM) and secondary storage (usually a hard disk) to accommodate the limited physical memory available. When a system experiences thrashing, it means that the majority of the time is spent swapping pages in and out of the disk, rather than executing useful tasks. This results in a significant degradation of performance and responsiveness. Several methods can be employed to control thrashing Working Set Model:	[5]	3	L1

- The working set model focuses on keeping in memory only those pages that are actively used by a process.
- The working set of a process is the set of pages that it is currently accessing. By monitoring and adjusting the working set, the system can minimize unnecessary page swaps.

Page-Fault Frequency (PFF) Scheme:

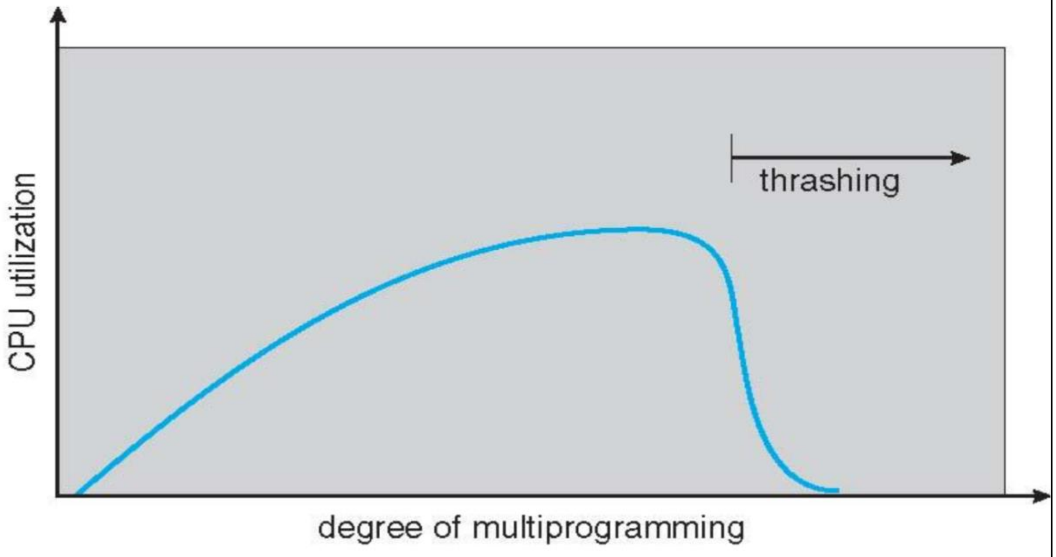
- This method involves measuring the rate of page faults and adjusting the system parameters accordingly.
- If the page-fault rate is too high, it indicates that the system is spending too much time swapping pages. Adjustments to the system, such as increasing the amount of physical memory or modifying page replacement algorithms, can be made based on this information.

Locality of Reference:

- Exploiting the principle of locality can help reduce thrashing. Programs often exhibit temporal and spatial locality, meaning they access a relatively small portion of memory frequently (temporal) and adjacent memory locations (spatial).
- Algorithms like Least Recently Used (LRU) or Clock can be used to prioritize keeping pages in memory that are more likely to be accessed soon.

Priority Paging:

- Priority paging involves assigning priorities to different pages based on their importance or likelihood of being accessed.
- When the system needs to swap pages, priority paging ensures that essential pages are given precedence, reducing the likelihood of thrashing.



b

Explain the various types of Directory Structures.(Diagram-2.5+Explaination 2.5)

Directory structures, also known as file systems or filesystem structures, are methods of organizing and storing files on a computer storage device. Different types of directory structures offer various advantages and are suited to different purposes. Here are some common types of directory structures:

Single-Level Directory:

.

