| Q. | | Answer | M |
|---|---|---|---|
| **1** | 1) | Symmetric multiprocessing architecture of computer system uses shared ---- | |
| | A: | Processors, Memory | |
| | B: | Memory, Bus | |
| | C: | Bus | |
| | D: | Hard drives | |
| | Ans: | A: Processors, Memory | |
| | | | |
| | 2. | Aging is called as------ | |
| | A: | Increasing the time of execution | |
| | B: | Increasing the priority of a process | |
| | C: | Decreasing the priority of a process | |
| | D: | Drawback of FCFS | |
| | Ans: | C: Decreasing the priority of a process | |
| | | | |
| | 3. | Suppose that a process is in “Blocked” state waiting for some I/O service. When the service is completed, it goes to the----------- | |
| | A: | Running state | |
| | B: | Ready State | |
| | C: | Terminated state | |
| | D: | Suspended state | |
| | Ans: | B: Ready State | |
| | | | |
| | 4. | Which one of the following is not true? | |
| | A: | kernel is the program that constitutes the central core of the operating system | |
| | B: | kernel is the first part of operating system to load into memory during booting | |
| | C: | kernel is made of various modules which can not be loaded in running operating system | |
| | D: | kernel remains in the memory during the entire computer session | |
| | Ans: | C: kernel is made of various modules which can not be loaded in running operating system | |
| | | | |
| | 5. | Which of the statement is true in case of PCB | |
| | A: | PCB is used to identify process area | |
| | B: | PCB is created per process which is used to store relevant information about process. | |
| | C: | PCB is created to store process in user area. | |
| | D: | PCB is used by user to access process code and data. | |
| | Ans: | B: PCB is created per process which is used to store relevant information about process | |
| | | | |
| | 6. | Resource request is done in particular order to avoid ----- | |
| | A: | Deadlock | |
| | B: | Confusion | |
| | C: | Overhead on OS | |
| | D: | Resource conflict | |

|  |  |  |  |
|---|---|---|---|
|  | Ans: | A: Deadlock |  |
|  |  |  |  |
|  | 7. | -------is generally faster than ------ and ------. |  |
|  | A: | First fit, best fit, worst fit |  |
|  | B: | Best fit, first fit, worst fit |  |
|  | C: | Worst fit, best fit, first fit |  |
|  | D: | Worst fit, first fit, best fit |  |
|  | Ans: | A: First fit, best fit, worst fit |  |
|  |  |  |  |
|  | 8. | In the_algorithm, the disk arm starts at one end of the disk and moves toward the other end, servicing requests till the other end of the disk. At the other end, the direction is reversed and servicing continues. |  |
|  | A: | LOOK |  |
|  | B: | SCAN |  |
|  | C: | C-SCAN |  |
|  | D: | C-LOOK |  |
|  | Ans: | B: SCAN |  |
|  |  |  |  |
|  | 9. | Run time mapping from virtual to physical address is done by _____ |  |
|  | A: | Memory management unit |  |
|  | B: | CPU |  |
|  | C: | PCI |  |
|  | D: | API |  |
|  | Ans: | A: Memory management unit |  |
|  |  |  |  |
|  | 10. | Consider the following set of processes, the length of the CPU burst time given in milliseconds. Process Burst time P1- 24, P2- 3, P3- 7, P4- 13, P5- 21. Assuming the above process being scheduled with the SJF scheduling algorithm, which of the following statement is true? |  |
|  | A: | The waiting time for the process P5 is 10 ms |  |
|  | B: | The waiting time for the process P5 is 0 ms |  |
|  | C: | The waiting time for the process P5 is 44 ms |  |
|  | D: | The waiting time for the process P5 is 23 ms |  |
|  | Ans: | C: The waiting time for the process P5 is 44 ms |  |
|  |  |  |  |
| **2** | a) | What is Internal fragmentation? Explain static partitioned allocation with partition sizes 400,180, 100,300,45. Assuming First fit and Best fit method indicate the memory status after memory request for sizes 95, 180, 285, 380, |  |
|  | Ans: | • Internal fragmentation is a phenomenon that occurs when a process is allocated to a memory block whose size is more than the size of that process, and due to which some part of the memory is left unused. This unused memory is called internal fragmentation.<br>• Static partitioned allocation is a memory allocation technique in which the memory is divided into fixed-sized partitions, and each partition is assigned to a process. The partition sizes are fixed and do not change during the execution of the program. The partitions can be allocated to the processes in two ways:<br>i. First Fit |  |

ii. Best Fit.

