FINAL SCHEME

02000CST206062207

			Total Pages: 15

Reg No.:_______________ Name:__________________________

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

FOURTH SEMESTER B.TECH DEGREE (R,S) EXAMINATION, JUNE 2023 (2019 SCHEME)

Course Code: CST 206

Course Name: OPERATING SYSTEMS

Max. Marks: 100 Duration: 3 Hours

PART A

	Answer all questions, each carries 3 marks.	Marks
1	• The creation and deletion of files • The creation and deletion of directories • The support of primitives for manipulating files and directories • The mapping of files onto secondary storage • The backup of files on stable (non-volatile) storage media	(3)
2	Booting process is done in 6 steps: o Loading of BIOS o POST i.e. power-on self-test o Loading of Operating System o System Configuration o Loading utilities o User Authentication.	(3)
3	**BUFFERING -** During direct or indirect communication, messages exchanged between communicating processes reside in a temporary queue which are implemented in the following three ways: **Zero capacity:** The queue has maximum length 0; thus, the link cannot have any message waiting in it. In this case, the sender must block until the recipient	(3)

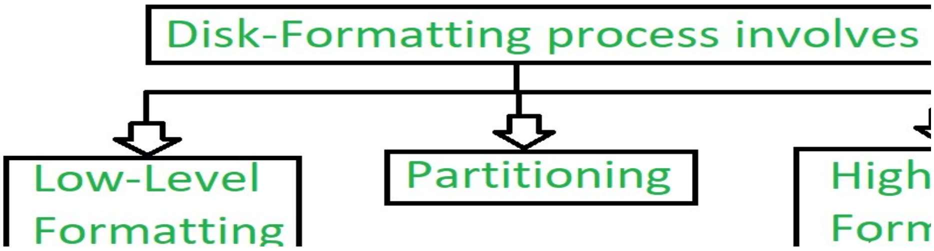
	receives the message. This is referred to as no buffering. **Bounded capacity:** The queue has finite length *n*; thus, at most *n* messages can reside in it. If the queue is not full when a new message is sent, the latter is placed in the queue (either the message is copied or a pointer to the message is kept), and the sender can continue execution without waiting. If the link is full, the sender must block until space is available in the queue. **Unbounded capacity:** The queue has potentially infinite length; thus, any number of messages can wait in it. The sender never blocks.	
4	In general, a multilevel feedback-queue scheduler is defined by the following parameters: • The number of queues. • The scheduling algorithm for each queue. • The method used to determine when to upgrade a process to a higher-priority queue. • The method used to determine when to demote a process to a lower-priority queue. • The method used to determine which queue a process will enter when that process needs service. Normally, when the multilevel queue scheduling algorithm is used, processes are permanently assigned to a queue when they enter the system. If there are separate queues for foreground and background processes, processes do not move from one queue to the other, since processes do not change their foreground or background nature. This setup has the advantage of low scheduling overhead, but it is inflexible. The **multilevel feedback-queue scheduling algorithm**, in contrast, allows a process to move between queues. **However, Q 4 may be considered as out of syllabus as the syllabus specifically names each of the scheduling algorithms to be covered and this does not include multilevel queue scheduling. Hence full credits may be awarded , if attempted.**	(3)
5	A solution to the critical-section problem must satisfy the following three requirements: **Mutual Exclusion:** If process Pi is executing in its critical section, then no other	(3)

	process can be executed in their critical sections. **Progress:** If no process is executing in its critical section and some processes wish to enter their critical sections, then only those processes that are not executing in their remainder section can participate in the decision on which will enter its critical section next, and this selection cannot be postponed indefinitely. **Bounded Waiting:** There exists a bound on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.	

