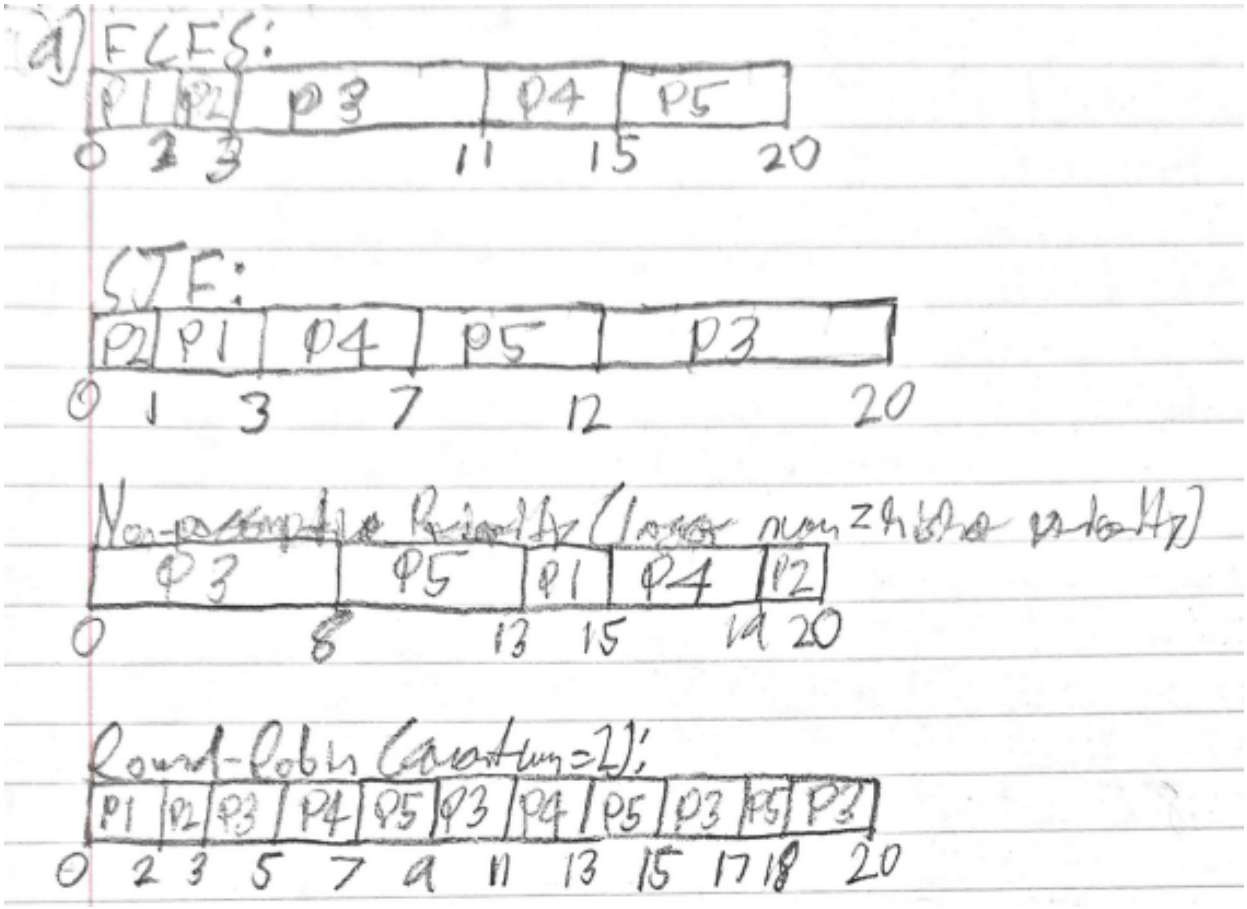
5.4 Consider the following set of processes, with the length of the CPU burst time given in milliseconds:

Process	Burst Time	Priority
P_1	2	2
P_2	1	1
P_3	8	4
P_4	4	2
P_5	5	3

The processes are assumed to have arrived in the order P_1, P_2, P_3, P_4, P_5, all at time 0.

a. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, nonpreemptive priority (a larger priority number implies a higher priority), and RR (quantum = 2).



b. What is the turnaround time of each process for each of the scheduling algorithms in part a?