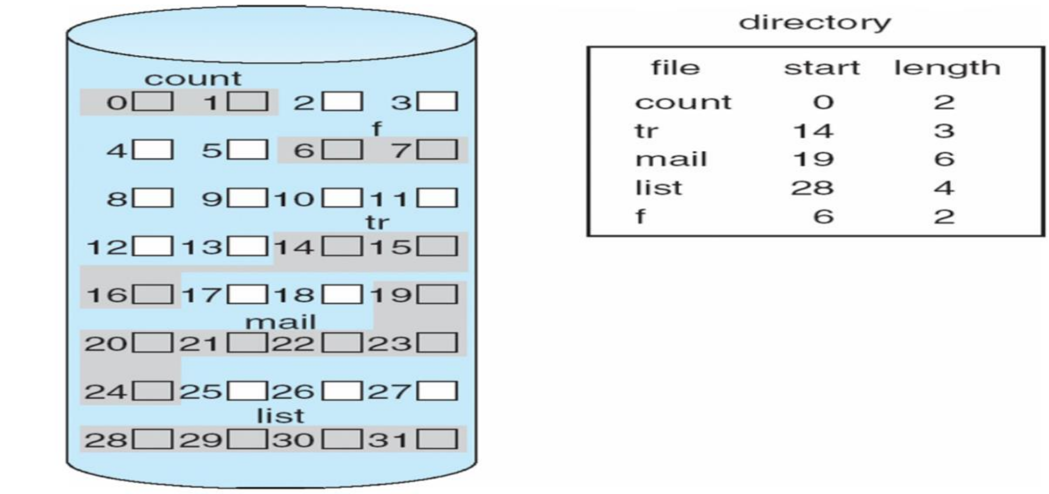
□In a single-level directory structure, all files are stored in a single directory without any subdirectories.

□This structure is simple but can become unmanageable as the number of files increases.

[5] 4 L2

		. Two-Level Directory: . This structure introduces the concept of dividing the file system into two levels - a user directory and a system directory. Each user has their own directory, and the system has a directory for each user. The user directory names are unique within the system directory. . Tree-Structured Directory: . In a tree-structured directory, directories are organized hierarchically, forming a tree-like structure. Each node in the tree represents a directory, and the leaves represent files. This structure allows for more efficient organization and retrieval of files. . Acyclic-Graph Directory: . An acyclic-graph directory structure is similar to a tree structure but allows for shared subdirectories. This structure eliminates the restriction that each directory can only appear in one location, enabling more flexible organization. . General Graph Directory: . Unlike acyclic-graph structures, general graph directory structures allow for cycles, which means directories can have multiple links and connections. While this flexibility can be powerful, it can also lead to complexity and potential issues if not managed carefully. . Distributed Directory: . Distributed directory structures are designed for distributed file systems where files are stored on multiple servers or locations. This structure facilitates access to files across a network, providing a unified view of the directory structure for users.			
4	a	**Briefly discuss about the different types of Allocation Methods in File System implementation(Diagram-5+Explaination 5)** File system allocation methods determine how disk space is allocated to files. There are several allocation methods, each with its own advantages and disadvantages. Here are the main types of file allocation methods: . Contiguous Allocation: . In contiguous allocation, each file occupies a contiguous block of disk space. Advantages include simple and fast access since the entire file is stored in a single block.	[10]	4	L2

Disadvantages include fragmentation issues, as over time, the disk becomes fragmented with small gaps between files.