- In First Fit, the memory is allocated to the first partition that is large enough to hold the process. In contrast, in Best Fit, the memory is allocated to the partition that is closest in size to the process but still larger than the process.

  Let's assume we have the following partitions: 400, 180, 100, 300, and 45. We will now allocate memory to the processes of sizes 95, 180, 285, 380, and 30 using both First Fit and Best Fit methods.

  **First Fit Method:**

| Process Size | Partition Allocated | Memory Status |
|---|---|---|
| 95 | 100 | 5 |
| 180 | 180 | 0 |
| 285 | 400 | 115 |
| 380 | 400 | 35 |
| 30 | 45 | 15 |

**Best Fit Method:**

| Process Size | Partition Allocated | Memory Status |
|---|---|---|
| 95 | 100 | 5 |
| 180 | 180 | 0 |

| | | |
|---|---|---|
| 285 | 300 | 15 |
| 380 | 400 | 20 |
| 30 | 45 | 15 |

b) Give the explanation of necessary conditions for deadlock. Explain how a resource allocation graph determines a deadlock.

Ans:

- The necessary conditions for deadlock are four conditions that must hold simultaneously for a deadlock situation to occur. These conditions are also known as Coffman conditions, and they are:
  - **Mutual exclusion:** This condition requires that at least one resource be held in a non-shareable mode, which means that only one process can use the resource at any given time. If another process requests the same resource, it has to wait until the resource is released by the current holder.
  - **Hold and wait:** This condition specifies that a process must be holding at least one resource while waiting for other resources that are currently held by other processes. This means that a process cannot release its allocated resources until it gets all the resources it needs.
  - **No pre-emption:** This condition states that a resource cannot be forcibly taken away from a process that is holding it. A process can only release a resource voluntarily after completing its task. This means that a process cannot be interrupted by another process that needs the same resource.
  - **Circular wait:** This condition implies that there exists a set of processes {P1, P2, ..., Pn} such that P1 is waiting for a resource held by P2, P2 is waiting for a resource held by P3, ..., Pn is waiting for a resource held by P1. This forms a circular chain of processes that are waiting for each other, and none of them can proceed.
- A resource allocation graph (RAG) is a graphical representation of the state of a system in terms of processes and resources. It shows which resources are held by which processes and which processes are waiting for which resources. It consists of two types of vertices: process vertices (circles) and resource vertices (squares). It also consists of two types of edges: request edges (from processes to resources) and assignment edges (from resources to processes).
- A resource allocation graph can be used to determine whether a deadlock exists in the system or not. The rules are:
  - If the RAG has no cycles, then there is no deadlock.

- If the RAG has a cycle, then there may or may not be a deadlock, depending on the type of resources involved.
  - If all the resources in the cycle are single-instance resources, then there is a deadlock. This is because each process in the cycle is holding one resource and waiting for another resource that is held by another process in the cycle, and none of them can proceed.

If some of the resources in the cycle are multi-instance resources, then there may or may not be a deadlock. This is because some processes in the cycle may be able to obtain the resources, they need from the available instances of the multi-instance resources and break the cycle. To confirm whether there is a deadlock or not, a more detailed analysis, such as the banker's algorithm, is required.

c) Consider the page reference string 1,2,3,5,2,4,5,6,2,1,2,3,7,6,3,2,1,2,3,6. Calculate the Page fault using 1. Optimal 2. LRU 3. FIFO algorithms for a memory with three

Ans:

### 1. Optimal Algorithm

- Total frames: 3
- Algorithm: OPT
- Reference string length: 20 references
- String: 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

Solution visualization

| **t** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **ref** | | **1** | **2** | **3** | **4** | **2** | **1** | **5** | **6** | **2** | **1** | **2** | **3** | **7** | **6** | **3** | **2** | **1** | **2** | **3** | **6** |
| **f** | | 1 | 2 | 3 | 4 | 4 | 4 | 5 | 6 | 6 | 6 | 6 | 3 | 7 | 7 | 7 | 2 | 1 | 1 | 1 | 6 |
| **f** | | | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 6 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 |
| **f** | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 6 | 6 | 6 | 6 | 3 | 3 | 3 | 3 |
| **hit** | | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ |
| **v** | | | | | 3 | | | 4 | 5 | | | | 1 | 2 | | | 7 | 6 | | | 1 |