Turnaround Time (FCFS):	Turnaround Time (SJF):	Turnaround Time (Priority):	Turnaround Time (Round-Robin):
P1 = 2 - 0 = 2ms P2 = 3 - 0 = 3ms P3 = 11 - 0 = 11ms P4 = 15 - 0 = 15ms P5 = 20 - 0 = 20ms	P1 = 3 - 0 = 3ms P2 = 1 - 0 = 1ms P3 = 20 - 0 = 20ms P4 = 7 - 0 = 7ms P5 = 12 - 0 = 12ms	P1 = 15 - 0 = 15ms P2 = 20 - 0 = 20ms P3 = 8 - 0 = 8ms P4 = 19 - 0 = 19ms P5 = 13 - 0 = 13ms	P1 = 2 - 0 = 2ms P2 = 3 - 0 = 3ms P3 = 20 - 0 = 20ms P4 = 13 - 0 = 13ms P5 = 18 - 0 = 18ms

c. What is the waiting time of each process for each of these scheduling algorithms?

Waiting Time (FCFS):	Waiting Time (SJF):	Waiting Time (Priority):	Waiting Time (Round-Robin):
P1 = 2 - 2 = 0ms P2 = 3 - 1 = 2ms P3 = 11 - 8 = 3ms P4 = 15 - 4 = 11ms P5 = 20 - 5 = 15ms	P1 = 3 - 2 = 1ms P2 = 1 - 1 = 0ms P3 = 20 - 8 = 12ms P4 = 7 - 4 = 3ms P5 = 12 - 5 = 7ms	P1 = 15 - 2 = 13ms P2 = 20 - 1 = 19ms P3 = 8 - 8 = 0ms P4 = 19 - 4 = 15ms P5 = 13 - 5 = 8ms	P1 = 2 - 2 = 0ms P2 = 3 - 1 = 2ms P3 = 20 - 8 = 12ms P4 = 13 - 4 = 9ms P5 = 18 - 5 = 13ms

d. Which of the algorithms results in the minimum average waiting time (over all processes)?

Average waiting time (FCFS):
(0 + 2 + 3 + 11 + 15) / 5 = 31 / 5 = 6.2

Average waiting time (SJF):
(0 + 1 + 3 + 7 + 12) / 5 = 23 / 5 = 4.6

Average waiting time (Priority):
(0 + 8 + 13 + 15 + 19) / 5 = 55 / 5 = 11

Average waiting time (Round-Robin):
(0 + 2 +12 + 9 + 13) / 5 = 36 / 5 = 7.2

Shortest Jobs First gives us the minimum average waiting time!

5.7 Many CPU-scheduling algorithms are parameterized. For example, the RR algorithm requires a parameter to indicate the time slice. Multilevel feedback queues require parameters to define the number of queues, the scheduling algorithms for each queue, the criteria used to move processes between queues, and so on.

These algorithms are thus really sets of algorithms (for example, the set of RR algorithms for all time slices, and so on). One set of algorithms may include another (for example, the FCFS algorithm is the RR algorithm with an infinite time quantum). What (if any) relation holds between the following pairs of algorithm sets?

a. Priority and SJF

The shortest job has the highest priority.

b. Multilevel feedback queues and FCFS

The lowest level of multilevel feedback queues is FCFS.

c. Priority and FCFS

FCFS gives the highest level of priority to the job that has arrived first.

d. RR and SJF

None.

9.13 Given six memory partitions of 100 MB, 170 MB, 40 MB, 205 MB, 300 MB, and 185 MB (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of size 200 MB, 15 MB, 185 MB, 75 MB, 175 MB, and 80 MB (in order)? Indicate which—if any—requests cannot be satisfied. Comment on how efficiently each of the algorithms manages memory.

Partitions: 100, 170, 40, 205, 300, 185
Processes: 200, 15, 185, 75, 175, 80

First-Fit:

Step:	100 mb	170 mb	40 mb	205 mb	300 mb	185 mb
1.	_____	_____	_____	200	_____	_____
2.	15	_____	_____	_____	_____	_____
3.	_____	_____	_____	_____	185	_____
4.	_____	75	_____	_____	_____	_____

5.						175
6.						