file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

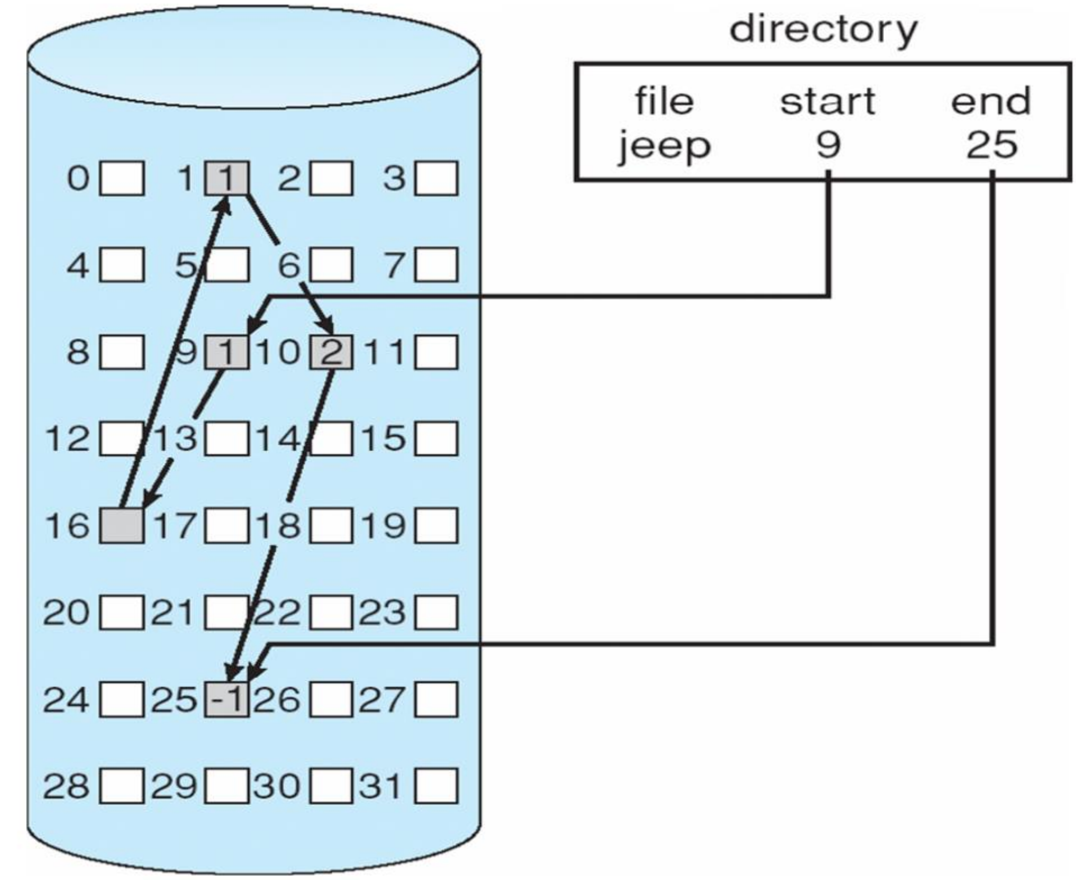
Linked Allocation:

.

□Linked allocation uses a linked list data structure to connect blocks of a file scattered throughout the disk.

□Each block contains a pointer to the next block in the sequence.

□Advantages include efficient space utilization, but accessing a specific portion of the file can be slower due to scattered blocks and traversal of the linked list.



file	start	end
jeep	9	25

Indexed Allocation:

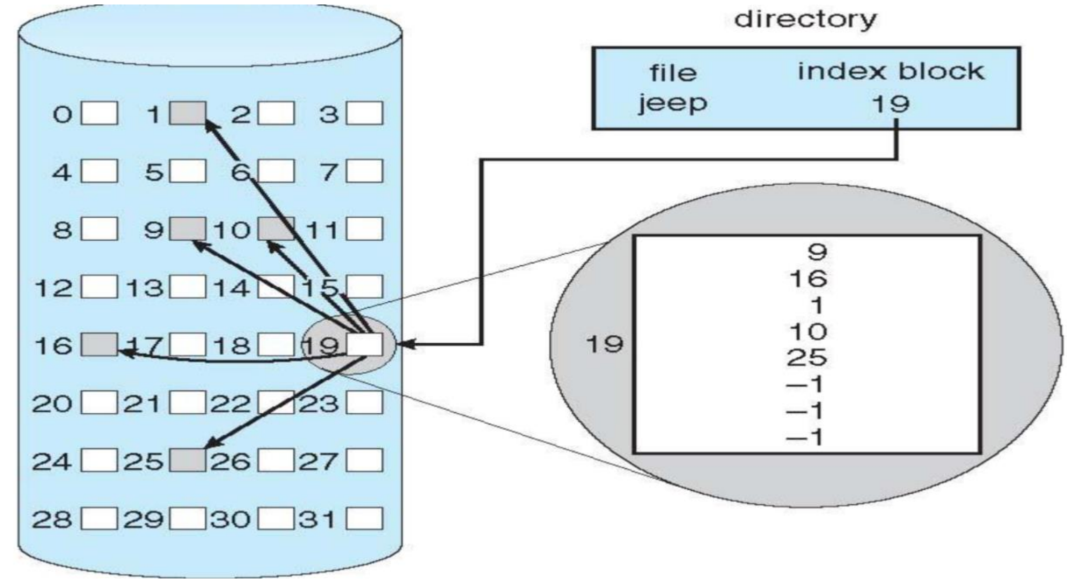
.

