Pradip Dhungana
201751 - BESE (Day)
Assignment - 3

Q1) Explain the difference between internal & external fragmentation.

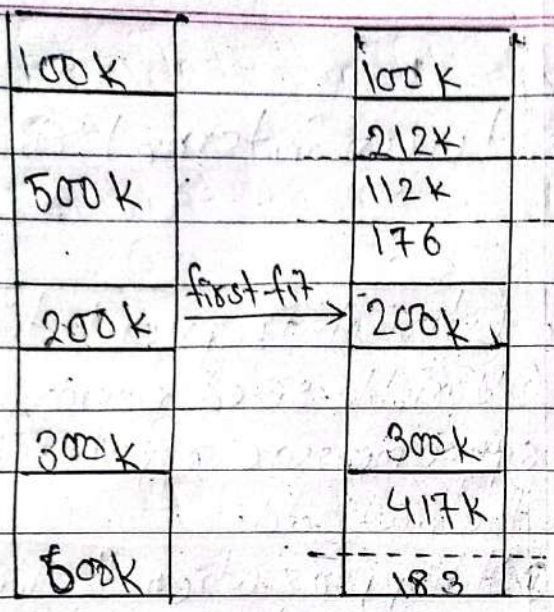
=>

Internal fragmentation	External fragmentation
(i) fixed-sized memory, blocks square measure appointed to process.	(i) Variable-sized memory block square measure appointed to the method.
(ii) It happens when the method or process is smaller than the memory.	(ii) It happens when the method or process is removed.
(iii) The solution of internal fragmentation is the best-fit block.	(iii) The solution to external fragmentation is paging.
(iv) It occurs when memory is divided into fixed-sized partitions	(iv) It occurs when memory is divided into variable size partitions based on the size of processes.
(v) It occurs with paging & fixed partitioning.	(v) It occurs with segmentation & dynamic partitioning.
(vi) Best fit block search is the solution.	(vi) Available memory blocks are non-contiguous.

Q2) Given five memory partitions of 100KB, 500KB, 200KB, 300KB, & 600KB (in order), how would each of the first-fit, best-fit & worst-fit algorithm place processes of 212KB, 417KB, 112KB, & 426KB (in order)? Which algorithm makes the most efficient use of memory?

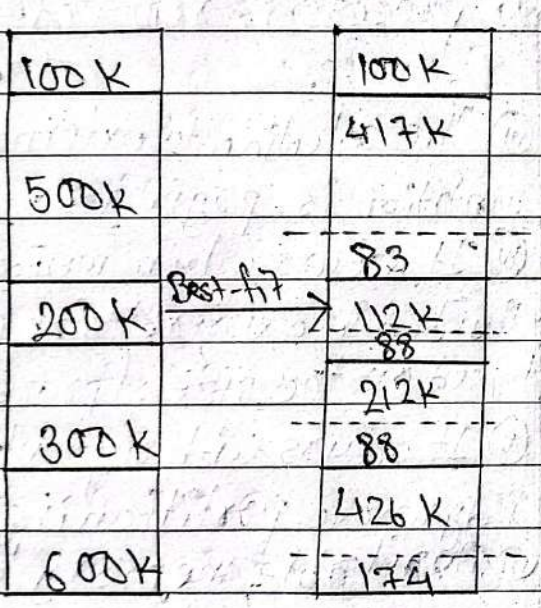
Date
Page

1)



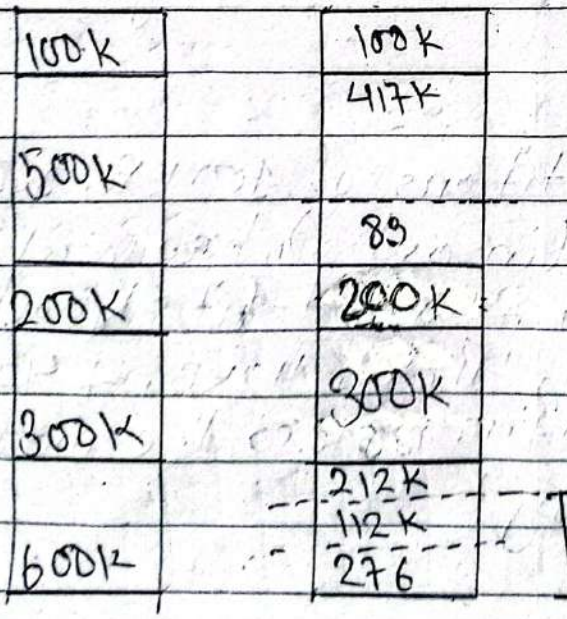
① First-fit

212k is put in the 500k partition.
417k is put in the 600k partition
112k is put in the 288k partition
(500-212 = 288k new partition)
426k must wait.



