NEPAL COLLEGE OF INFORMATION TECHNOLOGY

Balkumari, Lalitpur

Affiliated to Pokhara University

ASSIGNMENT FOR OPERATING SYSTEM

ASSIGNMENT 3

Submitted by:
Name: Arpan Adhikari
Roll No.: 231309
BE-CE (3rd SEM)

Submitted to:
Amit Shrivastava

Name : Arpan Adhikari
Roll No. : 231309
BECE (Day)
OS

Assignment 3

Q1) Explain the difference between internal and external fragmentation.

	Aspect	Internal	External
1.	Definition	Unused memory within allocated partitions.	Unused memory outside allocated partitions.
2.	cause	Allocated memory block is larger than required by the process.	Small, non-contiguous free memory blocks scattered throughout RAM.
3.	Occurrence	Happens inside an allocated block	Happens outside allocated blocks.
4.	Memory Loss	Wastes memory within fixed partitions.	Wastes memory as small gaps cannot be utilized by processes.
5.	Environment	Common in fixed-size partitioning schemes.	Common in dynamic memory allocation schemes.

6	Impact	Minor wastage per partition	significant wastage if gaps accumulate and cannot be consolidated.
7.	solution	Use dynamic partitioning or allocate exact memory needed.	Use techniques like compaction or paging to reduce gaps.

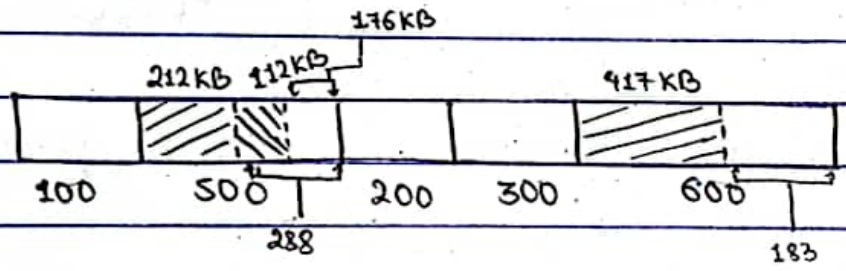
Q2)

Given,

Memory Partitions = 100KB, 500KB, 200KB, 300KB, 600KB

Process size = 212 KB, 417 KB, 112 KB, 426 KB

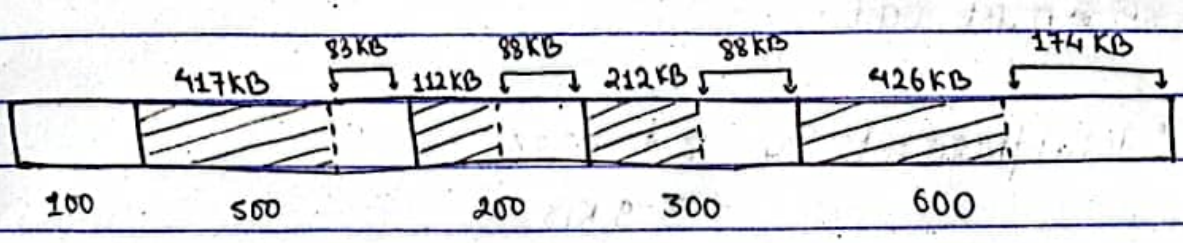
(i) First Fit



- 212 KB is placed in 500KB
  ∴ Remaining size = 500 - 212 = 288 KB
- 417 KB is placed in 600KB
  ∴ Remaining size = 600 - 417 = 183 KB
- 112 KB is placed in 288 KB
  ∴ Remaining size = 288 - 112 = 176 KB
- 426 KB must wait.

∴ Total fragmentation = 176 + 183
= 359 KB

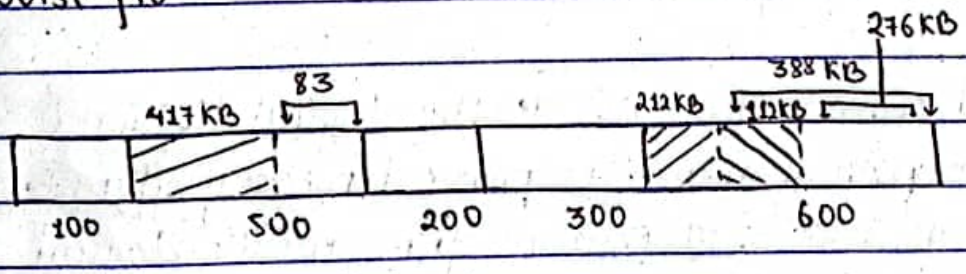
(ii) Best-fit



- 212 KB is placed in 300 KB.
∴ Remaining size = 300 - 212 = 88 KB
- 417 KB is placed in 500 KB.
∴ Remaining size = 500 - 417 = 83 KB
- 112 KB is placed in 200 KB.
∴ Remaining size = 200 - 112 = 88 KB
- 426 KB is placed in 600 KB.
∴ Remaining size = 600 - 426 = 174 KB