<table>
<tr>
<td></td>
<td></td>
<td>

□Indexed allocation uses an index block that contains pointers to all the blocks of a file.

□The index block is either in memory or stored on the disk.

□Advantages include quick access to any block of the file through the index, but the index block itself can become a bottleneck for large files.



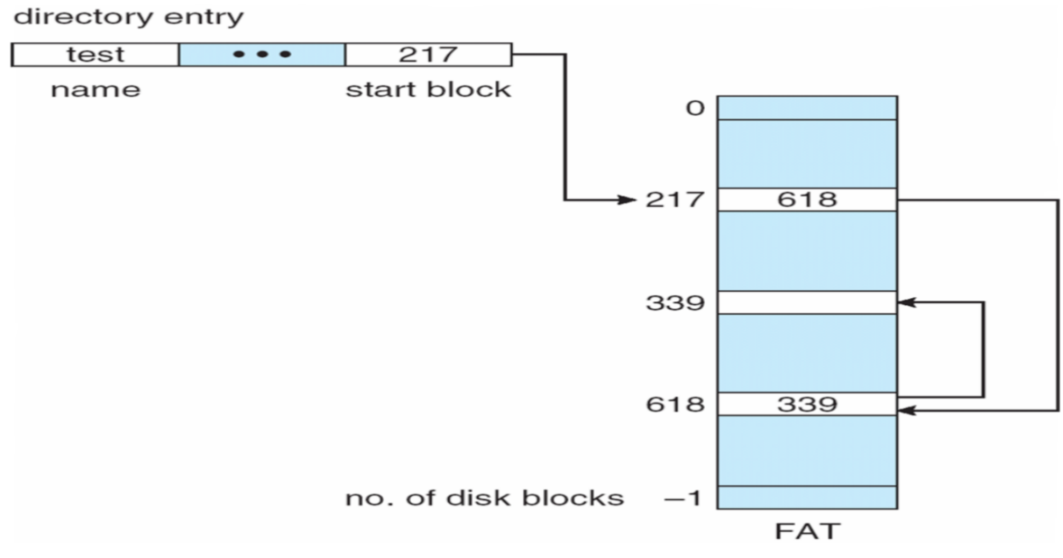
File Allocation Table (FAT):

.

□The File Allocation Table is a variation of indexed allocation, commonly used in FAT file systems like FAT12, FAT16, and FAT32.

□A central table maintains pointers to data blocks for each file.

□Advantages include simplicity and ease of implementation, but the table can become large and affect performance.



</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>a</td>
<td><b>Explain in detail about Free Space Management (Diagram-2.5+Explaination 2.5)</b></td>
<td>[5]</td>
<td>4</td>
<td>L2</td>
</tr>
</table>

Free space management is a critical aspect of file systems, ensuring efficient utilization of available storage space on a computer's disk. It involves keeping track of which blocks or clusters on the disk are available for allocation to new files or for extending existing ones. Different free space management techniques exist, each with its own advantages and challenges. Here are several methods used for managing free space:

.

Bit Vector (Bitmap) Allocation:

.

□In this method, a bitmap is used to represent the allocation status of each block on the disk.

□Each bit in the bitmap corresponds to a block, where a '0' indicates the block is free, and a '1' indicates it is allocated.

□The bitmap provides a quick overview of available and occupied blocks, making it efficient for searching free space.

.

Linked List (Linked Clusters) Allocation:

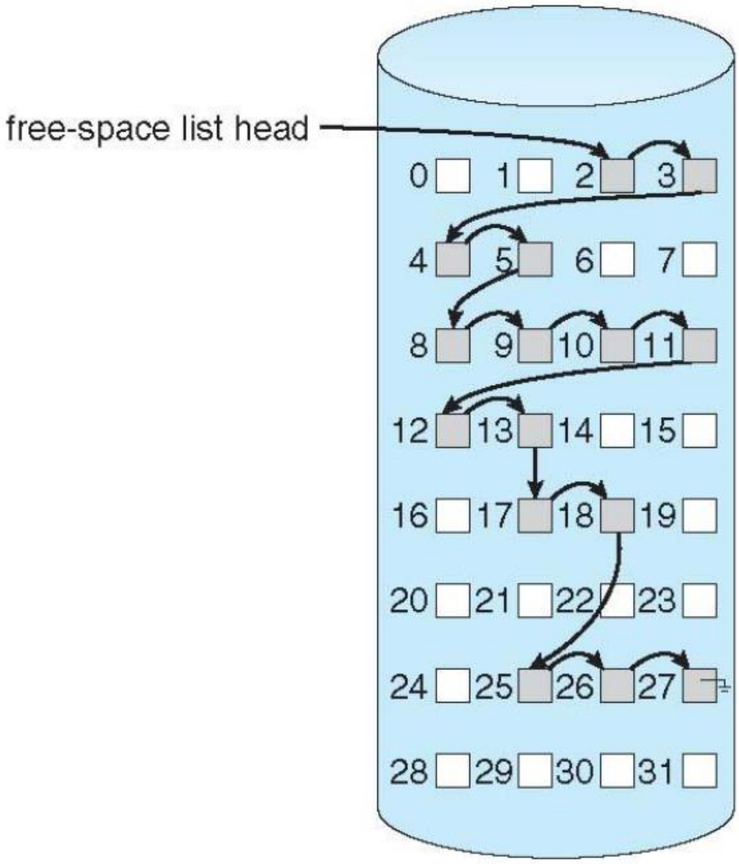
.

□Free space can be managed using a linked list of free blocks or clusters.

□Each free block contains a pointer to the next free block, forming a linked list.

□To allocate space, the system traverses the linked list, updating pointers as necessary.

□This method is simple but can be inefficient for large storage systems due to traversal overhead.