② 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



③ Worst-fit

212k is put in the 600k partition
417k is put in the 500k partition
112k is put in the 388k partition
426k must wait.

In this case, best-fit makes the most efficient use of memory.

Q/3) Why are segmentation & paging sometimes combined into one scheme?

=> Segmentation & paging are often combined in order to improve upon each other. Segmented paging is helpful when the page table becomes very large. A large contiguous section of the page table that is unused can be collapsed into a single segment table entry with a page table address of zero. Paged segmentation handles the case of having very long segments that require a lot of time for allocation. By paging the segments we reduce wasted memory due to external fragmentation as well as simplify the allocation.

Some more advantages include:

(i) Efficient memory allocation & utilization due to fixed-size pages.
(ii) logical division & modular structure provided by segments
(iii) Flexibility in managing code & data within a process.
(iv) Reduced external fragmentation as the pages are of same size.

Q4) Consider the following segment table:

Segment	Base	length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical addresses for the following logical addresses?

(a) 0, 430
(b) 1, 10
(c) 2, 500
(d) 3, 400
(e) 4, 112

=> (a) 0, 430
219 + 430 = 649

(b) 1, 10
2300 + 10 = 2310

(c) 2, 500
Illegal address since size of segment 2 is 100 & 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 & the offset in logical address is 12.

Q5) What is the purpose of paging the page tables?

=> The purpose of paging the page table is given below:

1) To allow a computer's operating system to efficiently manage memory. This is achieved by dividing it into smaller, fixed-sized units called pages.

2) This method of memory management is more efficient & flexible than using a contiguous block of memory for every process.

3) Page table play a crucial role in this process. They are used to map virtual memory addresses to physical memory addresses.

4) When a program accesses a memory location, the virtual memory address is looked up in the page table. This lookup process determines the corresponding physical memory address.

5) An important aspect of paging the page tables is

that it allows the computer to keep only the ne parts of the page table in physical memory at given time.

6) This is particularly important because page tables be quite large. keeping the entire table in phy memory would be wasteful of resources.

7) When a process needs to access a memory locati that is not is physical memory, the page table en for that location indicates that a page fault has occured.

Q) 6) What is the minimum number of page faults f an optimal page replacement strategy for the following reference string with four page frames? A repeat this problem for LRU, Second Chance, FIFO NRU, MFU.

1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2

=> ① Optimal page Replacement: (for 3 page frames)

Here, let frame size = 3,
Initially all frame are empty.

1	2	3	4	5	3	4	1	6	7	8	7	8	9	7
1	1	1	1	5	No page fault	No page fault	5	5	5	5	No page fault	No page fault	9	No page fault
	2	2	4	4			4	4	4	8			8	
		3	3	3			1	6	7	7			7	

9	5	4	5	4	2
No page fault	9	9	No	No	2
	8	4	Page	Page	4
	5	5	faut	faut	5

Total page fault = 13 (for 3 page frames)
Total Hit = 9

(11) LRU (Least Recently Used) :-

Here,
Reference string:
1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2

frame size = 4

1	2	3	4	5	3	4	1	6	7	8	7	8	9
1	1	1	1	5	No	No	5	6	6	6	No	No	6
	2	2	2	2	Page	Page	1	1	1	1	Page	Page	9
		3	3	3	fault	fault	3	3	7	7	fault	fault	7
			4	4			4	4	4	8			8

7	8	9	5	4	5	4	2
No	No	No	5	5	No	No	5
Page	Page	Page	9	9	Page	Page	9
fault	fault	fault	7	4	fault	fault	4
			8	8			2

Total page fault = 13
Total Hit = 9

(III) SCA (Second Chance Algorithm) :-

frame size = 4

R.S.

