GID - 2109230289

142

12

1

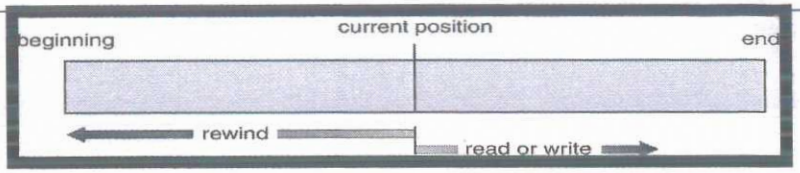
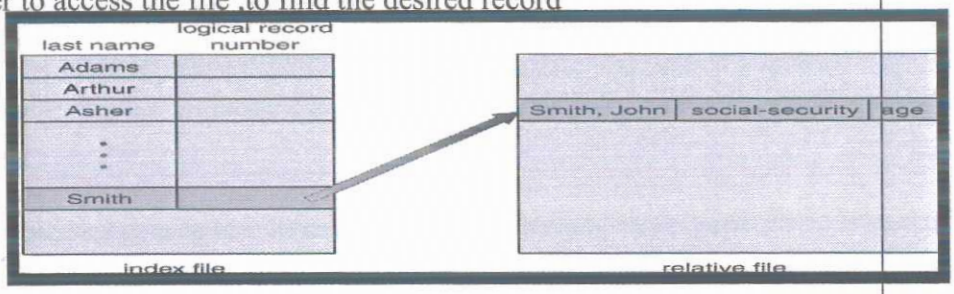
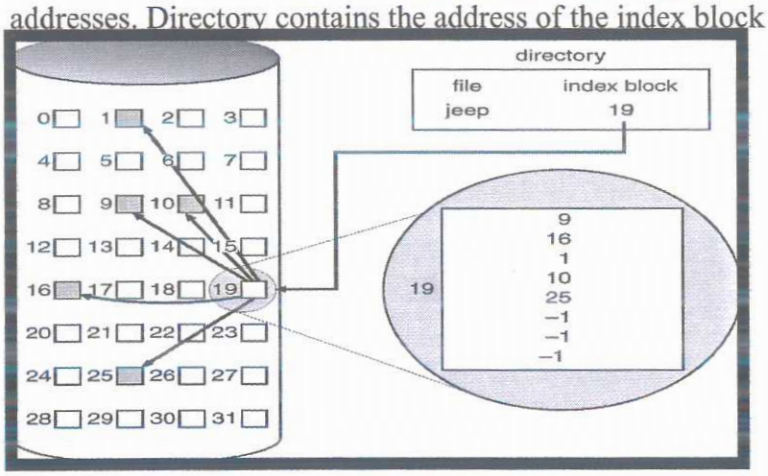
SCHEME OF EVALUATION
(Scoring Indicators)