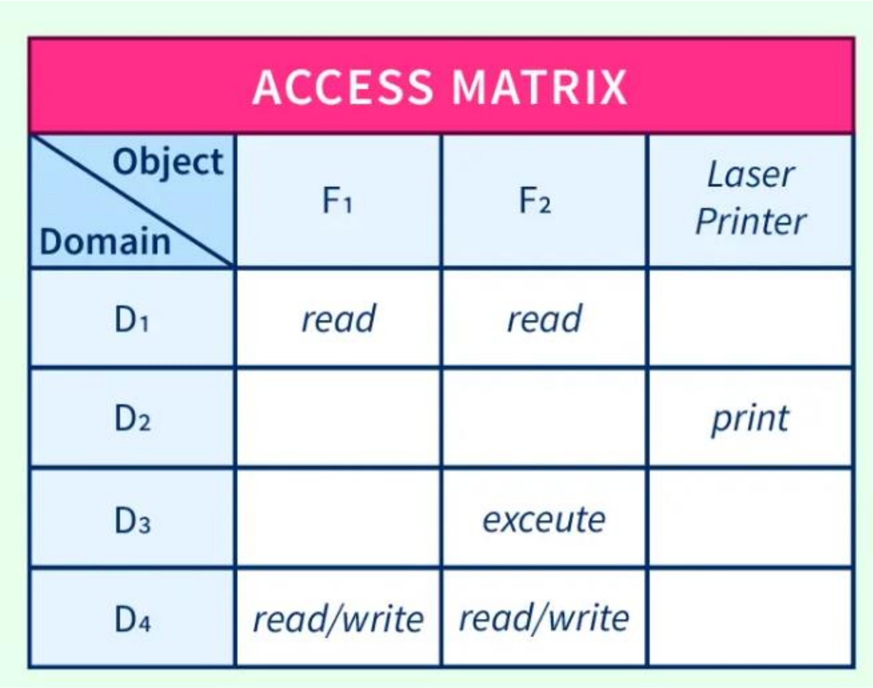


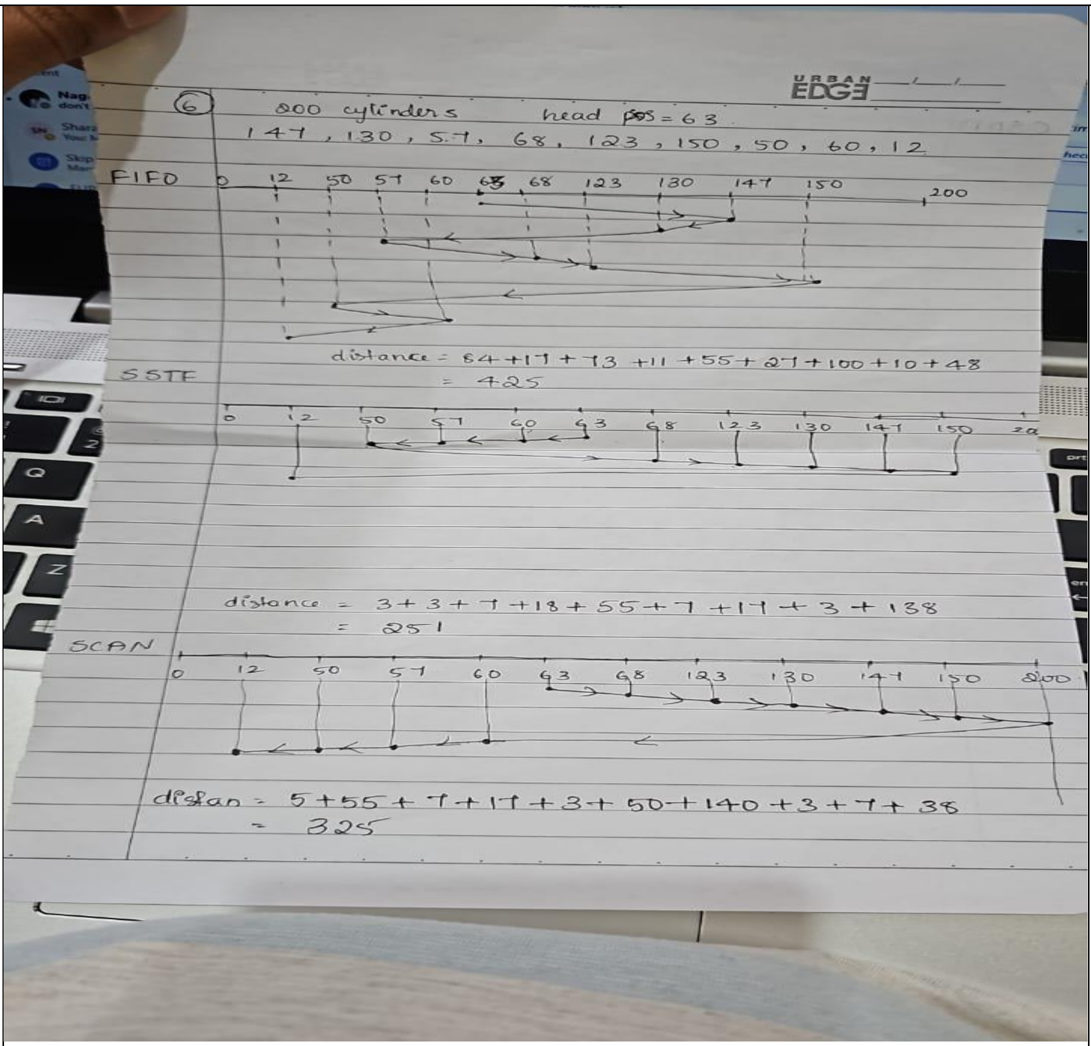
Grouping (Clustering) Free Space:

.

□Free space can be grouped into clusters or extents.

		□Clusters are contiguous blocks of free space, and the file system allocates entire clusters rather than individual blocks. □This helps reduce fragmentation and improves performance but can lead to internal fragmentation if the cluster size is larger than the file size. . Counting (Counting Blocks) Method: . □In this method, a counter keeps track of the number of free blocks on the disk. □When a block is allocated or deallocated, the counter is updated accordingly. □While simple, this method may suffer from inefficiency and contention in a multi-user environment.			
	b	**Explain the access matrix method of system protection in OS.(Diagram-2.5+Explaination 2.5)** The access matrix is a model used for implementing system protection and access control in operating systems. It provides a structured representation of the access rights between subjects (e.g., users, processes) and objects (e.g., files, resources) in a computer system. The access matrix method is a powerful and flexible approach that enables administrators to define and control access permissions effectively. Here are the key components of the access matrix: . Subjects: . □Subjects represent entities that can perform operations or actions on objects. □Examples of subjects include users, processes, or even devices. . Objects: . □Objects are resources or entities on which subjects perform operations. □Examples of objects include files, directories, devices, and system resources. . Access Rights: . □Access rights define the permissions or operations that subjects can perform on objects. □Common access rights include read, write, execute, delete, and control. . Access Matrix: . □The access matrix is a table that visually represents the relationships between subjects and objects based on access rights. □Rows in the matrix correspond to subjects, columns correspond to objects, and the cells contain the access rights associated with the respective subject-object pair. The access matrix method provides several benefits: Flexibility: The matrix allows fine-grained control over access permissions, supporting a wide range of security policies. Scalability: It can accommodate a growing number of subjects and objects without significant restructuring.	[5]	4	L2