1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2.

1	2	3	4	5	3	4	1	6	7	8	7	8	9	7
1(0)	1(0)	1(0)	1(0)	5(0)	5(0)	5(0)	5(0)	6(0)	6(0)	6(0)	6(0)	6(0)	6(0)	6(0)
	2(0)	2(0)	2(0)	2(0)	2(0)	2(0)	1(0)	1(0)	7(0)	7(0)	7(1)	7(1)	7(1)	7(1)
		3(0)	3(0)	3(0)	3(1)	3(1)	3(1)	3(1)	3(0)	8(0)	8(0)	8(0)	9(0)	9(0)
			4(0)	4(0)	4(0)	4(1)	4(1)	4(1)	4(0)	8(0)	8(0)	8(0)	8(0)	8(0)

8	9	5	4	5	4	2
6(0)	6(0)	5(0)	4(0)	4(0)	4(1)	4(1)
7(1)	7(1)	7(1)	7(0)	5(0)	5(0)	5(0)
9(0)	9(1)	9(1)	9(0)	9(0)	9(0)	9(0)
8(1)	8(1)	8(1)	8(0)	8(0)	8(0)	2(0)

(IV) FIFO (first In first Out) :-

frame size = 4

R.S = 1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2

1	2	3	4	5	3	4	1	6	7	8	7	8	9	7	8	9	5
1	1	1	1	5	No	No	5	5	5	8	No	No	8	No	No	No	
	2	2	2	2	Page	Page	1	1	1	1	Page	Page	9	Page	Page	Page	
		3	3	3	fault	fault	3	6	6	6	fault	fault	6	fault	fault	fault	
			4	4			4	4	7	7			7				

4	5	4	2
8	No	No	8
9	Page	Page	9
5	fault	fault	5
4			4

Total Page fault = 9
Total Hit = 13

(9u) MFU (Most frequently Used) :

R.S: 1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4,
frame size = 4

1	2	3	4	5	3	4	1	6
1	1	1	1	5	No	No	5	5
	2	2	2	2	Page	Page	1	1
		3	3	3	fault	fault	3	6
			4	4			4	4

7	8	7	8	9	7	8	9	5	4
5	5	No	No	5	5	5	No	No	5
1	8	Page	Page	8	7	8	Page	Page	4
6	6	fault	fault	6	6	6	fault	fault	6
7	7			9	9	9			9

~~Counter~~	Counter
1	+2
2	+2
3	+2
4	+2+3+4
5	+2+3
6	1
7	+2+3
8	+2+3
9	+2

5	4	2
9	No	No	5
	Page	Page	2
	fault	fault	6
			9

∴ Total Page fault = 8
Total Hit = 14

(vi) NRU (Not Recently Used):-

R.S = 1,2,3,4,5,3,4,1,6,7,8,7,8,9,7,8,9,5,4,5,4,2.
frame size = 4

1	2	3	4	5	3	4	1	6	7	8	7	8	9
1	1	1	1	5	No	No	5	5	5	8	No	No	8
	2	2	2	2	Page	Page	1	1	1	1	Page	Page	9
		3	3	3	fault	fault	3	6	6	6	fault	fault	6
			4	4			4	4	7	7			7

7	8	9	5	4	5	4	2
No	No	No	8	8	No	No	2
Page	Page	Page	9	9	Page	Page	9
fault	fault	fault	5	5	fault	fault	5
			7	4			4

Total Page fault = 13
Total Hit = 9

Q7) What is the cause of thrashing? How does the system detect thrashing? Once it detectes thrashing, what can the system do to eliminate this problem

⇒ Thrashing is a condition that occurs when a system spends more time servicing page faults than executing transactions. This happens when a system's virtual memory resources are overused, leading to a constant state of paging & page faults.

Causes:
Under-allocation of the minimum number of pages required by a process, forcing it to continuously page fault.

High degree of multiprogramming:
↳ If CPU utilization is too low, the degree of multiprogramming is increased by introducing a new process to the system. A global page replacement algorithm is used. The CPU scheduler sees the decreasing CPU utilization & increase the degree of multiprogramming

Detection of Thrashing:
↳ The system can detect thrashing by evaluating the level of CPU utilization as compared to the level of multiprogramming. If the page fault rate is too high, it indicates that the process has too few frames allocated to it.