6	The implementation of semaphore with a waiting queue may result in a satiation where two more processes are waiting indefinitely for an event that can be caused only by one of the waiting processes. When such a state is reached that process are said to be deadlocked.
	P0 P1
	wait(S); wait(Q);
	wait(Q); wait(S);
	. . . .
	signal(S); signal (Q);
	signal (Q); signal (S);
	P0 executes wait (S) and then P1 executes wait (Q). When p0 executes wait (Q), it must wait until p1 executes Signal (Q) in the same way P1 must wait until P0 executes signal (S). So p0 and p1 are deadlocked
		(3)

7	An address generated by the CPU is commonly referred to as a logical address, whereas an address seen by the memory unit is commonly referred to as a physical address. The compile-time and load-time address-binding schemes result in an environment where the logical and physical addresses are the same. The execution-time address-binding scheme results in an environment where the logical and physical addresses differ, in this case, we usually refer to the logical address as a virtual address. The set of all logical addresses generated by a program is referred to as a logical address space; the set of all physical addresses corresponding to these logical addresses is referred to as a physical address space.	(3)

8	**DYNAMIC LOADING:** Better memory-space utilization can be done by dynamic loading. With dynamic loading, a routine is not loaded until it is called. All routines are kept on disk in a re-locatable load format. The main program is loaded into memory and is executed. The advantage of dynamic loading is that an unused routine is never loaded. **DYNAMIC LINKING:** Most operating systems support only static linking, in which system language libraries are treated like any other object module and are combined by the leader into the binary program image. The concept of dynamic linking is similar to that of dynamic loading. Rather than loading being postponed until execution time, linking is postponed. This feature is usually used with system libraries, such as language subroutine libraries. With dynamic linking, a stub is included in the image for each library-routine reference. This stub is a small piece of code that indicates how to locate the appropriate memory-resident library routing.	(3)
9	1. Firstly, an internal table for this process to assess whether the reference was valid or invalid memory access. 2. If the reference becomes invalid, the system process would be terminated. Otherwise, the page will be paged in. 3. After that, the free-frame list finds the free frame in the system. 4. Now, the disk operation would be scheduled to get the required page from the disk. 5. When the I/O operation is completed, the process's page table will be updated with a new frame number, and the invalid bit will be changed. Now, it is a valid page reference. 6. If any page fault is found, restart these steps from starting.	(3)

10	 Disk-Formatting process involves Low-Level Formatting Partitioning High Forn	(3)

PART B

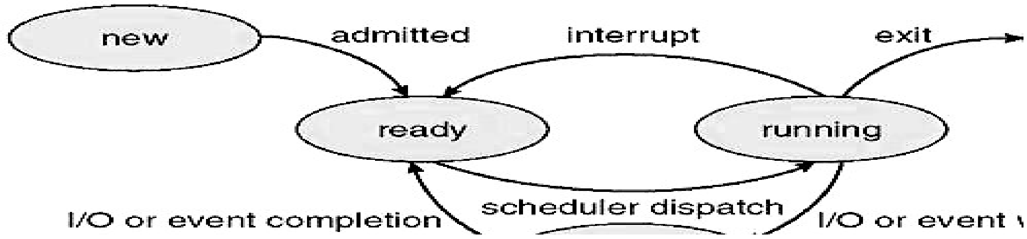
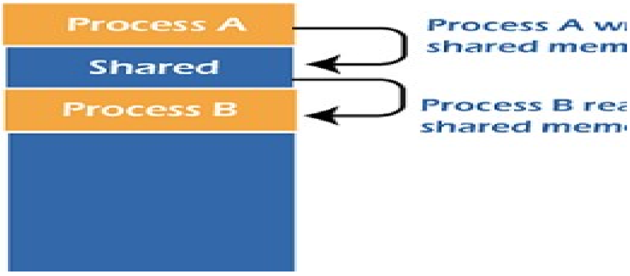
Answer any one question from each module