- Total references: 20
- Total distinct references: 7
- Hits: 9
- Faults: 11
- **Hit rate:** 9/20 = **45%**
- **Fault rate:** 11/20 = **55%**

### 2. LRU Algorithm

- Total frames: 3
- Algorithm: LRU
- Reference string length: 20 references

- String: 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

Solution visualization

| t | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **ref** | | **1** | **2** | **3** | **4** | **2** | **1** | **5** | **6** | **2** | **1** | **2** | **3** | **7** | **6** | **3** | **2** | **1** | **2** | **3** | **6** |
| **f** | | 1 | 2 | 3 | 4 | 2 | 1 | 5 | 6 | 2 | 1 | 2 | 3 | 7 | 6 | 3 | 2 | 1 | 2 | 3 | 6 |
| **f** | | | 1 | 2 | 3 | 4 | 2 | 1 | 5 | 6 | 2 | 1 | 2 | 3 | 7 | 6 | 3 | 2 | 1 | 2 | 3 |
| **f** | | | | 1 | 2 | 3 | 4 | 2 | 1 | 5 | 6 | 6 | 1 | 2 | 3 | 7 | 6 | 3 | 3 | 1 | 2 |
| **hit** | | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ |
| **v** | | | | | 1 | | 3 | 4 | 2 | 1 | 5 | | 6 | 1 | 2 | | 7 | 6 | | | 1 |

- Total references: 20
- Total distinct references: 7
- Hits: 5
- Faults: 15
- **Hit rate:** 5/20 = **25%**
- **Fault rate:** 15/20 = **75%**

### 3. FIFO Algorithm

- Total frames: 3
- Algorithm: FIFO
- Reference string length: 20 references
- String: 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

Solution visualization

| t | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **ref** | | **1** | **2** | **3** | **4** | **2** | **1** | **5** | **6** | **2** | **1** | **2** | **3** | **7** | **6** | **3** | **2** | **1** | **2** | **3** | **6** |
| **f** | | 1 | 2 | 3 | 4 | 4 | 1 | 5 | 6 | 2 | 1 | 1 | 3 | 7 | 6 | 6 | 2 | 1 | 1 | 3 | 6 |
| **f** | | | 1 | 2 | 3 | 3 | 4 | 1 | 5 | 6 | 2 | 2 | 1 | 3 | 7 | 7 | 6 | 2 | 2 | 1 | 3 |
| **f** | | | | 1 | 2 | 2 | 3 | 4 | 1 | 5 | 6 | 6 | 2 | 1 | 3 | 3 | 7 | 6 | 6 | 2 | 1 |
| **hit** | | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ |
| **v** | | | | | 1 | | 2 | 3 | 4 | 1 | 5 | | 6 | 2 | 1 | | 3 | 7 | | 6 | 2 |

- Total references: 20
- Total distinct references: 7
- Hits: 4
- Faults: 16
- **Hit rate:** 4/20 = **20%**
- **Fault rate:** 16/20 = **80%**

**3** a) What are the various objectives and functions of Operating Systems?

Ans:

- Operating systems (OS) are an essential part of any computer system.
- They act as a communication bridge between the user and computer hardware, providing a platform on which a user can execute programs conveniently and efficiently.
- The primary goal of an operating system is to make the computer environment more convenient to use, and the secondary goal is to use the resources most efficiently.
- The following are the main objectives of an operating system:
    - To make the computer system convenient to use in an efficient manner.
    - To hide the details of the hardware resources from the users.
    - To provide users a convenient interface to use the computer system.
    - To act as an intermediary between the hardware and its users, making it easier for the users to access and use other resources.
    - To manage the resources of a computer system.