∴ Total fragmentation = 88 + 83 + 88 + 174
= 433 KB

(iii) Worst-fit



- 212 KB is placed in 600 KB.
∴ Remaining size = 600 - 212 = 388 KB

- 417 KB is placed in 500 KB.
  ∴ Remaining size = 500 - 417 = 83 KB
- 112 KB is placed in 388 KB.
  ∴ Remaining size = 388 - 112 = 276 KB
- 426 KB must wait.

∴ Total fragmentation = 83 + 276
= 359 KB

As all the process are placed in memory, Best-fit algorithm makes the most efficient use of memory.

Q3) Why are segmentation and paging sometimes combined into one scheme?

→

Segmentation and paging are combined to take advantage of their strengths and overcome their individual limitations. segmentation divides a program into logical units (like code, data, and stack), making it easier to manage and provide protection. On the other hand, paging breaks memory into fixed-size blocks, eliminating external fragmentation and simplifying memory allocation.

When combined, segmentation handles the logical division of programs, while paging manages physical memory efficiently, reducing both internal and external fragmentation. Logical addresses in this scheme consist of a segment number and an offset. This segment is further is divided into pages, and the system translates this into physical memory

addresses. This approach provides better memory utilization, enhanced protection, and scalability, making it ideal for modern operating system.

Q4)

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

What are physical address for following logical addresses!

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

Hint:
Logical Address → (segment, offset)

a) (0,430)
Segment 0 has a length of 600, and offset 430 is valid.
Physical Address = 219 (Base) + 430 (offset)
= 649 Ans

b) (1,10)
segment 1 has length of 14, so offset 10 is valid.

Physical Address = 2300 (Base) + 10 (offset)
= 2310 Ans

c) 2,500
segment 2 has a length of 100, and offset 500 exceeds the length.

So, this case is illogical (invalid Address).

d) 3,400
segment 3 has a length of 580, and offset 400 is valid.

So, Physical Address = 1327 (Base) + 400 (offset)
= 1727 Ans

e) (4,112)
segment 4 has a length of 96, and offset 112 exceeds the length.

so, this case is illogical (invalid Address).

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

→

The purpose and benefits of paging the page tables are as follows:

~~(i) Reducing Memory Overhead~~

(i) Efficient Use of Memory
The page table can be very large for processes with large address spaces. So, paging allows page table to be divided into smaller, more manageable chunks.

(ii) Handling large Page tables
Paging the page table helps in handling the large number of pages by breaking it into smaller parts.

(iii) On-Demand loading of Page Table Entries
Only parts of the page table need to be loaded into memory, reducing the overhead of storing the entire table in physical memory at all times.

(iv) Hierarchical Page Tables
Paging the page table leads to the creation of multi-level page tables, where each level is a page table itself. This hierarchial structure reduces the memory overhead associated with storing large, flat page tables.

(v) Scalability
It allows the virtual memory system to scale efficiently for processes with very large address spaces, supporting modern applications

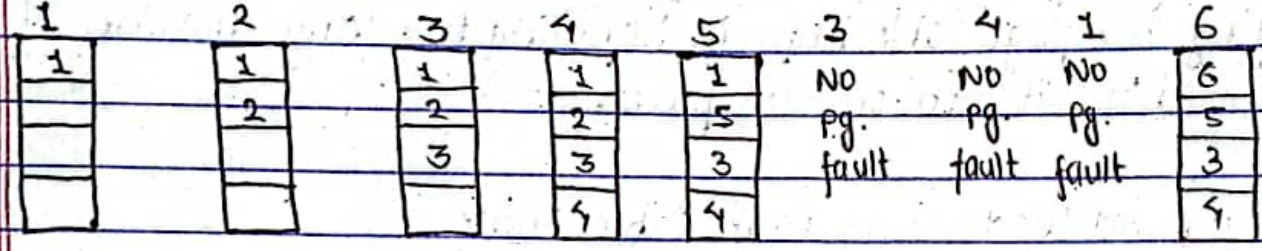
without requiring excessive physical memory.