Module 1

11	(a)	**Operating System- definition – 1 mark** A program that acts as an intermediary between a user of a computer and the computer hardware Operating system goals: • Execute user programs and make solving user problems easier • Make the computer system convenient to use • Use the computer hardware in an efficient manner • Explanation of any 3 types of Operating System.- 3 X 2Marks 1. Batch processing systems 2. Time sharing systems 3. Multiprocessing systems 4. Real time systems 5. Distributed systems 6. Desktop systems 7. Handheld systems	(7)

		8. Clustered systems **Types of Operating System not mentioned in the syllabus. 6 marks may be given if attempted**	
	(b)	System call-explanation – 1 mark Three general methods used to pass parameters to the OS • Simplest: pass the parameters in registers • In some cases, may be more parameters than registers Parameters stored in a block, or table, in memory, and address of block passed as a parameter in a register. This approach taken by Linux and Solaris • Parameters placed, or pushed, onto the stack by the program and popped off the stack by the operating system Types of System Calls with examples – 3 Marks • Process control • File management • Device management • Information maintenance • Communications • Protection	(7)
OR			
12	(a)	Write notes on the following operating system structures. (i) Microkernel structure (ii) Simple Structure (iii) Layered Structure Explanation of each with diagram – 3 X 2 marks; Advantages and disadvantages – 2 Marks	(8)
	(b)	**The differences between symmetric and asymmetric multiprocessing – 3 marks** Symmetric Multiprocessing system: in this case each processor runs an identical copy of the OS, and hence they can communicate with each other as needed. Example: all modern OS (UNIX, LINUX, windows 7, 10). Asymmetric Multiprocessing system: master-slave concept. A master processor	(6)

		controls the system, the other processor either look to the master for instruction or have predefined task assigned. Example SunOS v4. Advantages of multiprocessor systems with explanation – 2 Marks 1.Increased throughput 2.economy of scale 3.reliability more Disadvantages with explanation - 1 Mark 1.Common computer bus, clock, memory and peripheral devices. 2.cost is more **Multiprocessor systems not mentioned in the syllabus. full credits may be given if attempted**	
Module 2			
13	(a)	Define process. - 3 marks Process can be defined as: • A program in execution. • A unit of activity characterized by the execution of a sequence of instructions, a current state, and an associated set of system resources. A process is an entity that consists of a number of elements. A process is more than the program code, which is sometimes known as the **text section**. It also includes the current activity, as represented by the value of the **program counter** and the contents of the processor's registers. A process generally also includes the process **stack**, which contains temporary data (such as function parameters, return addresses, and local variables), and a **data section**, which contains global variables. A process may also include a **heap**, which is memory that is dynamically allocated during process run time. Process state diagram with explanation – 4 Marks	(7)

		 new admitted interrupt exit ready running I/O or event completion scheduler dispatch I/O or event w	
	(b)	Diagram + Illustration – 3 Marks  Process A Shared Process B Process A wı shared mem Process B rea shared mem A process creates a shared memory segment using **shmget()**. The original owner of a shared memory segment can assign ownership to another user with **shmctl()**. It can also revoke this assignment. Other processes with proper permission can perform various control functions on the shared memory segment using **shmctl()**. Once created, a shared segment can be attached to a process address space using **shmat()**. It can be detached using **shmdt()**. The attaching process must have the appropriate permissions for **shmat()**. Once attached, the process can read or write to the segment, as the permission requested in the attach operation allows. A shared segment can be attached multiple times by the same process. A shared memory segment is described by a control structure with a unique ID that points to an area of physical memory. The identifier of the segment is called the **shmid**. The structure definition for the shared memory segment control structures and prototypes can be found in **<sys/shm.h>**. Bounded-Buffer – Shared-Memory Solution – 4 Marks	(7)
OR			
14	(a)	**Preemptive SJF** **Gantt chart: - 1 mark**	(8)

P0 P2 P1 P3 P4 P0

0 10 35 75 105 150 215

Processing – 2 marks

Process ID	Arrival Time	Burst Time (ms)	Waiting Time (ms)	Turnaround Time (ms)
P0	0	75	140	215
P1	10	40	25	65
P2	10	25	0	25
P3	55	30	20	50
P4	95	45	10	55