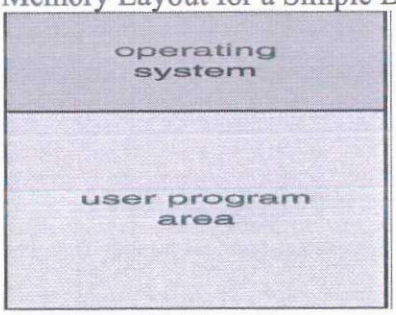
Revision: 21 **Course Code: 5132**

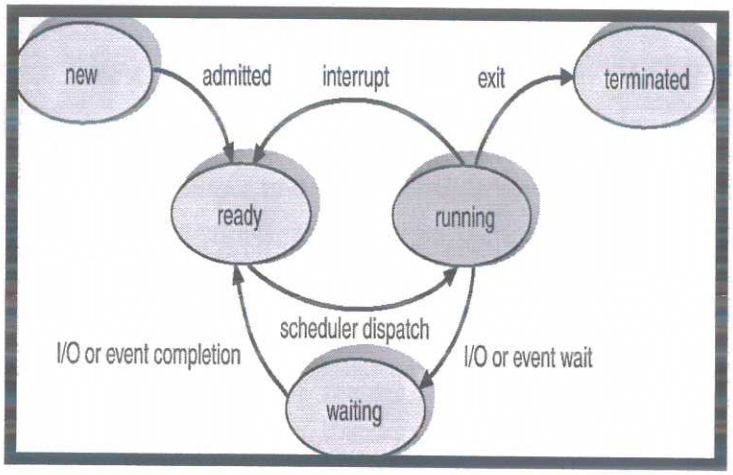
Course Title: OPERATING SYSTEM

Qst. No.	Scoring Indicator	Split up score	Sub Total	Total
	PART A			
1	Assembler			1
2	Any two examples each carries ½ mark	½ x 2 =1		1
3	Queue which store set of all processes in the system			1
4	Process control Block			1
5	Priority scheduling			1
6	Address binding			1
7	First fit/best fit			1
8	File allocation table			1
9	File			1
II 1.	**COMPILER** / **INTERPRETER** It takes an entire program at a time. / It takes a single line of code or instruction at a time. It generates intermediate object code. / It does not produce any intermediate object code. The compilation is done before execution. / Compilation and execution take place simultaneously. Comparatively faster / Slower Display all errors after compilation, all at the same time. / Displays error of each line one by one. Difficult / Easier comparatively Gcc, g++, javac / Python, jvm	Any three comparisons each carries 1 mark	3x1	3
II 2.	○ **Allocation** : the space for program is allocated in the main memory, by calculating the size of the program. ○ **Loading** – brings the object program into memory for execution. ○ **Relocation** – modifies the object program so that it can be loaded at an address different from the location originally specified. ○ **Linking**, which combines two or more separate object programs and supplies the necessary information.	Any three functions	3 x 1	3
II 3.	1. Mutual Exclusion 2. Hold and wait 3. No preemption 4. Circular wait	Any three conditions	3 x 1	3
II 4.	• CPU utilization • Throughput • Turnaround time • Waiting time • Response time	Any three 1 mark for each		3

	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.			
II 5.	1. Compile time 2. Run time 3. Link time	1 mark for each		3
II 6	• No internal fragmentation • Improved memory utilization • Provides virtual memory • Dynamic linking and loading • Facilitate shared segments • Infer controlled access • less overhead (smaller tables) compared to variable partitioning	Any three 1 mark for each		3
II 7.	**Contiguous Memory Allocation** → The main memory must accommodate both the operating system and the various user processes. → The memory is usually divided into two partitions: one for the resident operating system and one for the user processes. → Main memory usually into two partitions: – Resident operating system, usually held in low memory with interrupt vector. – User processes then held in high memory. → In contiguous memory allocation, each process is contained in a single section of memory that is contiguous to the section containing the next process. → Relocation registers used to protect user processes from each other, and from changing operating-system code and data → Base register contains value of smallest physical address → Limit register contains range of logical addresses – each logical address must be less than the limit register → MMU maps logical address *dynamically* → Can then allow actions such as kernel code being **transient** and kernel changing size	3		3
II.8	Different file Organizations 1. Sequential Access • In this method, information is processed in order, one after the other. • File pointer advances after read operation. • Write appends to the end of the file.	1 mark	1+2	3

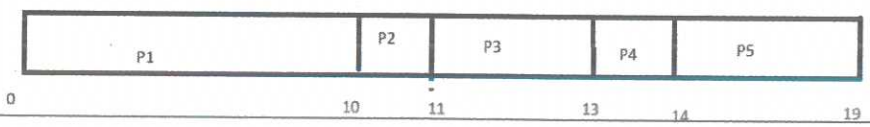
	 2. Indexed Access • The index contains pointers to the various blocks. • To find a record in the file, we first search the index, and then use the pointer to access the file ,to find the desired record 	2 marks		
II.9	**Indexed Allocation** • Indexed allocation bringing all the pointers together into one location: the index block. • Each file has its own index block, which is an array of disk-block addresses. Directory contains the address of the index block 	Explanation- 1 Figure:2 mark	1+2	3
II.10	File Operations 1. CREATING A FILE: Step1: space in the file system must be found for the file. Step2: an entry for the new file must be made in the directory. 2. WRITING A FILE: system call is made with the name of the file and the information to be written to the file 3. READING A FILE: system call is made with the name of the file and where the next block of the file should be put 4. REPOSITIONING WITHIN A FILE: The directory is searched for	Any three Each carries 1 mark	3	3