<table>
<tr>
<td></td>
<td></td>
<td>Expressiveness: The matrix can capture complex relationships and access patterns in a concise and structured manner.

ACCESS MATRIX
<table>
<tr><th>Object / Domain</th><th>F_1</th><th>F_2</th><th>Laser Printer</th></tr>
<tr><td>D_1</td><td>read</td><td>read</td><td></td></tr>
<tr><td>D_2</td><td></td><td></td><td>print</td></tr>
<tr><td>D_3</td><td></td><td>exceute</td><td></td></tr>
<tr><td>D_4</td><td>read/write</td><td>read/write</td><td></td></tr>
</table>
</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>a</td>
<td><b>A disk has 200 cylinders, it is currently servicing request at 63. The Queue of
pending request in FIFO order is 147,130, 57, 68, 123, 150, 50, 60, 12. Starting at the
current head position what is the distance travelled by the disk arm (in cylinders)
using FCFS, SSTF, SCAN, CSCAN, LOOK and C LOOK
(Diagrams-5+ Explanation 5)</b></td>
<td>[10]</td>
<td>4</td>
<td>L3</td>
</tr>
</table>

6 200 cylinders head pos = 63

147, 130, 57, 68, 123, 150, 50, 60, 12

FIFO

0
12
50
57
60
63
68
123
130
147
150
200

distance = 84 + 17 + 73 + 11 + 55 + 27 + 100 + 10 + 48
= 425

SSTF

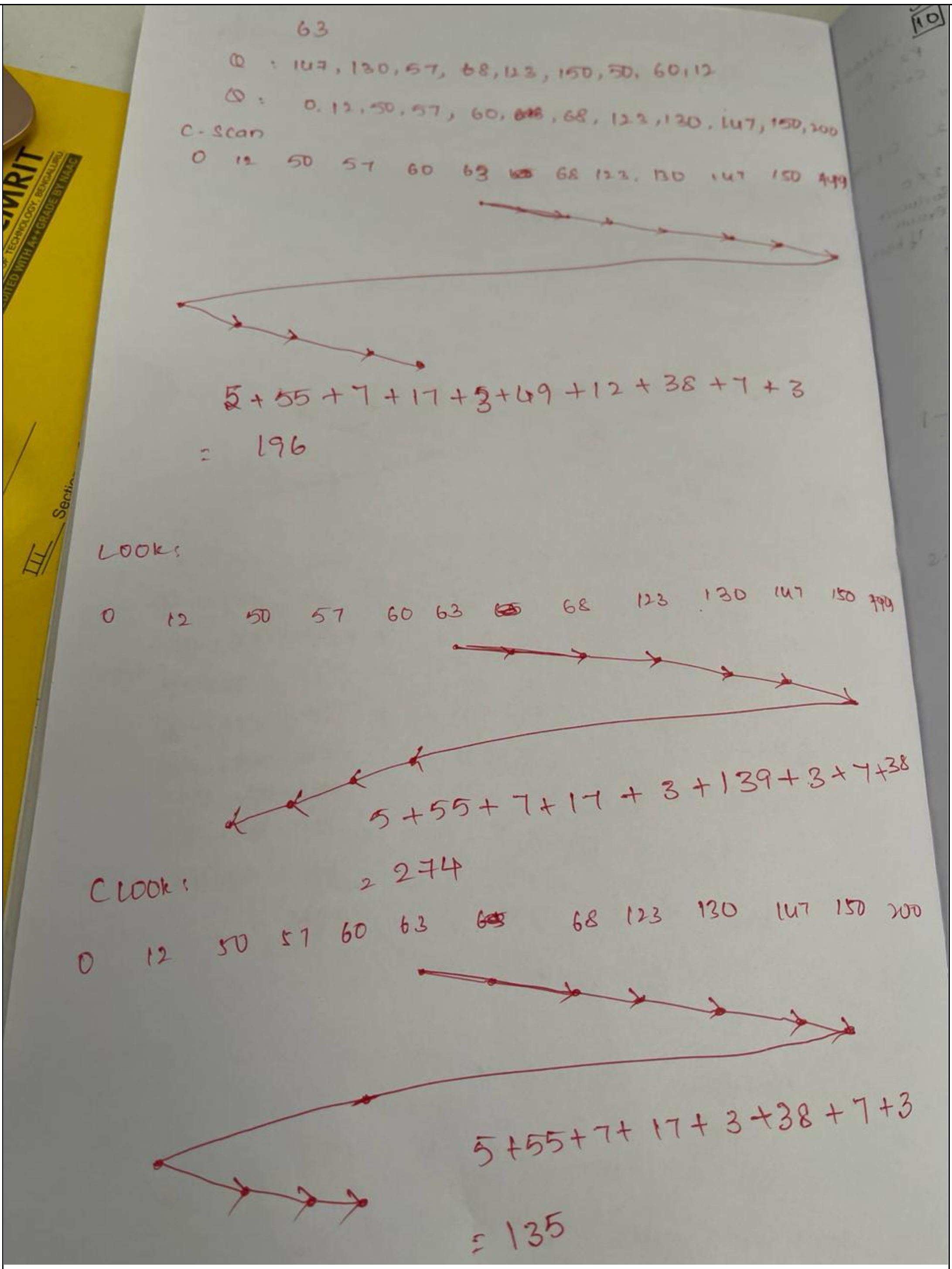
0
12
50
57
60
63
68
123
130
147
150
200

distance = 3 + 3 + 7 + 18 + 55 + 7 + 17 + 3 + 138
= 251

SCAN

0
12
50
57
60
63
68
123
130