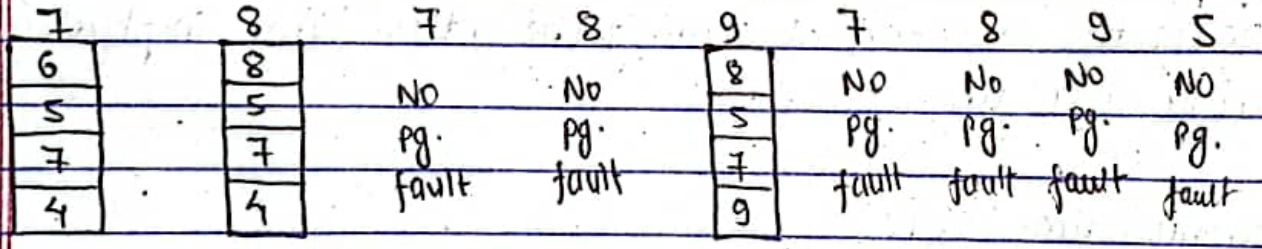
Q6) What is the minimum number of page faults for an optimal page replacement strategy for the following reference string with four page frames?
Now, repeat this problem for LRU, second chance, FIFO, MFU.

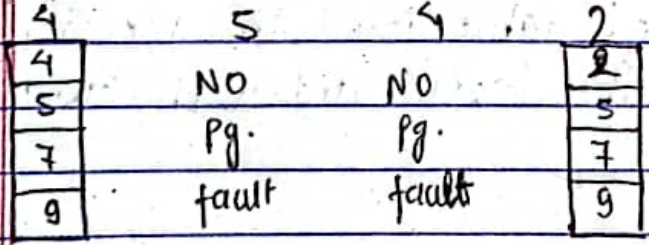
Sol?

Page frame = 4

(i) Optimal Page Replacement







∴ Total page fault = 11
Total hit = 11

(ii) LRU (Least Recently Used)

1	2	3	4	5	3	4	1	6
1	1	1	1	5	No	No	5	6
	2	2	2	2	Pg.	Pg.	1	1
		3	3	3	fault	fault	3	3
			4	4			4	4

7	8	7	8	9	7	8	9	5
6	6	No	No	6	No	No	No	5
1	1	Pg.	Pg.	9	Pg.	Pg.	Pg.	9
7	7	fault	fault	7	fault	fault	fault	7
4	8			8				8