Elimination of Thrashing:
Increase the amount of RAM.
Decrease the number of programs being run on the computer.
Adjust the size of the swap file.

Q)8) A computer has four page frames. The time of loading, time of last access, & the R & M bits for each page are shown below (the times are in clock ticks):

Page	Loaded	Last Ref.	R	M
0	126	280	1	0
1	230	265	0	1
2	140	270	0	0
3	110	285	1	1

(a) Which page will NRU replace?
(b) Which page will FIFO replace?
(c) Which page will LRU replace?
(d) Which page will second chance replace?

a) According to NRU, page 2 will replaced. Because since the result is randomly selected for pages with dirty-bit low & referenced-bit low (a page of the lowest class). If that class wasn't available, then it will replace not-referenced & dirty first & select a random page out of that class.

b) According to FIFO, Page 1 will be replaced. Because It's the first one in the list.

c) LRU, the page which is used most earlier in the present memory block will be replaced.

According to LRU, Page 2 will be replaced because it is the oldest un-used page.

d) The second chance will change page 0 (R=0) & replace page 1 because its R value is already 0.

(a) Consider a file system that uses a modified contiguous-allocation scheme with support for extents. A file is a collection of extents, with each extent corresponding to a contiguous set of blocks. A key issue in such system is the degree of variability in the size of the extents. What are the advantages & disadvantages of the following schemes?

(a) All extents are of the same size, & the size is predetermined.

(b) Extents can be of any size & are allocated dynamically.

(c) Extents can be of a few fixed sizes, & these sizes are predetermined.

⇒ 2) All extents are of the same size, & the size is predetermined.

↳ Advantages:

i) Simplifies the block allocation scheme.

ii) A simple bit map or free list for extent would suffice.

iii) Blocks are preallocated & pre-formatted specifically for the data base when they are created.

Disadvantages:
i) It might not be flexible enough to handle files of varying sizes.
ii) It could lead to internal fragmentation if the file sizes do not align well with the extent size.

(b) Extents can be of any size & are allocated dynamically.

↳ Advantages:
i) More flexible as it can accomodate files of any size.
ii) Can result in less file fragmentation.
iii) Enables efficient use of memory space.

↳ Disadvantages:
i) Requires more complex schemes
ii) It might be difficult to find an extent of the appropriate size.
iii) Could lead to external fragmentation.

(c) Extents can be of a few fixed sizes, & these sizes are predetermined.

↳ Advantages:
i) Offers a balance between flexibility & complexity.
ii) Can handle files of varying sizes more efficiently than a system with extents of the same size.

↳ Disadvantages:
i) Requires maintaining a separate bitmap or free list for each possible size.
ii) It might still lead to internal fragmentation if the file size do not align well with the available extent sizes.

10) Fragmentation on a storage device could be eliminated by recompaction of the information. Typical disk devices do not have relocation or base registers (such as are used when memory is to be compacted), so how can we relocate files? Give three reasons why recompacting & relocation of files often are avoided.

⇒ Relocating & recompacting files on a storage device are often avoided for following reasons:
1) Overhead:
↳ Relocating files involves reading data blocks from their original locations into the main memory & then writing them back to their new locations. This

process requires additional I/O operations, which can be time-consuming & degrade system performance.

2) limited applicability:
↳ Relocation registers are typically used for sequential files, but many disk files are not sequential. As a result, relocating & recompacting files may not be effective for all types of files.

3) Non-contiguous allocation:
↳ Many new files do not require contiguous disk space, & even sequential files can be allocated non-contiguous blocks if links between logically sequential blocks are maintained by the disk system. Therefore, recompacting & relocating files may not be necessary for many files.

12) What are the advantages & disadvantages of supporting memory mapped I/O to device control registers?

⇒ Advantages:
① Memory-mapped I/O uses the same address space for both memory & I/O device control registers. This eliminates the need for special I/O instructions in the CPU, simplifying the architecture.

(iii) It doesn't require enforcing protection rules that prevent user programs for executing I/O instructions, as these are now regular memory access instructions.