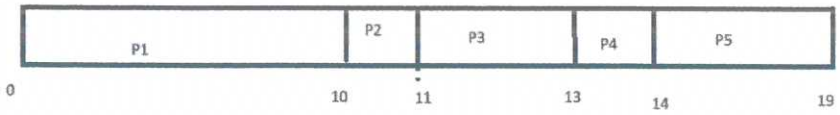
	the appropriate entry, and the current-file-position is set to a given value 5. DELETING A FILE: The directory is searched for the named file. Having found the associated directory entry, release all file space and the directory entry 6. TRUNCATING A FILE: To erase the contents of a file but keep its attributes			
III.1	1. File management 2. Memory management 3. Process management 4. Security 5. Device management 6. Input output management 7. User interface			7
III.2	**Batch processing system** • The user has to submit a job (written on cards or tape) to a computer operator. • Then computer operator places a batch of several jobs on an input device. • Jobs are batched together by type of languages and requirement. • Then a special program, the monitor, manages the execution of each program in the batch. • The monitor is always in the main memory and available for execution. • Following are some disadvantages of this type of system : • Zero interaction between user and computer. No mechanism to prioritize processes. Memory Layout for a Simple Batch System  **Multiprocessor system** A multiprocessor system consists of several processors that share a common physical memory. Multiprocessor system provides higher computing power and speed. In multiprocessor system all processors operate under single operating system. Multiplicity of the processors and how they do act together are transparent to the others. Following are some advantages of this type of system. • Enhanced performance	4 marks		7

	• Execution of several tasks by different processors concurrently, increases the system's throughput without speeding up the execution of a single task. • If possible, system divides task into many subtasks and then these subtasks can be executed in parallel in different processors. Thereby speeding up the execution of single tasks.	3 marks		
III.3	Process States ★ As a process executes, it changes its state  ✪ NEW : The process is being created. ✪ READY : The process is waiting to be assigned to a processor. ✪ RUNNING : Instructions are being executed. ✪ WAITING : The process is waiting for some event to occur. ✪ TERMINATED : The process has finished execution	Figure -2 Each stage carries 1 mark	2+5	7
III.4	Process Synchronization is the task of coordinating the execution of processes in a way that no two processes can have access to the same shared data and resources. On the basis of synchronization, processes are categorized as one of the following two types: • Independent Process: The execution of one process does not affect the execution of other processes. • Cooperative Process: A process that can affect or be affected by other processes executing in the system. Process synchronization problem arises in the case of Cooperative process also because resources are shared in Cooperative processes. Race condition Several processes access and manipulate the same data concurrently and	3 marks 2 marks		7

	the outcome of the execution depends on the particular order in which the access takes place, is called a race condition.	2 marks		

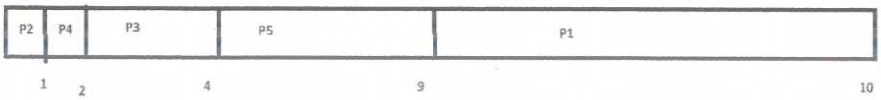
III.5	**Critical Section Problem**
	Consider a system consisting of n processes {P0, P1, ..., Pn−1}.
	Each process has a segment of code, called a critical section in which the process may be accessing and updating data that is shared with at least one other process.
	When one process is executing in its critical section, no other process is allowed to execute in its critical section. That is, no two processes are executing in their critical sections at the same time.
	The critical-section problem is to design a protocol that the processes can use to synchronize their activity so as to cooperatively share data.
	Each process must request permission to enter its critical section. The section of code implementing this request is the entry section.
	The critical section may be followed by an exit section.
	The remaining code is the remainder section.
	The general structure of a typical process is
	`while (true) {`
	entry section
	critical section
	exit section
	remainder section
	`}`
	A solution to the critical-section problem must satisfy the following three requirements:
	1. Mutual exclusion. If process Pi is executing in its critical section, then no other processes can be executing in their critical sections.
	2. 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 sections can participate in deciding which will enter its critical section next, and this selection cannot be postponed indefinitely.
	3. Bounded waiting. There exists a bound, or limit, 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.
		Explanation-5 marks
		