80 mb request cannot be satisfied.

Best-Fit:

Step:	100 mb	170 mb	40 mb	205 mb	300 mb	185 mb
1.				200		
2.			15			
3.						185
4.	75					
5.					175	
6.		80				

Worst-Fit:

Step:	100 mb	170 mb	40 mb	205 mb	300 mb	185 mb
1.					200	
2.				15		
3.						185
4.		75				
5.						
6.	80					

175 mb request cannot be satisfied.

None of the algorithms is best and it entirely depends upon the situation.

First-Fit is really efficient to use and the most widely used algorithm. It is efficient in terms of speed and time-wise because it simply chooses the first free hole that is large enough. However, it suffers 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.

Best Fit takes time to search the best possible fit memory partition and the remaining space which is left after becomes useless. It is slower compared to the First-Fit Algorithm and it also suffers from external fragmentation.

Worst-Fit will allocate the largest hole and as a result of searching for this hole, it will take some time. Again, we must search the entire list, unless it is sorted by size. This strategy produces the largest leftover hole, which may be more useful than the smaller leftover hole from a best-fit approach

Best-Fit effectively minimizes the wastage space but it takes time to find the best fit memory partition to fill.
First-Fit Algorithm is the best overall in terms of time and memory.

9.25 Consider a paging system with the page table stored in memory.

a. If a memory reference takes 50 nanoseconds, how long does a paged memory reference take?

Given that memory reference takes 50 nanoseconds, and that the paging system with a page table is stored in memory. In a paging system, accessing a memory location takes two memory references i.e, 1.accessing page table and 2.accessing actual memory locations (page table -> memory location) and the page table is stored in memory.
Therefore paged memory reference takes = 2* one memory reference = 2 * 50 ns = 100 ns.

b. If we add TLBs, and if 75 percent of all page-table references are found in the TLBs, what is the effective memory reference time? (Assume that finding a page-table entry in the TLBs takes 2 nanoseconds, if the entry is present.)

After adding TLBs, Effective Memory Reference time
= (Time required to find page table entry in TLBs) + (Average time required by memory reference)
= 2 ns + (0.75*50 + 0.25*100)
= 64.5 ns

10.24 Apply the (1) FIFO, (2) LRU, and (3) optimal (OPT) replacement algorithms for the following page-reference strings:

- 2,6,9,2,4,2,1,7,3,0,5,2,1,2,9,5,7,3,8,5
- 0,6,3,0,2,6,3,5,2,4,1,3,0,6,1,4,2,3,5,7
- 3,1,4,2,5,4,1,3,5,2,0,1,1,0,2,3,4,5,0,1
- 4,2,1,7,9,8,3,5,2,6,8,1,0,7,2,4,1,3,5,8
- 0,1,2,3,4,4,3,2,1,0,0,1,2,3,4,4,3,2,1,0

Indicate the number of page faults for each algorithm assuming demand paging with three frames.

2,6,9,2,4,2,1,7,3,0,5,2,1,2,9,5,7,3,8,5

FIFO:

1	**2**	**3**		**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**		**13**	**14**	**15**	**16**	**17**	**18**
2	2	2		4	4	4	7	7	7	5	5	5		9	9	9	3	3	3
	6	6		6	2	2	2	3	3	3	2	2		2	5	5	5	8	8
		9		9	9	1	1	1	0	0	0	1		1	1	7	7	7	5

18 Page Faults!

LRU:

1	**2**	**3**		**4**		**5**	**6**	**7**	**8**	**9**	**10**	**11**		**12**	**13**	**14**	**15**	**16**	**17**
2	2	2		2		2	2	3	3	3	2	2		2	2	7	7	7	5
	6	6		4		4	7	7	7	5	5	5		9	9	9	3	3	3
		9		9		1	1	1	0	0	0	1		1	5	5	5	8	8

17 Page Faults!

OPT:

1	**2**	**3**		**4**	**5**	**6**	**7**	**8**	**9**		**10**	**11**	**12**	**13**
2	2	2		2	2	2	2	2	2		9	9	3	3
	6	6		4	1	1	1	1	1		1	7	7	8
		9		9	9	7	3	0	5		5	5	5	5

13 Page Faults!

0,6,3,0,2,6,3,5,2,4,1,3,0,6,1,4,2,3,5,7

FIFO:

1	**2**	**3**		**4**		**5**		**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0	0	0		2		2		2	1	1	1	6	6	6	2	2	2	7
	6	6		6		5		5	5	3	3	3	1	1	1	3	3	3
		3		3		3		4	4	4	0	0	0	4	4	4	5	5

16 Page Faults!

LRU:

1	**2**	**3**		**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**
0	0	0		0	0	3	3	3	4	4	4	0	0	0	4	4	4	5	5
	6	6		2	2	2	5	5	5	1	1	1	6	6	6	2	2	2	7
		3		3	6	6	6	2	2	2	3	3	3	1	1	1	3	3	3

19 Page Faults!

OPT:

1	**2**	**3**		**4**		**5**		**6**	**7**		**8**	**9**		**10**	**11**	**12**	**13**
0	0	0		2		2		2	1		1	1		2	2	2	7
	6	6		6		5		4	4		4	4		4	3	3	3
		3		3		3		3	3		0	6		6	6	5	5

13 Page Faults!

3,1,4,2,5,4,1,3,5,2,0,1,1,0,2,3,4,5,0,1

FIFO:

1	**2**	**3**	**4**	**5**		**6**	**7**		**8**	**9**	**10**		**11**	**12**	**13**	**14**	**15**
3	3	3	2	2		2	3		3	3	1		1	1	5	5	5
	1	1	1	5		5	5		2	2	2		3	3	3	0	0
		4	4	4		1	1		1	0	0		0	4	4	4	1

15 Page Faults!

LRU:

1	**2**	**3**	**4**	**5**		**6**	**7**	**8**	**9**	**10**	**11**		**12**	**13**	**14**	**15**	**16**
3	3	3	2	2		1	1	1	2	2	2		2	2	5	5	5
	1	1	1	5		5	3	3	3	0	0		0	4	4	4	1
		4	4	4		4	4	5	5	5	1		3	3	3	0	0

16 Page Faults!

OPT:

1	**2**	**3**	**4**	**5**		**6**		**7**	**8**		**9**	**10**	**11**	
3	3	3	2	5		5		2	2		3	4	5	
	1	1	1	1		1		1	1		1	1	1	
		4	4	4		3		3	0		0	0	0	

11 Page Faults!

4,2,1,7,9,8,3,5,2,6,8,1,0,7,2,4,1,3,5,8

FIFO:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
4	4	4	7	7	7	3	3	3	6	6	6	0	0	0	4	4	4	5	5
	2	2	2	9	9	9	5	5	5	8	8	8	7	7	7	1	1	1	8
		1	1	1	8	8	8	2	2	2	1	1	1	2	2	2	3	3	3

20 Page Faults!

LRU:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
4	4	4	7	7	7	3	3	3	6	6	6	0	0	0	4	4	4	5	5
	2	2	2	9	9	9	5	5	5	8	8	8	7	7	7	1	1	1	8
		1	1	1	8	8	8	2	2	2	1	1	1	2	2	2	3	3	3

20 Page Faults!

OPT:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**		**9**	**10**		**11**	**12**		**13**		**14**	**15**	**16**
4	4	4	7	9	8	3	5		6	8		0	7		4		3	3	3
	2	2	2	2	2	2	2		2	2		2	2		2		2	5	5
		1	1	1	1	1	1		1	1		1	1		1		1	1	8

16 Page Faults!

0,1,2,3,4,4,3,2,1,0,0,1,2,3,4,4,3,2,1,0

FIFO:

1	**2**	**3**	**4**	**5**		**6**	**7**		**8**	**9**	**10**		**11**	**12**
0	0	0	3	3		3	0		0	0	4		4	4
	1	1	1	4		4	4		2	2	2		1	1
		2	2	2		1	1		1	3	3		3	0

12 Page Faults!

LRU:

1	**2**	**3**	**4**	**5**		**6**	**7**		**8**	**9**		**10**	**11**
0	0	0	3	3		3	0		3	3		3	0
	1	1	1	4		1	1		1	4		1	1
		2	2	2		2	2		2	2		2	2

11 Page Faults!

OPT:

1	**2**	**3**	**4**	**5**		**6**	**7**		**8**	**9**		**10**	**11**
0	0	0	3	3		3	0		3	3		1	1
	1	1	1	4		1	1		1	4		3	0
		2	2	2		2	2		2	2		4	4

11 Page Faults!

10.18 The following is a page table for a system with 12-bit virtual and physical addresses and 256-byte pages. Free page frames are to be allocated in the order `9, F, D`. A dash for a page frame indicates that the page is not in memory.

Page	Page Frame
0	0 x 4
1	0 x B
2	0 x A
3	–
4	–
5	0 x 2
6	–
7	0 x 0
8	0 x C
9	0 x 1

Convert the following virtual addresses to their equivalent physical addresses in hexadecimal. All numbers are given in hexadecimal. In the case of a page fault, you must use one of the free frames to update the page table and resolve the logical address to its corresponding physical address.

- `0x2A1`
- `0x4E6`
- `0x94A`
- `0x316`

Given:
Virtual address = 12 bits
Physical address = 12 bits
Page size = 256 bytes
Number of bits used to represent the offset or page size: log2(256) = 8 bits

Number of bits used to represent virtual pages in the virtual address: Virtual address bits - Page size bits => 12 - 8 = 4 bits
Number of bits used to represent frame number in the virtual address: Physical address bits - Page size bits => 12 - 8 = 4 bits
The virtual address is represented as:

Page number bits (4 bits)	Page size bits (8 bits)

Virtual address = 12
The physical address is represented as:

Frame number bits (4 bits)	Page size bits (8 bits)

Physical address = 12
The free frames are allocated in the order: 9, F, D.

0x2A1 (*In hexadecimal format):
From hex to binary:
2 = 0010, A = 1010, 1 = 0001, so we get: 0010 1010 0001
According to the representation of the virtual address, the first 4 bits from the left represent the page number and the next 8 bits represent the page size or offset.
So,
Page number = 0010 or 2
Page size or offset = 10100001 or A1 in hexadecimal
In the page table, the page number 2 corresponds to frame number 0xA
The offset remains the same in both the virtual and physical address.
So,
The physical address = 0xAA1.

0x4E6 (*In hexadecimal format):
From hex to binary:
4 = 0100, E = 1110, 6 = 0110, so we get: 0100 1110 0110
According to the representation of the virtual address, the first 4 bits from the left represent the page number and the next 8 bits represent the page size or offset.
So,
Page number = 0100 or 4
Page size or offset = 11100110 or E6 in hexadecimal
In the page table, the page number 4 does not have any frame mapping
So, this results in a page fault.
The first allocated free frame is 9.
So, we update the page table for the frame number of page 4 as 9.
The page number 4 now corresponds to frame number 9.
The offset remains the same in both the virtual and physical address.
So,
The physical address = 0x9E6.

0x94A (*In hexadecimal format):
From hex to binary:
9 = 1001, 4 = 0100, A = 1010, so we get: 1001 0100 1010

According to the representation of the virtual address, the first 4 bits from the left represent the page number and the next 8 bits represent the page size or offset.
So,
Page number = 1001 or 9
Page size or offset = 01001010 or 4A in hexadecimal
In the page table, the page number 9 corresponds to frame number 0x1
The offset remains the same in both the virtual and physical address.
So,
The physical address = 0x14A.

0x316 (*In hexadecimal format):
From hex to binary:
3 = 0011, 1 = 0001, 6 = 0110, so we get: 0011 0001 0110
According to the representation of the virtual address, the first 4 bits from the left represent the page number and the next 8 bits represent the page size or offset.
So,
Page number = 0011 or 3
Page size or offset = 00010110 or 16 in hexadecimal
In the page table, the page number 3 does not have any frame mapping
So, this results in a page fault.
The second allocated free frame is F.
So, we update the page table for the frame number of page 3 as F.
The page number 3 now corresponds to frame number F.
The offset remains the same in both the virtual and physical address.
So,
The physical address = 0xF16.