Disadvantages:
↳ while this method provide flexibility, It needs to be managed carefully. The memory addresses linked to the device control registers must be protected from user program access to ensure system stability & security. This requires careful design of the memory.

Q11) Suppose that a disk drive has 5000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 143, & the previous request was at cylinder 125. The queue of pending requests, in FIFO order is

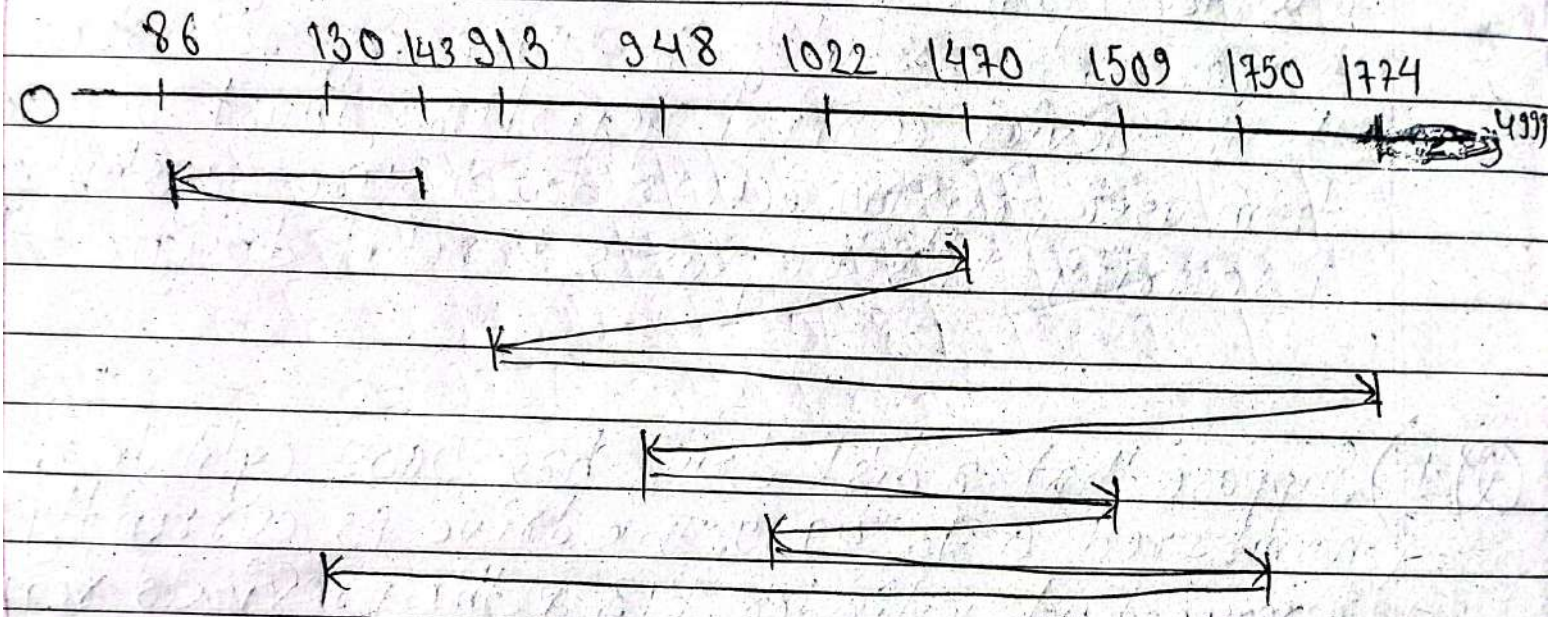
86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests, for each of the following disk-scheduling algorithms?