4	5	4	2
5	No	No	5
9	Pg.	Pg.	9
4	fault	fault	4
8			2

∴ Total page fault = 13
Total hit = 9

(iii) FIFO

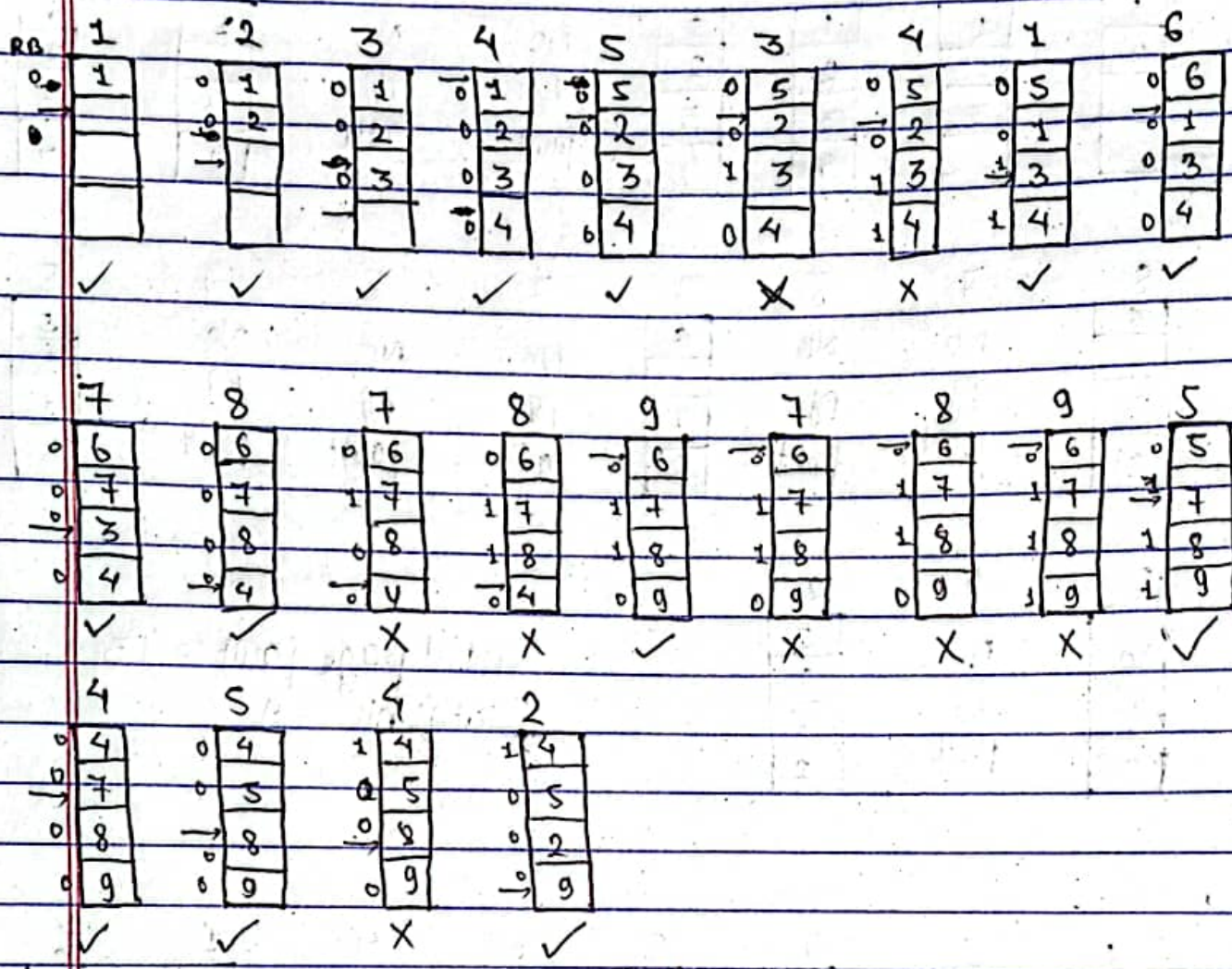
1	2	3	4	5	3	4	1	6
1	1	1	1	5	No	No	5	5
	2	2	2	2	Pg.	Pg.	1	1
		3	3	3	fault	fault	3	6
			4	4			4	4

7	8	7	8	9	7	8	9	5
5	8	No	No	8	No	No	No	8
1	1	Pg.	Pg.	9	Pg.	Pg.	Pg.	9
6	6	fault	fault	6	fault	fault	fault	5
7	7			7				7

4	5	4	2
8	No	No Pg.	2
9	Pg.	fault	9
5	fault		5
4			4

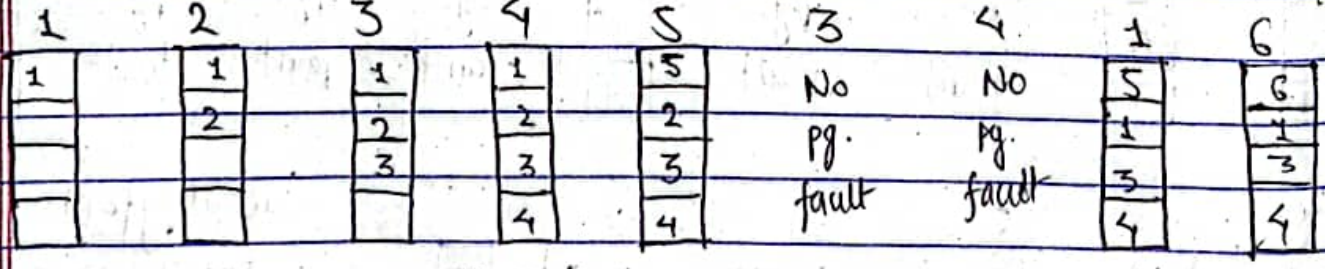
∴ Total pg fault = 13
Total hit = 9

(iv) Second Chance Page Replacement



Total page fault = 14
Total hit = 8

(v) NRU (Not Recently Used)