11.13 Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4,999. The drive is currently serving a request at cylinder 2,150, and the previous request was at cylinder 1,805. The queue of pending requests, in FIFO order, is:

2,069; 1,212; 2,296; 2,800; 544; 1,618; 356; 1,523; 4,965; 3,681

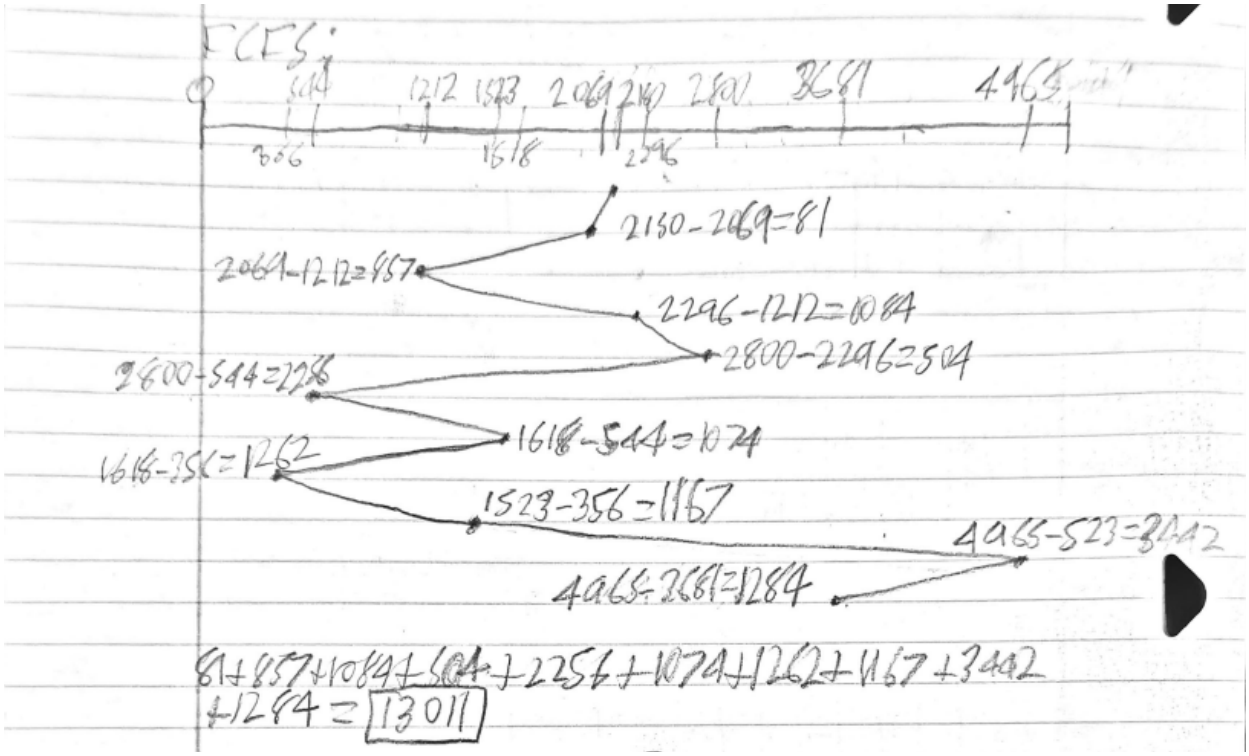
Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms?