(a) FCFS (b) SSTF (c) SCAN (d) C-SCAN

Here,
Disk size = 5000 cylinders.
Head pointer = 143
Previous request = 125
Request Queue: 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

FCFS:

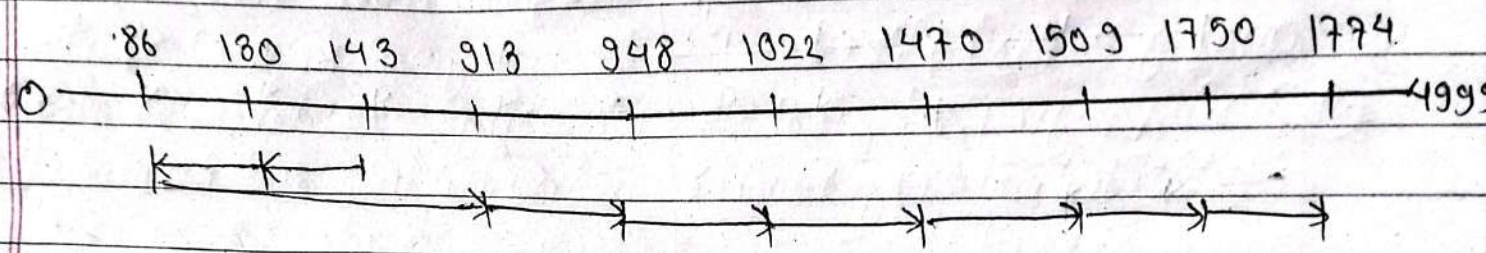


Seek order: 143, 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

∴ Seek distance = |143-86| + |86-1470| + |1470-913| + |913-1774| + |1774-948| + |948-1509| + |1509-1022| + |1022-1750| + |1750-130|

= 7081 cylinders

(II) SSTF:

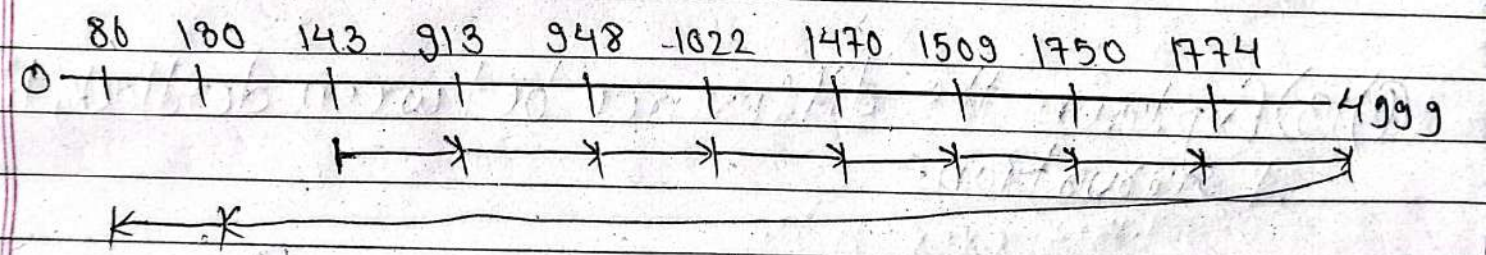


Seek order: 143, 130, 86, 913, 948, 1022, 1470, 1509, 1750, 1774

∴ Seek distance = |143-130| + |130-86| + |86-913| + |913-948| + |948-1022| + |1022-1470| + |1470-1509| + |1509-1750| + |1750-1774|

= 1745 cylinders #

(III) SCAN:

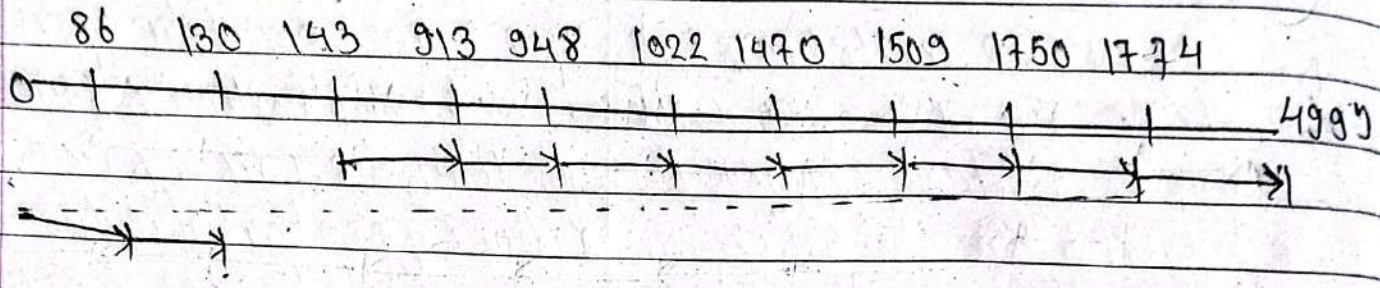


Seek order: 143, 913, 948, 1022, 1470, 1509, 1750, 1774, 4999, 130, 86

∴ Seek distance = |143-913| + |913-948| + |948-1022| + |1022-1470| + |1470-1509| + |1509-1750| + |1750-1774| + |1774-4999| + |4999-130| + |130-86|

= 9769 cylinder.