		Structure - 2 marks
				7

III.6	FCFS Gantt chart 			7



Waiting time for p1 =0
Waiting time for p2 =10
Waiting time for p3 =11
Waiting time for p4=13
Waiting time for p5=14
Average waiting time = (0+10+11+13+14)/5 = 9.6

SJF
Gantt chart



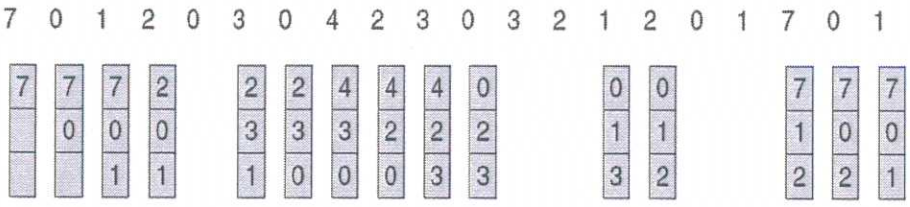
Waiting time for p1=9
Waiting time for p2=0
Waiting time for p3=2
Waiting time for p4=1
Waiting time for p5=4
Average waiting time = (9+0+2+1+4)/5= 3.2

III.7

FIFO Page Replacement

- The simplest page-replacement algorithm is a ***first-in, first-out (FIFO)*** algorithm.
- A FIFO replacement algorithm associates with each page the time when that page was brought into memory.
- When a page must be replaced, the oldest page is chosen.
- Create a FIFO queue to hold all pages in memory.
- Replace the page at the head of the queue. When a page is brought into memory, insert it at the tail of the queue.
- Reference string: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1
- 3 frames (3 pages can be in memory at a time per process)

reference string



page frames

Disadvantage

Belady's anomaly: For some page-replacement algorithms, the page-fault rate may *increase* as the number of allocated frames increases.

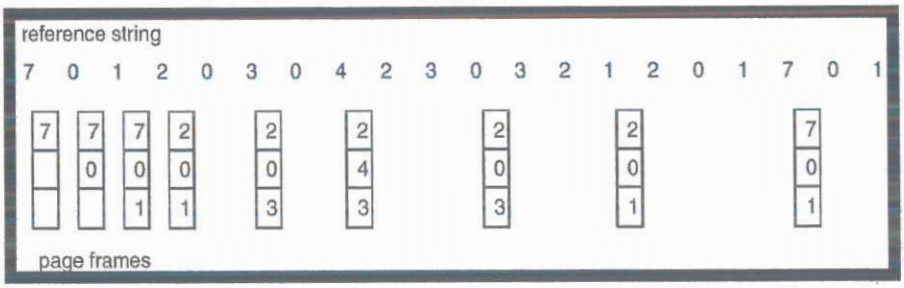
Optimal Page Replacement

- Replace page that will not be used for longest period of time

3 marks

7

to implement, because it requires future knowledge of the reference string.

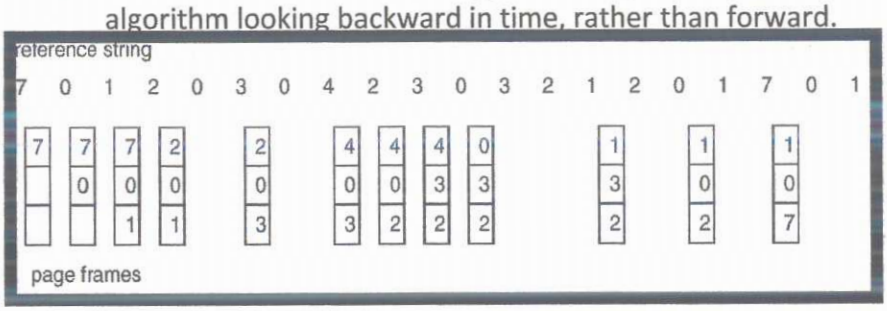


2 marks

LRU Page Replacement

- The key distinction between the FIFO and OPT algorithms is that the FIFO algorithm uses the time when a page was brought into memory, whereas the OPT algorithm uses the time when a page is to be used.
- LRU replacement associates with each page the time of that page's last use.
- When a page must be replaced, LRU chooses the page that has not been used for the longest period of time.
- We can think of this strategy as the optimal page-replacement algorithm looking backward in time, rather than forward.