7	8	7	8	9	7	8	9	5
6	6	No pg. fault	No pg. fault	6	No pg. fault	No pg. fault	No pg. fault	5
1	1			9				9
7	7			7				7
4	8			8				8

4	5	4	2
5	No pg. fault	No pg. fault	5
9			9
4			4
8			2

Total page fault = 13
Total hit = 9

(vi) MFU (Most frequently Used)

1	2	3	4	5	3	4	1	6	7
1	1	1	1	5	No pg. fault	No pg. fault	5	5	5
	2	2	2	2			2	2	2
		3	3	3			1	1	7
			4	4			4	6	6

RS.	counter
1	X2
2	1
3	X2
4	XX3
5	XX3
6	1
7	XX3

8	7	8	9	7	8	9	5	4	5
5	No pg. fault	No pg. fault	5	[illegible]	5	No pg. fault	No pg. fault	5	No pg. fault
8			8		7			4	
7			9		9			9	
6			6		6			6	

9	X2	4	No pg. fault	2
				2
				4
				9
				6

∴ Total page fault = 14
Total hit = 8

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

→

Thrashing occur when a system spends more time swapping pages in and out of memory than executing actual processes. The primary cause of thrashing is insufficient memory, leading to frequent page faults.

This can be due to:

(i) Overcommitting memory
Too many processes running, each requiring more memory than is available.

(ii) Poor locality of refernce
Processes access data in a way that leads to frequent page replacements.

(iii) small page frames
A limited number of page frames allocated to processes cause excessive page faults.

Thrashing Detection :

- If the degree of multiprogramming increases and the CPU utilization drops (due to increased page faults and I/O operations), thrashing is likely occurring.
- The system can also use the working set model, which tracks the set of pages a process uses over a period of time. If the total demand for memory exceeds the available physical memory, thrashing is detected.

Eliminating Thrashing :

1. Reduce the degree of multiprogramming
   Suspend or swap out processes to free memory for remaining process.

2. Adjust page frame allocation
   Allocate more frames to processes with high page fault rates to fit their working sets in memory.

3. Use working set model
   Ensure each process's working set is in memory to optimize paging behavior.

4. Increase physical memory
   Add more RAM to reduce memory contention and page faults.

5. Tune system parameters.
   Adjust page size or swap space to better suit the workload and reduce paging overhead.

Q8)

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 MRU replace?

→ As MRU replace the most recently used data, it will replace page 3 as it is most recently used data i.e last reference is 285.

b) Which page will FIFO replace?

→ As FIFO replace the data that is loaded first, it will replace page 3 (110) as it is landed at first i.e. 110.

c) Which page will LRU replace?

→ As LRU replace the data that is least recently used, it will replace page 1 as it is least recently used i.e. last reference is 265.

d) Which page will second chance replace?

→ FIFO order : page 3 → page 0 → page 2 → page 1

checking R-bit :

page 3 : R=1 → skip

page 0 : R=1 → skip

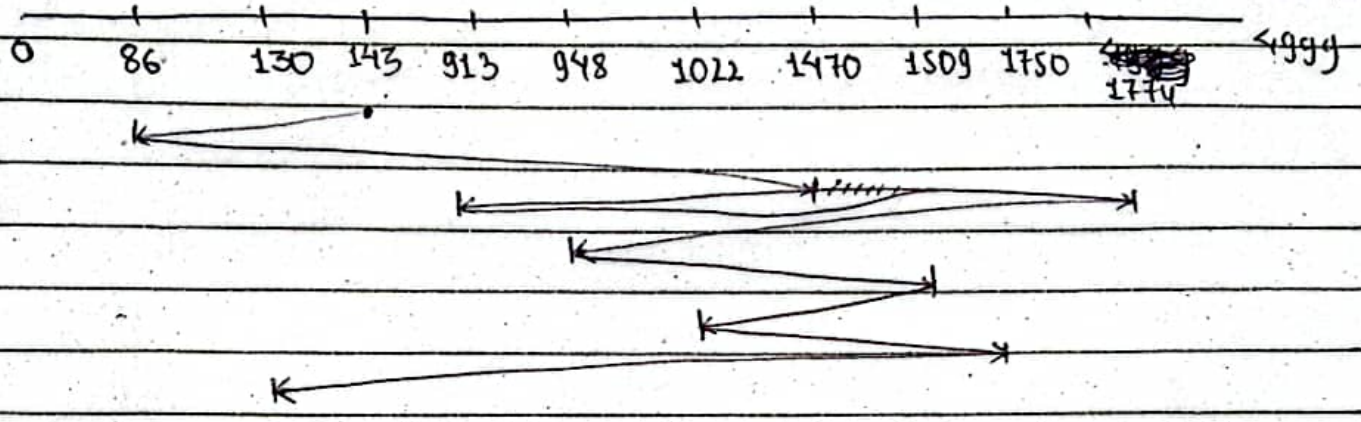
page 2 : R=0 → replaced

∴ page 2 will be replaced.