Average waiting time = 39 – 0.5 mark

Average turnaround time=82 – 0.5 mark

Note : As the name of preemptive algorithm is not explicitly mentioned in the question, the following preemptive algorithms should be considered as correct answer and full marks should be given
1. Preemptive SJF (SRTF)
2. Preemptive Priority Scheduling
3. Round Robin scheduling with any assumed time slice

RR scheduling(Time quantum= 15ms)

Gantt chart: - 1 mark

P0 P1 P2 P0 P1 P2 P3 P0 P1 P4 P3 P0 P4 P0 P4

0 15 30 45 60 75 85 100 115 125 140 155 170 185 200 215

Processing – 2 marks

Process ID	Arrival Time	Burst Time (ms)	Waiting Time (ms)	Turnaround Time (ms)
P0	0	75	125	200
P1	10	40	75	115
P2	10	25	50	75
P3	55	30	70	100
P4	95	45	75	120

Average waiting time = 59 – 0.5 mark

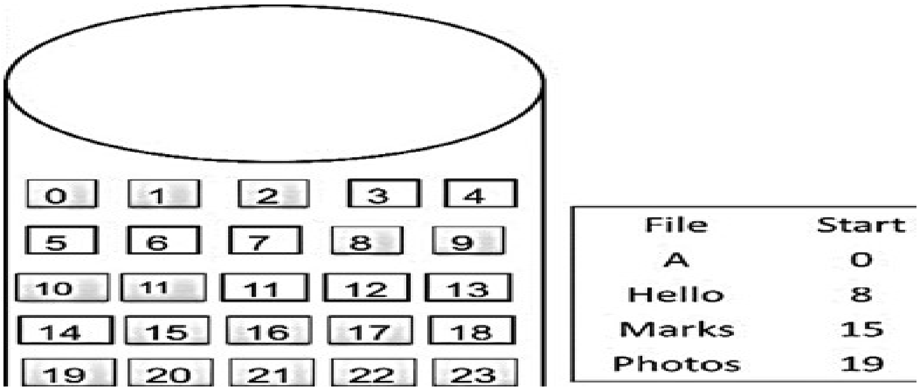
		Average turnaround time=122 – 0.5 mark	
	(b)	Threads – Definition – 1 Mark A thread is a basic unit of CPU utilization; it comprises a thread ID , a program counter, a register set, and a stack. It shares with other threads belonging to the same process its code section, data section, and other operating-system resources, such as open files and signals. A traditional (or heavyweight) process has a single thread of control. If a process has multiple threads of control, it can perform more than one task at a time. The benefits of multithreaded programming- listing and explanation – 4 Marks 1. Responsiveness. 2. Resource sharing. 3. Economy. 4. Scalability. List the ways of establishing relationship between user threads and kernel thread. - 1 mark Three common ways of establishing such a relationship: the many-to-one model, the one-to-one model, and the many-to-many model. **However, detailed study of thread is not expected to be covered in teaching plan. Second and third part of the question as an out of syllabus questions. Hence 2 marks may be given to explanation of threads and 4 marks may be awarded if advantages and ways of establishing relationship between user threads and kernel thread are attempted.**	(6)
Module 3			
15	(a)	Dining Philosophers Problem – 2 marks A solution for the problem using monitors- 4 marks	(6)
	(b)	Deadlock- def- 1mark The four necessary conditions for deadlock to occur with explanation- 3 Marks Various deadlock prevention mechanisms+ Description – 4 Marks	(8)
OR			
16	(a)	What is a semaphore? - 2 Marks A semaphore S is an integer variable that, apart from initialization, is accessed only through two standard atomic operations: wait () and signal() . The wait() operation was originally termed P (from the Dutch proberen, “to test”); signal() was originally called V (from verhogen, “to increment”). The definition of wait() is as follows:	(7)

		wait(S) {
		while (S <= 0) ; // busy wait
		S--; }
		The definition of signal() is as follows:
		signal(S) {
		S++; }
		Usage of semaphore- 2 Marks Counting semaphore; Binary semaphore
		Implementation semaphore – 3 Marks	