2 marks



III.8 **Paging**

→ Paging is a memory-management scheme that permits the physical address space of a process to be non-contiguous.

→ Paging avoids external fragmentation and the need for compaction

→ Physical memory is broken into fixed sized blocks called frames.

→ Logical memory is also broken into blocks of the same size called pages.

→ When a process is to be executed, its pages are loaded into available memory frames from the backing store.

→ The backing store is divided into fixed sized blocks that are of the same size as the memory frames.

Paging- 3 marks

7

Paging Hardware

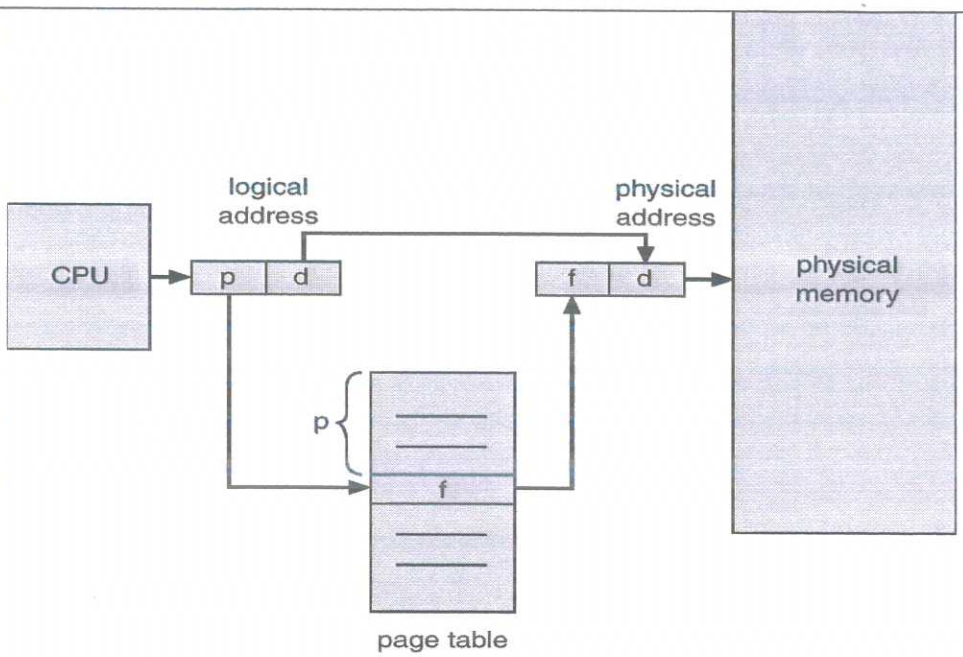


Figure-3marks

- Every address generated by the CPU is divided into two parts:
  1. Page number, p *(no: of pages)*
  2. Page offset, d *(size of the pages)*

- Page number is used as an index into the page table. The page table contains the base address of each page in physical memory.

1 mark

Logical address is as follows:

Page number	page offset
P	D

III.9

Fragmentation

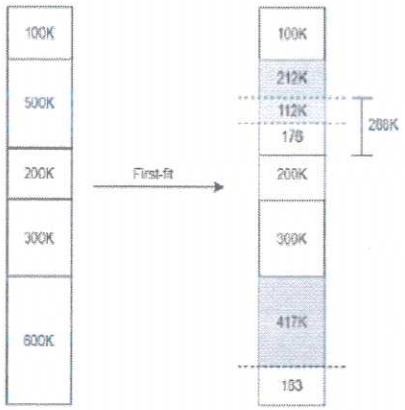
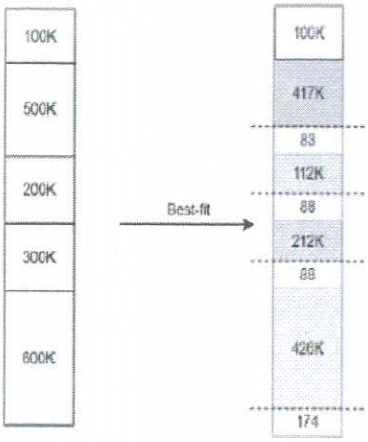
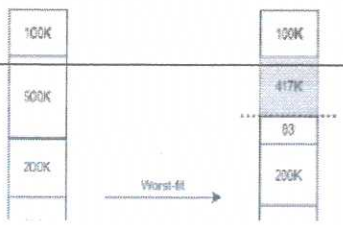
- Both the first-fit and best-fit strategies for memory allocation suffer from external fragmentation.
- As processes are loaded and removed from memory, the free memory space is broken into little pieces. External fragmentation exists when there is enough total memory space to satisfy a request but the available spaces are not contiguous: storage is fragmented into a large number of small holes. This fragmentation problem can be severe. In the worst case, we could have a block of free (or wasted) memory between every two processes. If all these small pieces of memory were in one big free block instead, we might be able to run several more processes.
- As processes are loaded and removed from memory, the free memory space is broken into little pieces.
- External fragmentation exists when there is enough total