(12) C-SCAN:



Seek order: 143, 913, 948, 1022, 1470, 1509, 1750, 1774, 4999, 0, 86, 130

Seek distance: |143-913| + |913-948| + |948-1022| + |1470-1509| + |1750-1774| + |1774-4999| + |4999-0| + |0-86| + |86-130|
= 9296 cylinders. #

Q13) Explain the difference between deadlock, livelock & starvation.

⇒

Characteristics	Deadlock	Starvation	Livelock
Definition	Two or more processes waiting for each other to release resources, creating a standstill.	A process unable to acquire necessary resources due to other processes monopolizing the	Two or more processes continuously changing their state in response to the state of the other processes, but no progress is made.
Causes	~~Block~~ Poor resource management & coordination bet^n processes,	Long-running process monopolizing resources, lack of proper resource	Poor coordination bet^n processes, lack of proper syncronization.

	multiple processes accessing shared resources simultaneously	allocation, insufficient time slice.	
Result	Blocked processes, decreased performance, potential system failure.	Delayed or blocked processes, decreased performance, potential system failure.	No progress is made, continuous state changes, decreased performance.
Solution	Deadlocks can be resolved by using methods such as detection & recovery, prevention & avoidance	starvation can be resolved by using priority scheduling, aging, & resource allocation.	Livelocks can be resolved by using methods such as breaking symmetry, randomness, & timeouts

14) Differentiate betn Segmentation & Paging.

=>

Paging	Segmentation
(i) In paging, a process address space is broken into fixed sized blocks called pages.	(i) In segmentation, a process address space is broken in varying sized blocks called sections.
(ii) OS divides the memory into pages	(ii) Compiler is responsible to calculate the segment size, the virtual address & actual address.
(iii) Page size is determined by hardware	(iii) Section size is determined by user
(iv) Paging technique is faster in terms of memory access.	(iv) Segmentation is slower than paging

(v) Paging can cause internal fragmentation as some pages may go underutilized.	(v) Segmentation can cause external fragmentation as some memory block may not be used at all.
(vi) During paging, a logical address is divided into page number & page offset	(vi) During segmentation, a logical address is divided into section number & section offset.

Q)15) Explain producer Consumer Problem using semaphore.

⇒ Producer Process:

The code that defines the producer process is given below:

```
do {
    .
    .PRODUCE ITEM
    .
    wait(empty);
    wait(mutex);
    .
    PUT ITEM IN BUFFER
    .
    signal(mutex);
    signal(full);
} while(1);
```

In the above code, mutex, empty & full are semaphores. Here mutex is initialized to 1, empty is initialized to n (maximum size of the buffer) and full is initialized to 0.

The mutex semaphore ensures mutual exclusion. The empty & full semaphores count the number of empty & full spaces in the buffer.

After the item is produced, wait operation is carried out on empty. This indicates that the empty space in the buffer has decreased by 1. Then wait operation is carried out on mutex so that consumer process cannot interfere.

After the item is put in the buffer, signal operation is carried out on mutex and full. The former indicates that consumer process can now act and the latter shows that the buffer is full by 1.

Consumer Process:

The code that defines the consumer process is given below:

```
do{
    wait(full);
    wait(mutex);

    REMOVE ITEM FROM BUFFER

    signal(mutex);
    signal(empty);
```

CONSUME ITEM

}while(1);

↳ The wait operation is carried out on full. This indicates that items in the buffer have decreased by 1. Then wait operation is carried out on mutex so that producer process cannot interfere.

↳ Then the item is removed form buffer. After that, signal operation is carried out on mutex & empty. The former indicates that consumer process can now act & the latter shows that the empty space in the buffer has increased by 1.