	(b)	2 2 3 0 3 2 0 0 0 3 2 3 2 5 0 7 2 0 0 1 Need Matrix - 2 Marks Inintial Work 3,0,0,1 Safe Sequence <P5,P1,P2,P3,P4> 5 Marks **Note : more than one safe sequence may be there marks can be given to correct answer**	(7)
		Module 4	
17	(a)	What are the physical addresses for the following logical addresses: a. 0, 430 219 + 430 = 649 b. 1, 10 2300 + 10 = 2310 c. 2, 500 Illegal address since size of segment 2 is 100 and the offset in logical address is 500. d. 3, 400 1327 + 400 = 1727 e. 4, 112 Illegal address since size of segment 4 is 96 and the offset in logical address is 112.	(5)
	(b)	i) LRU replacement ii) FIFO replacement iii) Optimal replacement Representation of demand paging with three frames – 3 X 2 marks No of page faults – 3 x 1 mark	(9)
		OR	
18	(a)	Define Demand Paging. - 2 MARKS	(6)

		A demand paging mechanism is very much similar to a paging system with swapping where processes stored in the secondary memory and pages are loaded only on demand, not in advance. During the program execution, if the program references a page that may not be available in the main memory because it was swapped, then the processor considers it as an invalid memory reference. That's because the page fault and transfers send control back from the program to the OS, which demands to store page back into the memory. How is swapping done – 1 mark Swapping is a memory management scheme in which any process can be temporarily swapped from main memory to secondary memory so that the main memory can be made available for other processes. It is used to improve main memory utilization. In secondary memory, the place where the swapped-out process is stored is called swap space. The concept of swapping has divided into two more concepts: Swap-in and Swap-out.- 2 Marks; Diagram – 1 mark	
	(b)	(i) First Fit (ii) Best Fit (iii) Worst Fit – 3 x 2marks- 6 marks i) First Fit P2 30K P1 60K P3 170K P4 100K 150K 300K 550K 400K 250K 200K P5 not possible to fit-EXTERNAL FRAGMENTATION; internal fragmentation – 30K, 60K (ii) Best Fit P2 30K P4 P5 200K P3 20K P1 10K 150K 300K 550K 400K 250K 200K NO external fragmentation; Internal fragmentation – 30K, 20K, 10K (iii) Worst Fit	(8)

		P1 P4 10 P2 280K K 150K 300K 550K 400K 250K 200K P3 and p5 cannot be fitted. External fragmentation Internal fragmentation - 10K Internal fragmentation and external fragmentation calculation – 2 marks	
		Module 5	
19	(a)	1. Single-level directory structure. 2. Two-level directory structure. ... 3. Tree Directory Structure. ... 4. Acyclic-Graph Directory Structure. 5. General graph Directory With figures – 5 marks **Full credits may be awarded if attempted, since directory structures is not explicitly mentioned in syllabus**	(5)
	(b)	There are three methods of file allocation:-3 x 3marks **1. Contiguous Allocation:** a. Contiguous allocation requires that each file occupy a set of contiguous blocks on the disk. The word contiguous means continuous. Because of contiguous allocation, there is minimal or no disk-head movement which reading/writing the blocks of the file. The seek time is minimal over here. Consequently, access time of a file and the I/O performance is greatly improved. To access a file, there we only need to know the starting location and length of the file which are stored in the directory as shown in the figure.	(9)



File	Start
A	0
Hello	8
Marks	15
Photos	19

It has certain problems associated with it:

- Finding the space for a new file (usually done with First fit and Best Fit Algorithms)
- Problem of external fragmentation. It occurs whenever free space is broken into tiny chunks.
- Compaction (which is a solution for fragmentation) can take up lot of time and may need system to be down, wherein normal operation will not be permitted.
- Determining space to be allocated for file, especially if it needs to grow.