147
150
200

distan = 5 + 55 + 7 + 17 + 3 + 50 + 140 + 3 + 7 + 38
= 325



63

① : 147, 130, 57, 68, 123, 150, 50, 60, 12

② : 0, 12, 50, 57, 60, ~~65~~, 68, 123, 130, 147, 150, 200

C-Scan

0 12 50 57 60 63 ~~65~~ 68 123, 130 147 150 199

5 + 55 + 7 + 17 + 3 + 49 + 12 + 38 + 7 + 3

= 196

LOOK:

0 12 50 57 60 63 ~~65~~ 68 123 130 147 150 199

5 + 55 + 7 + 17 + 3 + 139 + 3 + 7 + 38

= 274

CLOOK:

0 12 50 57 60 63 ~~65~~ 68 123 130 147 150 200

5 + 55 + 7 + 17 + 3 + 38 + 7 + 3

= 135

10

CI **CCI** **HoD**

--All the Best--

CO-PO Mapping

Course Outcomes		Modules covered	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CO1	Describe the Operating System Structure and Services.	**1**	3	-	-	-	-	-	-	-	-	-	-	3	-	2	-	-
CO2	Summarize the Process Management concepts like Processes, Threads, CPU Scheduling, Process Synchronization and Deadlocks	**1, 2**	3	2	2	-	-	-	-	-	-	-	-	3	-	2	-	-
CO3	Interpret the Memory Management concepts with respect to Main Memory and Virtual Memory.	**3, 4**	3	2	2	-	-	-	-	-	-	-	-	3	-	2	-	-
CO4	Discuss the Storage Management concepts like File-System Interface, File-System Implementation and Mass-Storage Structure	**4, 5**	3	2	2	-	-	-	-	-	-	-	-	3	-	2	-	-
CO5	Elucidate the Protection features in Operating System and case study in Linux OS.	**5**	3	2	2	-	-	-	-	-	-	-	-	3	-	2	-	-