- The functions of an operating system include:

i. **Memory management:** managing the allocation and deallocation of memory to various processes and ensuring that the other process does not consume the memory allocated to one process.

ii. **Processor management:** allocating the processor to different processes and ensuring that the processor is used efficiently.

iii. **Device management:** managing the allocation of devices to different processes and ensuring that the devices are used efficiently.

iv. **File management:** managing the creation, deletion, and modification of files and directories.

v. **Security:** ensuring that the system is secure from unauthorized access and malicious software.

b) Differentiate between process and threads

Ans:

| **Process** | **Thread** |
|---|---|
| A process is an instance of a program in execution. | A thread is a sequence of instructions within a process. |
| A process has its own address space, code, data, and resources. | A thread shares the address space, code, data, and resources of its process. |
| Processes are isolated from each other and cannot directly communicate. | Threads can communicate with each other through shared memory and synchronization mechanisms. |
| Creating and switching between processes is expensive in terms of time and resources. | Creating and switching between threads is cheaper and faster than processes. |

| | |
|---|---|
| Processes can have multiple threads running concurrently within them. | Threads can only run within a process and cannot exist independently. |

c) What is virtual memory? Mention its advantages.

Ans:

- Virtual memory is a technique that allows a computer to use more memory than is physically available by using a part of the secondary storage, such as a hard disk or a solid-state drive, as an extension of the main memory, such as RAM.
- Virtual memory creates an illusion that the computer has more memory than it actually does and enables the execution of larger and more complex programs.

Some of the advantages of virtual memory are:

- It allows more processes to be maintained in the main memory, as only the required pages or segments of each process need to be loaded into memory. This leads to more efficient utilization of the processor and faster switching between processes.
- It provides memory isolation and protection, as each process operates in its own virtual address space and cannot access or modify the memory of other processes. This enhances the security and reliability of the system.
- It simplifies the programming and linking of applications, as the virtual address space is consistent and independent of the physical memory layout. This also enables the sharing of common libraries and code segments among different processes.

It allows the system to handle larger workloads and run more applications at once, as the virtual memory can be dynamically allocated and deallocated according to the demand. This also reduces the need for additional memory modules when the physical memory runs out.

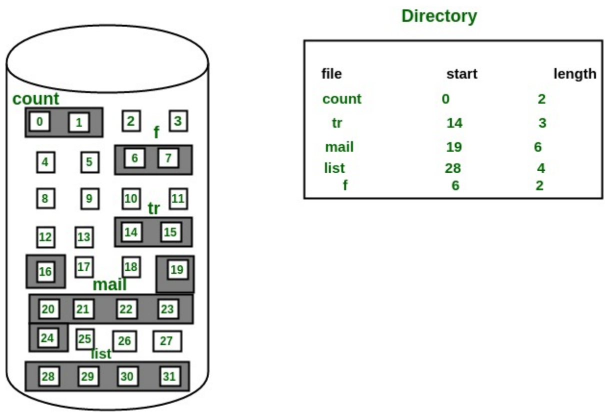
d) Explain about file attributes, file operations, and file types

Ans:

- File allocation methods are different ways of storing files on a disk, such that they can be accessed efficiently and securely by the operating system and the users.
- There are three main file allocation methods: contiguous, linked, and indexed.

i. Contiguous file allocation:
   - Each file occupies a contiguous (adjacent) set of blocks on the disk.
   - The directory entry for a file contains the address of the starting block and the length of the file (in terms of blocks).
   - It supports both sequential and direct access, as the address of any block of the file can be easily calculated by adding the block number to the starting address.

- It suffers from both internal and external fragmentation, as it is difficult to find a contiguous chunk of free blocks for a file, and the file size cannot be easily increased or decreased.
- It wastes disk space, as the file may not occupy the entire last block allocated to it.

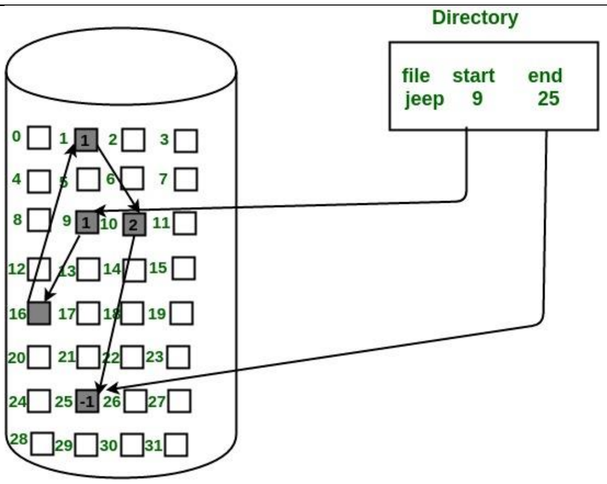


| file | start | length |
|---|---|---|
| count | 0 | 2 |
| tr | 14 | 3 |
| mail | 19 | 6 |
| list | 28 | 4 |
| f | 6 | 2 |