a. FCFS

b. SCAN

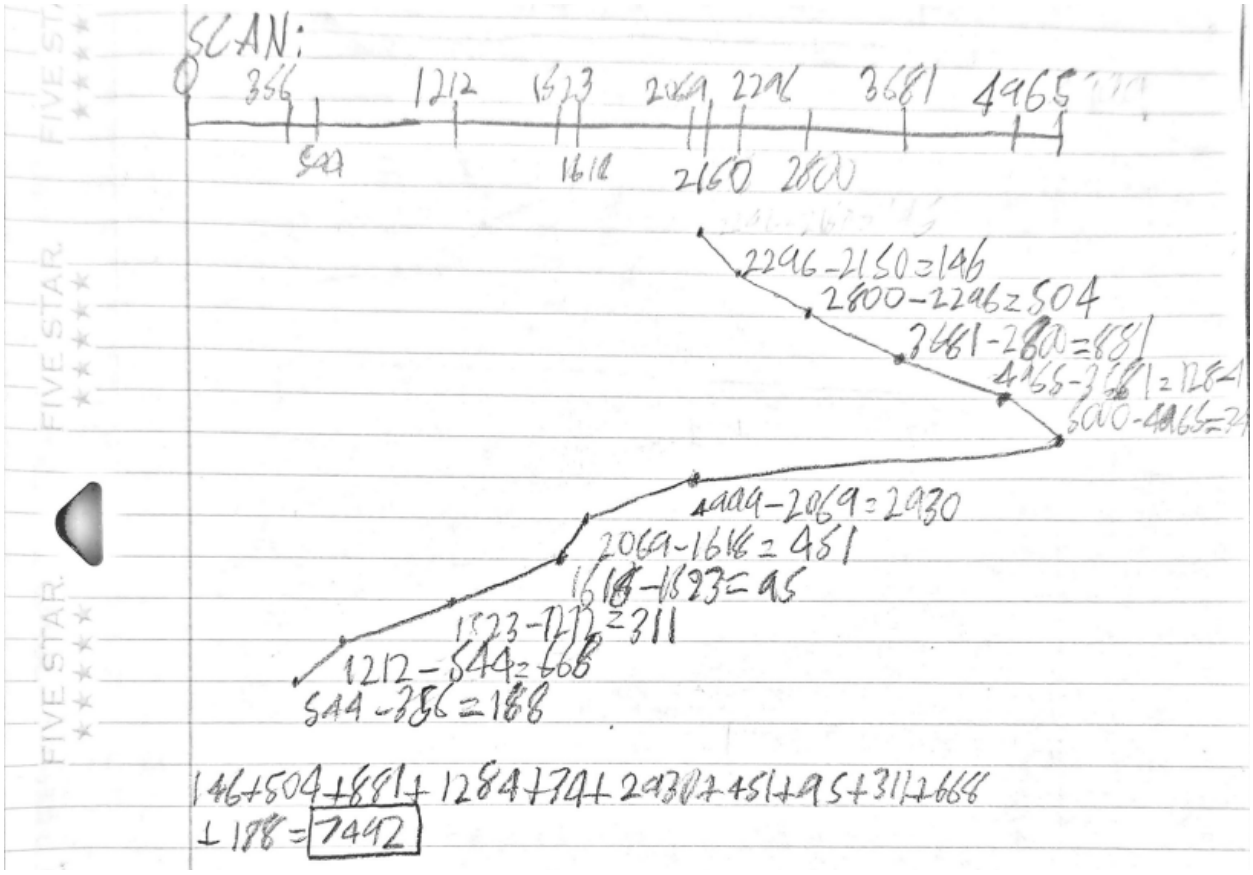
c. C-SCAN

a. FCFS



b. SCAN

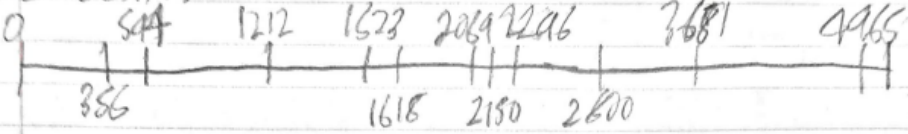
Since the request before 2150 is 1805 (*which comes off as initial movement towards the right), we know that we are going to start off by going to the right.



c. C-SCAN

Since the request before 2150 is 1805 (*which comes off as initial movement towards the right), we know that we are going to start off by going to the right.

C-SCAN:



2296-2150=146

2600-2296=304

3681-2600=681

4965-3681=1284

4999-4965=34

4999-0=4999

356-0=356

544-356=188

1212-544=668

1523-1212=311

1618-1523=95

2069-1618=451

146+304+681+1284+34+4999+356+188+668+311
+95+451=**9917**