Advantage:

- Contiguous allocation is easy to implement.

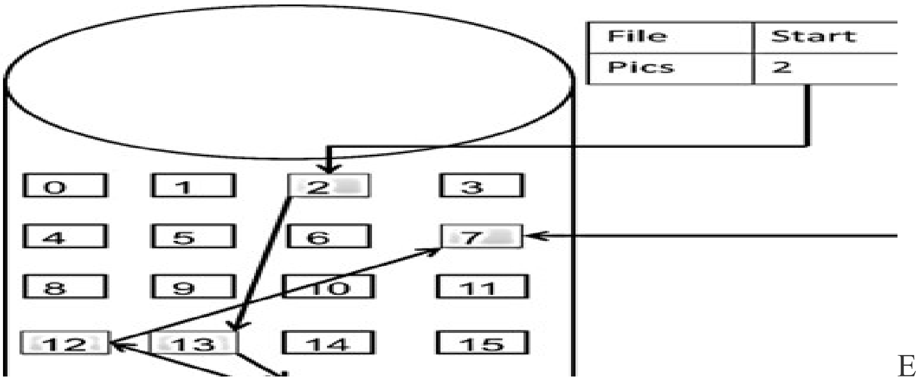
Disadvantage:

- It can be considered as a form of dynamic memory allocation, and external fragmentation may occur and compaction may be needed.
- It is difficult to estimate the file size. The size of a file may grow at run time and may be larger than the specified number of allocated blocks. In this case, the OS must move the blocks in order to provide mode space. In some systems, this is simply an error.

2. Linked Allocation:

With the linked allocation approach, disk blocks of a file are chained together with a linked-list. The directory entry of a file contains a pointer to the first block and a pointer to the last block.

To create a file, we create a new directory entry and the pointers are initialized to nil. When a write occurs, a new disk block is allocated and chained to the end of the list. This method solves the problems associated with contiguous allocation. Here the blocks of a single file can be scattered anywhere on the disk. The reason because the entire file is implemented as a Linked List. The directory maintained by the OS contains a pointer to the first and the last blocks of a file.



Each block of a file contains a pointer to the next block after it in the list. For creating a new file, we need to just create a new entry in the directory and not to search for sufficient space as in contiguous. The free space management system allocates space to a block for writing and is then appended to the end of the List. To Read a file, we need to follow the pointers from each block.

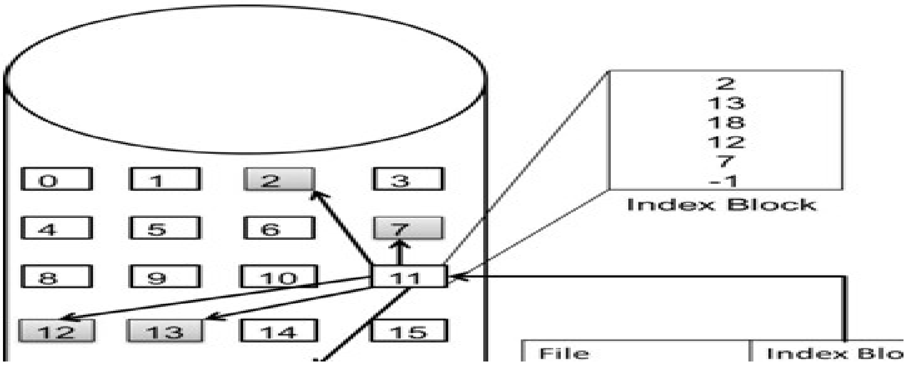
Advantages include no external fragmentation, size of file need not be declared at start, a file can grow as long as free blocks are available on disk.