○

ii. Linked file allocation:

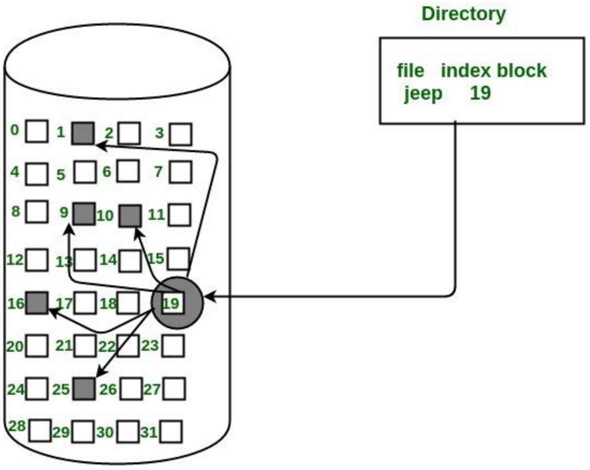
- Each file is a linked list of disk blocks, which need not be contiguous. The disk blocks can be scattered anywhere on the disk.
- The directory entry for a file contains a pointer to the first and the last block of the file. Each block contains a pointer to the next block occupied by the file. The last block contains a null pointer, indicating the end of the file.
- It is flexible in terms of file size, as the file can grow or shrink by adding or removing blocks from the linked list.
- It avoids external fragmentation, as any free block can be used to store a file block.
- It does not support direct access, as the blocks of the file have to be accessed sequentially by following the pointers.
- It incurs extra overhead for storing the pointers in each block and may cause disk failure if a pointer is corrupted or lost.



○

iii. Indexed file allocation:

- ○ Each file has a special block, called the index block, that contains the pointers to all the blocks occupied by the file. The index block can be either a single block or a linked list of blocks, depending on the size of the file.
- ○ The directory entry for a file contains a pointer to the index block of the file.
- ○ It combines the advantages of both contiguous and linked file allocation, as it supports direct access, avoids external fragmentation, and allows dynamic file size.
- ○ It also has some drawbacks, such as the extra overhead for storing the index block, the limitation on the maximum file size (depending on the number of pointers that can fit in the index block), and the possibility of disk failure if the index block is corrupted or lost.



○

e) Explain about Resource Allocation Graph (RAG).

|  |  |  |  |
|---|---|---|---|
|  | Ans: | • A Resource Allocation Graph (RAG) is a graphical representation of the state of a system in terms of processes and resources.<br>• It shows which resources are held by which processes and which processes are waiting for which resources.<br>• It consists of two types of vertices: process vertices (circles) and resource vertices (squares).<br>• It also consists of two types of edges: request edges (from processes to resources) and assignment edges (from resources to processes).<br>• It can be used to detect and avoid deadlock situations by identifying cycles in the graph.<br>• A deadlock occurs when a set of processes are waiting for each other in a circular chain, and none of them can proceed.<br>• To avoid deadlock, the system can use some strategies to prevent the formation of cycles in the RAG, such as resource ordering or the banker's algorithm. |  |
|  | f) | What are various features of Mobile and Real Time Operating Systems? |  |
|  | Ans: | Mobile and real-time operating systems are two types of operating systems that are designed for different purposes and devices. Mobile operating systems are specialized for mobile devices such as smartphones, tablets, and wearables. Real-time operating systems are specialized for embedded systems that require timely and predictable responses to events.<br><br>Some of the features of mobile operating systems are:<br><br>• They are optimized for low-power consumption, limited memory, and touch-screen interfaces.<br>• They provide a user-friendly graphical user interface (GUI) that allows users to interact with the device using icons, menus, windows, and other graphical elements.<br>• They support various applications and services that enhance the functionality and usability of the device, such as web browsers, email clients, social media apps, games, etc.<br>• They support various connectivity options, such as Wi-Fi, Bluetooth, cellular networks, GPS, NFC, etc.<br>• They support various security features, such as authentication, encryption, sandboxing, etc.<br><br>Some of the features of real-time operating systems are:<br><br>• They are designed to handle input and output data within a guaranteed time, whether it is in microseconds or minutes, depending on the task being performed.<br>• They are event-driven and preemptive, meaning they can monitor the priority of competing tasks and switch between them accordingly.<br>• They are deterministic, meaning they can produce the same output for the same input every time, regardless of the system load or external factors. |  |