Q9) Soln cylinders : 0 to 4999

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

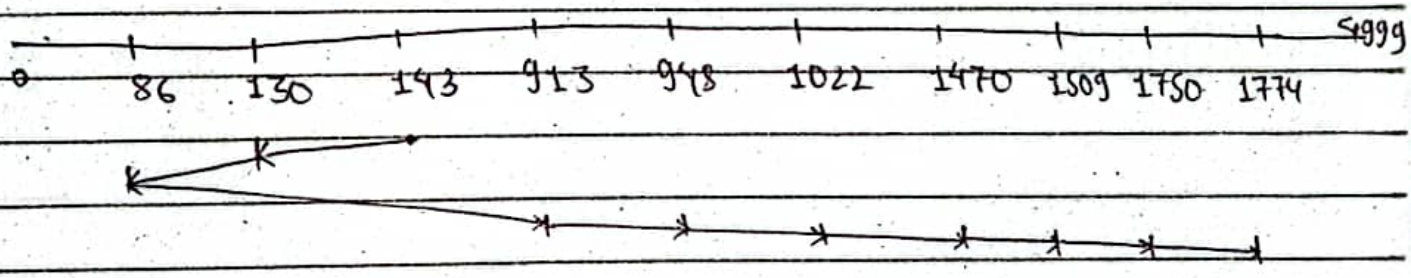
a) FCFS



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

Total seek distance = 7086 cylinders Ans

b) SSTF (shortest seek Time First)