Disadvantages: It works perfectly for Sequential access only, space needs to be allocated in block for pointers, error in pointer links can lead to Invalid read.

3. Indexed allocation:

Indexed Allocation With the contiguous allocation method, a user must indicate the file size before creating the file. Then, the operating system searches the disk to find contiguous disk blocks for the file. The directory entry is easy. It contains the initial disk address of this file and the number of disk blocks.

Each file has an index block that is an array of disk block addresses. The i-th

		entry in the index block points to the i-th block of the file. A file's directory entry contains a pointer to its index. Hence, the index block of an indexed allocation plays the same role as the page table. Index allocation supports both sequential and direct access without external fragmentation.  0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 2 13 18 12 7 -1 Index Block File Index Blo In indexed allocation method, all the pointers (pointing to the next block in the Linked list) are gathered together into one location known as Index Block. In the earlier method (i.e. Linked Allocation) the pointers along with the blocks were scattered across the disk and needed to be retrieved in order by visiting each block for access the file. This problem gets eliminated here. Each file has an index block of its own, which is an array of disk-block addresses. The kth entry of the index-block is a pointer to the kth block of the file. When a file is created initially, all pointers in the index block are set to null value. As new blocks are written, the pointers are modified accordingly. Indexed allocation supports direct access and does not suffer from any external fragmentation. Indexed allocation suffers from the problem of wasted space. E.g. if a file is made up of two blocks only, then a huge amount of space will be wasted.	
		OR	
20	(a)	File owner/creator should be able to control: ✦ what can be done ✦ by whom ■ Types of access ✦ Read ✦ Write ✦ Execute ✦ Append ✦ Delete ✦ List Access Lists and Groups ■ Mode of access: read, write, execute ■ Three classes of users RWX a) owner access 7 1 1 1 RWX b) group access	(6)

		6 1 1 0 RWX c) public access 1 0 0 1 ■ Ask manager to create a group (unique name), say G, and add some users to the group. ■ For a particular file (say game) or subdirectory, define an appropriate access. owner group public chmod 761 game Attach a group to a file chgrp G game	
	(b)	(i) FCFS -diagram - 53-98-183-41-122-14-124-65-67 Total head movements incurred while servicing these requests = (98 – 53) + (183 – 98) + (183 – 41) + (122 – 41) + (122 – 14) + (124 – 14) + (124 – 65) + (67 – 65) = 45 + 85 + 142 + 81 + 108 + 110 + 59 + 2 = 632 (ii) SSTF —diagram – 53-65-67-41- 14-98-122-124-183 Total head movements incurred while servicing these requests = (65 – 53) + (67 – 65) + (67 – 41) + (41 – 14) + (98 – 14) + (122 – 98) + (124 – 122) + (183 – 124) = 12 + 2 + 26 + 27 + 84 + 24 + 2 + 59 = 236 (iii) C-SCAN-diagram - 53-65-67-98-122-124-183-199-0-14-41 Total head movements incurred while servicing these requests = (65 – 53) + (67 – 65) + (98 – 67) + (122 – 98) + (124 – 122) + (183 – 124) + (199 – 183) + (199 – 0) + (14 – 0) + (41 – 14) = 12 + 2 + 31 + 24 + 2 + 59 + 16 + 199 + 14 + 27 = 386 **Note: direction of head movement is not mentioned , so marks can be given either disk head moving towards higher address or lower address,** (iv) LOOK-diagram- 53-65-67-98-122-124-183-41-14 Total head movements incurred while servicing these requests = (65 – 53) + (67 – 65) + (98 – 67) + (122 – 98) + (124 – 122) + (183 – 124) + (183 – 41) + (41 – 14) = 12 + 2 + 31 + 24 + 2 + 59 + 142 + 27 = 299 **Note: direction of head movement is not mentioned , so marks can be given either disk head moving towards higher address or lower address**	(8)