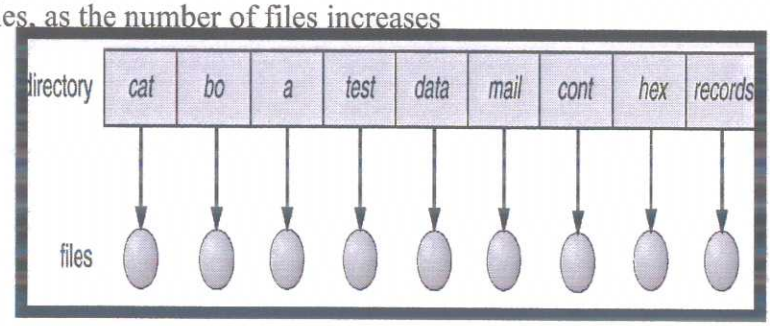
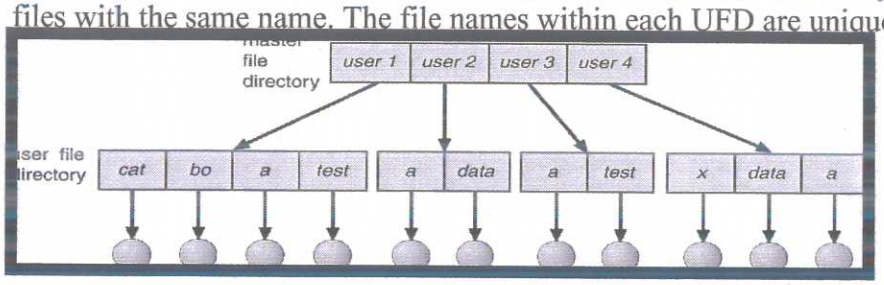
Fragmentation-3 marks