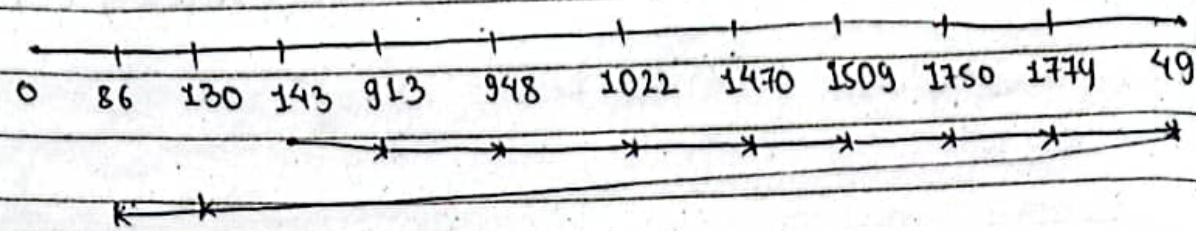


Seek Order : 143, 130, 86, 913, 948, 1022, 1470, 1509, 1750, 1774

Total seek distance = |86 - 143| + | 1774 - 86|

= 1745 cylinders Ans

c) SCAN

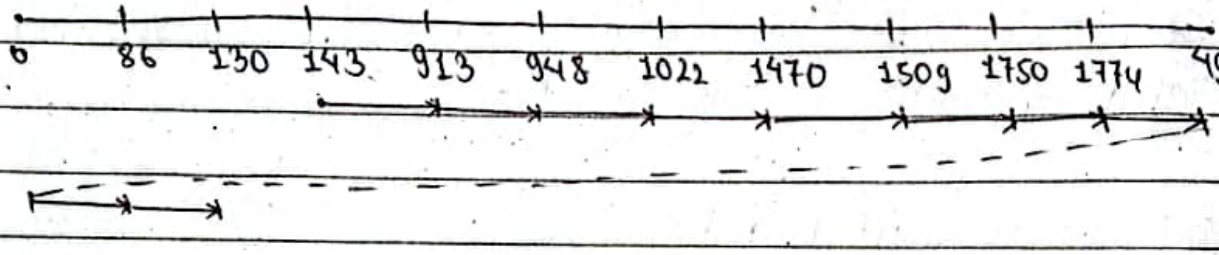


Seek order : 143, 913, 948, 1022, 1470, 1509, 1750, 1774, 4999, 130, 86

Seek distance = |4999 - 143| + |86 - 4999|

= 9769 cylinders Ans

d) C-SCAN

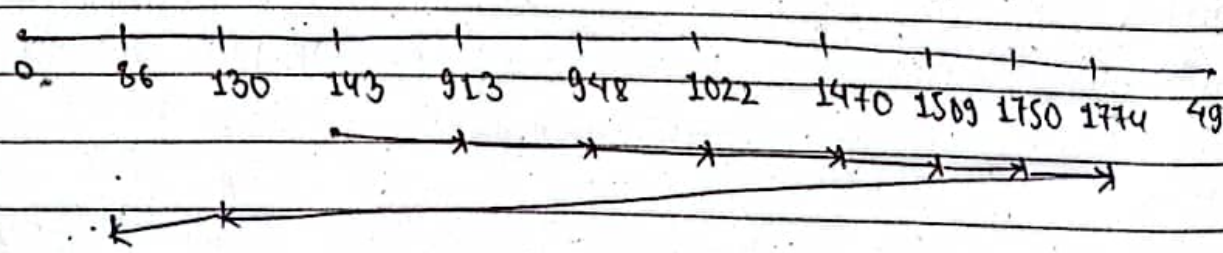


Seek Order : 143, 913, 948, 1022, 1470, 1509, 1750, 1774, 4999, 0, 86, 130

Seek distance = |4999 - 143| + |0 - 4999| + |130 - 0|

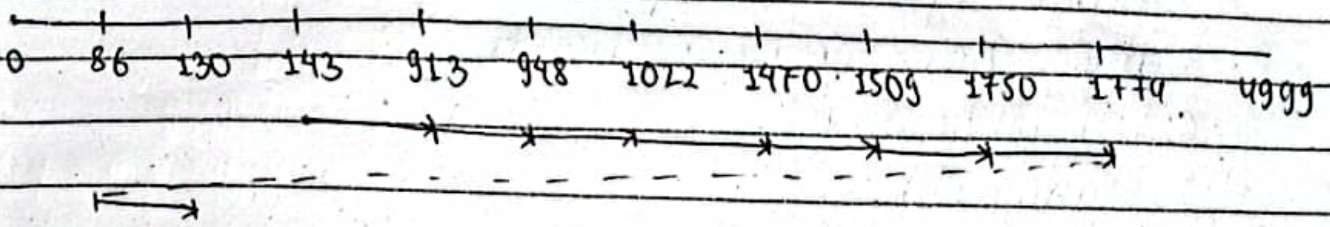
= 9985 cylinders Ans

e) LOOK



Seek Order: 143, 913, 948, 1022, 1470, 1509, 1750, 1774, 130, 86
Seek distance = |1774 - 143| + |86 - 1774|
= 3319 cylinders Ans

f) C-LOOK



Seek Order: 143, 913, 948, 1022, 1470, 1509, 1750, 1774, 86, 130
Seek distance = |1774 - 143| + |86 - 1774| + |130 - 86|
= 3363 cylinders Ans

Q10) What are the advantages and disadvantages of supporting memory mapped I/O to device control registers?

→

Advantages:

1. Unified Addressing
2. Ease of Programming
3. Efficient Content switching
4. Hardware simplification
5. Compatibility with caching
6. Flexibility in system design.

Date
Page

Disadvantages:

1. Conflict with Main Memory
2. Caching Issues
3. Performance Overhead
4. Complexity in Address Space Management
5. Device Access Timing Constraints
6. Probability Issues
7. Security Risks

Date
Page

Q11) Explain the difference between deadlock, livelock and starvation.

	Aspect	Deadlock	Livelock	Starvation
1.	Definition	A situation where two or more processes are indefinitely holding for resources held by other, preventing further progress.	A situation where two or more processes continuously changes their states to avoid conflict but fail to make progress.	A process is indefinitely delayed because higher-priority processes keep acquiring resources.
2.	Progress	NO progress as all involved processes are stuck waiting.	NO progress occurs despite processes being active & constantly changing state.	Some process progress, but the affected process never gets required resource.
3	Cause	Circular waiting for resources.	Excessive handling of conflict resolution or repeated retries to avoid deadlock.	Resource allocation policies (e.g. Priority scheduling) leads to infinite delays.
4.	State	Processes are blocked and not performing any action.	Processes are active but unable to proceed to next step.	A process is ready to execute but is perpetually ignored by the scheduler.
5.	Resolution	Requires external intervention (e.g. killing a process, preemption)	Adjust the algorithm to reduce excessive retries or randomize responses.	Adjust scheduling algorithms to ensure fairness (e.g. aging, Round-robin)

Q12) Differentiate between segmentation and paging.