- They are reliable, meaning they can handle errors and failures gracefully and recover quickly.
- They are scalable, meaning they can adapt to the changing requirements and constraints of the system.

**4** a) Suppose the head of a moving disk with 200 tracks, numbered 0 to 199, is Currently serving a request at track 143 and has just finished a request at track 125. If the queue of requests is kept in FIFO order: 86, 147, 91, 177, 94, 150, 102, 175, 130. What is the total head movement to satisfy these requests for the following Disk scheduling algorithms.
(a)FCFS
(b)SSTF
(c) C- SCAN

Ans: (a) FCFS (First Come First Serve): This algorithm executes the requests in the order they arrive in the disk queue. The total head movement is the sum of the absolute differences between the consecutive requests. The Gantt chart for FCFS is:

| **143** | **86** | **147** | **91** | **177** | **94** | **150** | **102** | **175** | **130** |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 57 | 64 | 56 | 86 | 83 | 56 | 48 | 73 | 45 |

The total head movement for FCFS is 57 + 64 + 56 + 86 + 83 + 56 + 48 + 73 + 45 = 568.

(b) SSTF (Shortest Seek Time First): This algorithm executes the requests based on their proximity to the current head position. It selects the request that requires the minimum head movement from the current position. The Gantt chart for SSTF is:

| **143** | **147** | **150** | **177** | **175** | **130** | **102** | **94** | **91** | **86** |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 4 | 3 | 27 | 2 | 45 | 28 | 8 | 3 | 5 |

The total head movement for SSTF is 4 + 3 + 27 + 2 + 45 + 28 + 8 + 3 + 5 = 125.

(c) C-SCAN (Circular SCAN): This algorithm moves the head from one end of the disk to the other, servicing the requests along the way. When the head reaches the other end, it jumps back to the beginning of the disk without servicing any requests on the return trip. The Gantt chart for C-SCAN is:

| **143** | **147** | **150** | **175** | **177** | **0** | **86** | **91** | **94** | **102** | **130** |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 4 | 3 | 25 | 2 | 23 | 86 | 5 | 3 | 8 | 28 |

The total head movement for C-SCAN is 4 + 3 + 25 + 2 + 23 + 86 + 5 + 3 + 8 + 28 = 187.

b) Consider the following five processes, with the length of the CPU burst time given in milliseconds. Process Burst time is P1-10, P2-29, P3-3, P4-7,P5-12. Consider the First come First serve (FCFS), Non Pre-emptive Shortest Job First(SJF), Round Robin(RR) (quantum=10ms) scheduling algorithms. Illustrate the scheduling using Gantt chart. Calculate the Average Waiting Time and Turn Around Time.

Ans: I will illustrate the scheduling using Gantt charts and calculate the average waiting time and turnaround time for each algorithm. The waiting time for a process is the amount of time that the process has to wait in the ready queue before it gets the CPU. The turnaround time for a process is the amount of time that the process takes to complete from its arrival to its completion, including the waiting time and the burst time.

- First come First serve (FCFS): This algorithm executes the processes in the order in which they arrive, i.e., the process that arrives first is executed first. It is non-preemptive, meaning that the process cannot be interrupted once it starts executing. The Gantt chart for FCFS is:

| **P1** | **P2** | **P3** | **P4** | **P5** |
|---|---|---|---|---|
| 0 | 10 | 39 | 42 | 49 |

The waiting time and turn around time for each process are:

| **Process** | **Waiting time** | **Turn around time** |
|---|---|---|
| P1 | 0 | 10 |
| P2 | 10 | 39 |
| P3 | 39 | 42 |
| P4 | 42 | 49 |
| P5 | 49 | 61 |

- The average waiting time is (0 + 10 + 39 + 42 + 49) / 5 = 28 ms.
- The average turnaround time is (10 + 39 + 42 + 49 + 61) / 5 = 40.2 ms.