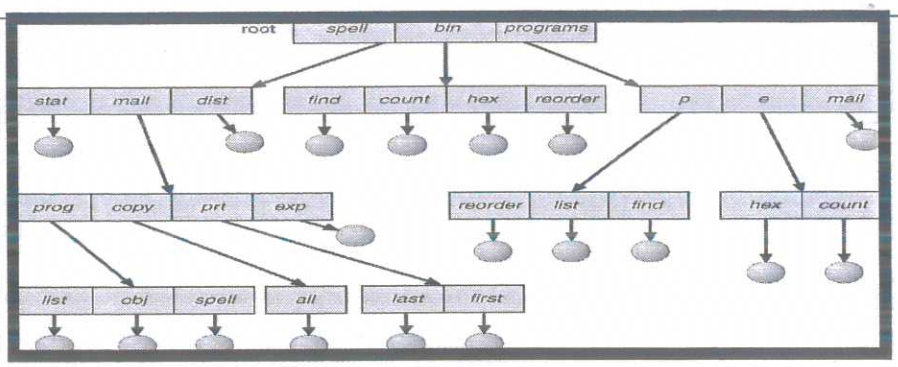
Internal fragmentation-2 marks

7

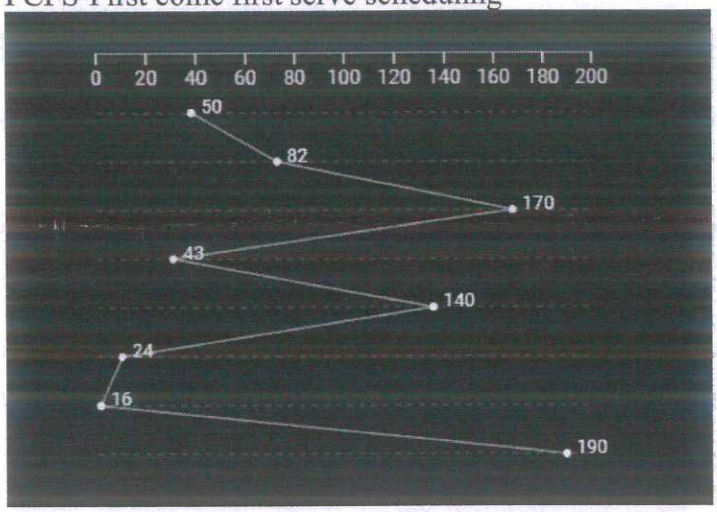
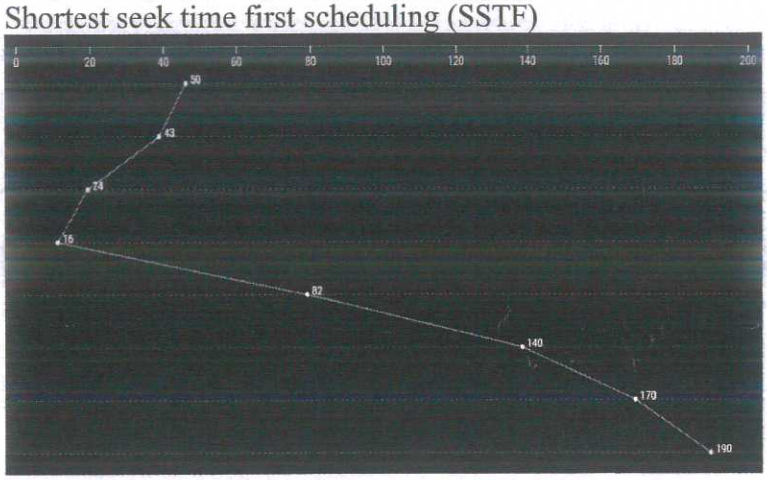
	memory space to satisfy a request but the available spaces are not contiguous: storage is fragmented into a large number of small holes.
	• **External fragmentation** – we could have a block of free (or wasted) memory between every two processes. If all these small pieces of memory were in one big free block instead, we might be able to run several more processes. Total memory space exists to satisfy a request, but it is not contiguous.
	• **Internal fragmentation** – in fixed size partition, allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used.
	• Compaction- external fragmentation can be reduced by compaction
	  • Shuffle memory contents to place all free memory together in one large block.
	  • Compaction is possible only if relocation is dynamic, and is done at execution time.
	Another possible solution to the external-fragmentation problem is: **Paging and Segmentation**
		External fragmentation- 2marks		

III.10	 **First-fit** 212K is put in the 500K partition. 417K is put in the 600K partition. 112K is put in the 288K partition. 426K must wait.  **Best-fit** 212K is put in the 300K partition. 417K is put in the 500K partition. 112K is put in the 200K partition. 426K is put in the 600K partition.  **Worst-fit** 212K is put in the 600K partition. 417K is put in the 500K partition.	2 marks 2 marks		7

		3 marks		
III.1 1	**Single Level Directory** All files are contained in the same directory. Single-level directory has significant limitations - when the number of files increases or when the system has more than one user. It is difficult to remember the names of all the files, as the number of files increases  **Two Level Directory** Each user has her own user file directory (UFD).Each UFD has a similar structure, but lists only the files of a single user. Different users may have files with the same name. The file names within each UFD are unique.  **Tree Structured Directory** Allowing the user to define his own subdirectories to impose a structure on his files. Separate directories for files associated with different topics or different forms of information. Users can access, in addition to their files, the files of other users	2 2	2+2+3	7



3

III.12	FCFS-First come first serve scheduling  Total head movement =(82-50)+(170-82)+(170-43)+(140-43)+(140-24)+(24-16)+(190-16) =642 1. Shortest seek time first scheduling (SSTF)  Total head movement =208	Figure-2 Calculation-1 Fig- 2 marks Cal - 2marks	3 4	7