	Aspect	Segmentation	Paging
1.	Definition	Divides memory into variable-based segments based on logical divisions.	Divides memory into fixed-sized pages.
2.	Size	Segments are variable in size.	Pages are fixed in size.
3.	Address	Consists of a segment no. and an offset.	Consists of a page number and a offset.
4.	Logical Division	Based on program's logical structure (e.g. code, data)	Divided purely for physical memory management, unrelated to program logic
5.	Fragmentation	Can cause external fragmentation	Can cause internal fragmentation.
6.	Translation	Requires segment table to map logical to physical addresses.	Requires a page table to map logical to physical addresses.
7	Ease of Use	More natural for programmers since it aligns with logical structures.	Easier for memory management in the operating system.
8.	Swapping	Entire segments swapped.	Individual pages swapped.
9.	Overhead	Higher	Lower

Q13) Explain Producer consumer Problem using semaphore.

→

The Producer consumer Problem is a classic synchronization problem that involves two types of processes: Producer and consumer, which share a fixed-size buffer.

The producer generates data and adds it to the buffer, while the consumer removes data from the buffer. The challenge is to ensure synchronization between these processes to prevent overwriting or reading empty data.

Semaphores are synchronization tool used to coordinate access to shared buffer.
We use three semaphores:

- Full } Counting semaphore
- Empty
- Muten } Binary semaphore

Date
Page

We initialize them as,

Mutex = 1 : Allows mutual exclusion for critical region
Full = 0 : Initially, buffer has no filled slots
Empty = N : Initially, buffer has 'N' empty slots
where, N ⇒ size of buffer

The algorithm for processes will be as,

Producer process:

```
do {
down(empty) ;          // Decrease number of empty slots
down(mutex) ;          // Enter critical region
 - - -
 // Add item to buffer
 - - -
 up(mutex) ;           // Exit critical region
 up(full) ;            // Increase number of filled slots
} while (true);
```

Consumer process :

```
do {
down(full);            // Decrease number of filled slots
down(mutex);           // Enter critical region
 - - - -
 // Remove item from buffer
 - - - -
 up(mutex);            // Exit critical region
```

```
        up(empty);          //Increase number of empty slots
    } while (true);
```

This semaphore based solution guarantees proper synchronization, prevents race conditions & ensures both producer and consumer process work correctly.

Q14)

Process	Max	Allocation	Available
	A, B, C, D	A, B, C, D	A, B, C, D
P_0	6 0 1 2	4 0 0 1	3 2 1 1
P_1	2 7 5 0	1 1 0 0	
P_2	2 3 5 6	1 2 5 4	
P_3	1 6 5 3	0 6 3 3	
P_4	1 6 5 6	0 2 1 2	

Soln

According to Banker's Algorithm,

For safe state,
Need ≤ Available

Need = Max - Allocation

Need Matrix:

	A	B	C	D
P_0	2	0	1	1
P_1	1	6	5	0
P_2	1	1	0	2
P_3	1	0	2	0
P_4	1	4	4	4

Here, Need of P_0 is ≤ available (3, 2, 1, 1). So, we run P_0.

Run P_0:

After completion, it releases its allocated resources.

Date
Page

∴ New Available = Allocation of P_0 + Available
= (4,0,0,1) + (3,2,1,1)
= (7,2,1,2)

The need of P_2 ≤ available. So, we run P_2.

Run P_2: New available = (1,2,5,4) + (7,2,1,2)
= (8,4,6,6)

Need of P_3 & P_4 ≤ available. So, we can run either P_3 or P_4.

Run P_4: New available = (8,6,7,8)

Need of P_1 & P_3 ≤ available. So, we can run either P_1 or P_3.

~~∴ New A~~

Run P_3: New available after completion = (8,12,10,11)

Run P_1: New available after completion = (9,13,10,11)

Yes, the system is in safe state, because all ~~system are~~ process are in safe state.

One of the safe sequence is P_0, P_2, P_4, P_3, P_1. Ans