- Non-Pre-emptive Shortest Job First (SJF): This algorithm executes the processes based on their burst time, i.e., the process with the shortest burst time is executed first. It is also non-preemptive, meaning that the process cannot be interrupted once it starts executing. The Gantt chart for SJF is:

| **P3** | **P4** | **P1** | **P5** | **P2** |
|---|---|---|---|---|
| 0 | 3 | 10 | 20 | 32 |

The waiting time and turn around time for each process are:

| **Process** | **Waiting time** | **Turn around time** |
|---|---|---|
| P1 | 10 | 20 |
| P2 | 32 | 61 |
| P3 | 0 | 3 |
| P4 | 3 | 10 |
| P5 | 20 | 32 |

- The average waiting time is (10 + 32 + 0 + 3 + 20) / 5 = 13 ms.
- The average turnaround time is (20 + 61 + 3 + 10 + 32) / 5 = 25.2 ms.

- Round Robin (RR) (quantum=10ms): This algorithm executes the processes in a circular order, giving each process a fixed amount of time (quantum) to execute. If the process does not finish within the quantum, it is preempted and added to the end of the ready queue. The Gantt chart for RR is:

| **P1** | **P2** | **P3** | **P4** | **P5** | **P1** | **P2** | **P5** | **P2** |
|---|---|---|---|---|---|---|---|---|
| 0 | 10 | 19 | 22 | 29 | 39 | 49 | 59 | 69 |

The waiting time and turnaround time for each process are:

| **Process** | **Waiting time** | **Turn around time** |
|---|---|---|
| P1 | 29 | 39 |
| P2 | 30 | 59 |
| P3 | 16 | 19 |
| P4 | 15 | 22 |
| P5 | 27 | 39 |

- The average waiting time is (29 + 30 + 16 + 15 + 27) / 5 = 23.4 ms.
- The average turnaround time is (39 + 59 + 19 + 22 + 39) / 5 = 35.6 ms.

c) What is semaphore and its types? How the classic synchronization problem - Dining philosopher is solved using semaphores?

Ans:

- **Semaphore** is a synchronization object that controls access to a shared resource in a concurrent system.
- It is a non-negative integer variable that is used to signal between processes or threads.
- Semaphores are of different types, including general semaphores, binary semaphores, counting semaphores, and strong semaphores.
- The **Dining Philosopher Problem** is a classic synchronization problem that involves a group of philosophers who sit around a circular table and share chopsticks.
- The problem is to design a solution that allows each philosopher to eat without causing a deadlock or a starvation.
- One solution to this problem is to use semaphores to represent the chopsticks.
- In this solution, each fork is represented by a semaphore, and a philosopher must acquire both the semaphore for the fork to their left and the semaphore for the fork to their right before they can eat.

|  |  |  |  |
|---|---|---|---|
|  |  | • This solution ensures that no two philosophers will be able to pick up the same chopstick at the same time, thus avoiding a deadlock.<br>OR<br>• Semaphore is a synchronization object that controls access to a shared resource in a concurrent system.<br>• Semaphores are of different types, including general semaphores, binary semaphores, counting semaphores, and strong semaphores.<br>• The Dining Philosopher Problem is a classic synchronization problem that involves a group of philosophers who sit around a circular table and share chopsticks.<br>• One solution to this problem is to use semaphores to represent the chopsticks.<br>• In this solution, each fork is represented by a semaphore, and a philosopher must acquire both the semaphore for the fork to their left and the semaphore for the fork to their right before they can eat.<br>• This solution ensures that no two philosophers will be able to pick up the same chopstick at the same time, thus avoiding a deadlock.<br>• Semaphore is an essential synchronization mechanism for concurrent systems.<br>• Semaphores can be used to control access to shared resources, coordinate the execution of multiple threads or processes, and avoid race conditions and deadlocks.<br>• There are several types of semaphores, each with its own advantages and disadvantages.<br>• The choice of semaphore type depends on the specific requirements of the application. |  |