**CLEVELAND STATE UNIVERSITY**

**CIS 545 – Architecture and Operating System**

**Spring 2020: CIS 345/545 Exam**

**Submitted To**

**PROF. JANCHE SANG**

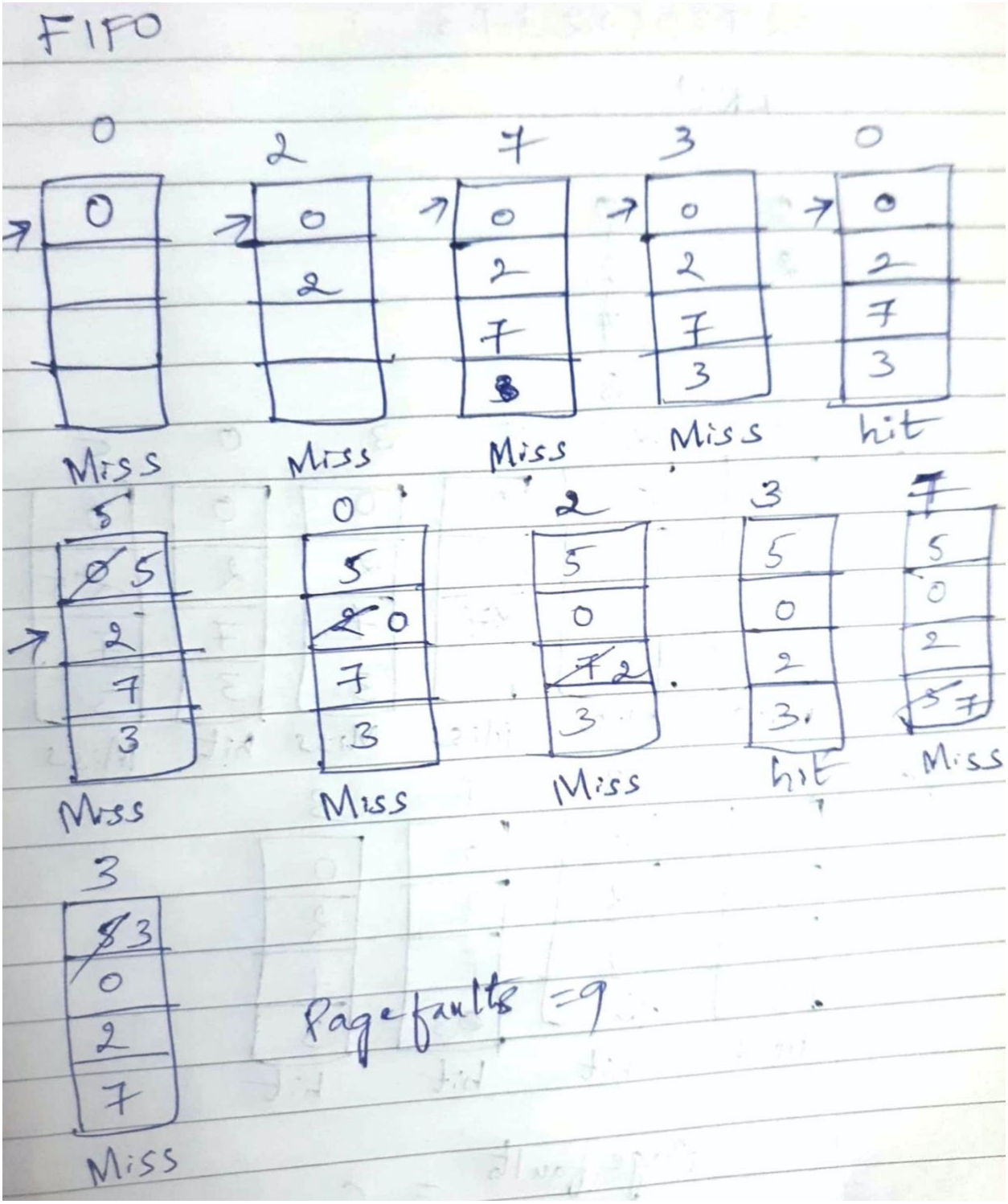
**Submitted By**

**PRUDHVI REDDY ARAGA (CSU ID – 2788245; Grail Login ID - praraga)**

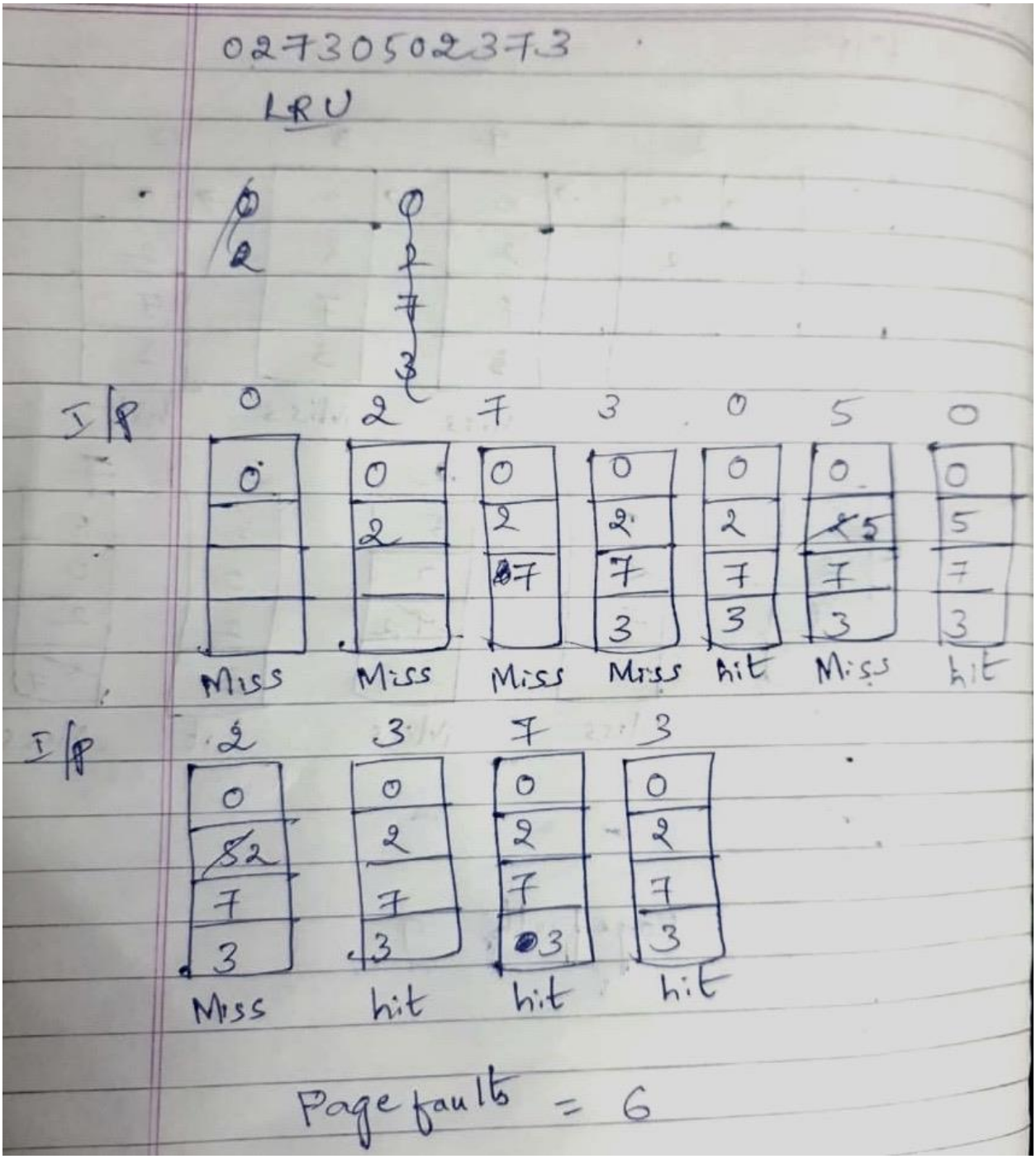
1. **(a).** If I'm the instructor for this course, I would follow the below practices to discourage cheating among students:
    - Establish expectations on Day 1 of the class.
    - Working collaboratively with the students directly to solve complex problems.
    - Set expectations on how grading is distributed among assignments and exam to students.
    - Conduct different quiz questions to the students so that they have an understanding of where they stand in the subject and where they would need improvement.
    - Provide variety of questions to students in exam so that they would not have chance to cheat in the exam.
    - I would provide practice exam questions and assignments.
    - Asking more questions based on applications rather than theory, in the final exam.
    - Would make the exam open book so that students do not rely on cheating to figure out answers.
1. **(b).** If a student is caught cheating, I would talk to the student in person to figure out why the student has resorted to copying. I'll try to explore the root cause of the problem where the student is lacking in his course and explain the ethics required for a student. Provide the student with all the information required to gain knowledge and get through the course successfully. Would monitor the student to see if there is progress in his learning methods. I would give an opportunity for the student to change his methods and start working towards enhancing the technical skill set. If the student does it for the second time, I would report it to the chair of the department and issue an official warning to the student. Even

after the official warning, if the student does not change his ways, I would then fail the student in the course and would recommend the school to take disciplinary action.

2. **(a) FIFO:** Reference String – 02730502373; **Page Faults = 9**



2. **(b). LRU:** Reference String – 02730502373; **Page Faults = 6**



3. **(a).** The outgoing physical address is: **0 0 0 0 0 0 0 1 0 0 0 0 0 1 0**.

**Explanation**

Virtual page number is 0 1 1 0 which is 6.

Page index present index in the index is 0 0 0.

Rest of the values are copied from input to output.

**(b).** The outgoing physical address is: **1 1 1 0 0 0 1 0 0 1 0 1 0 0 0**.

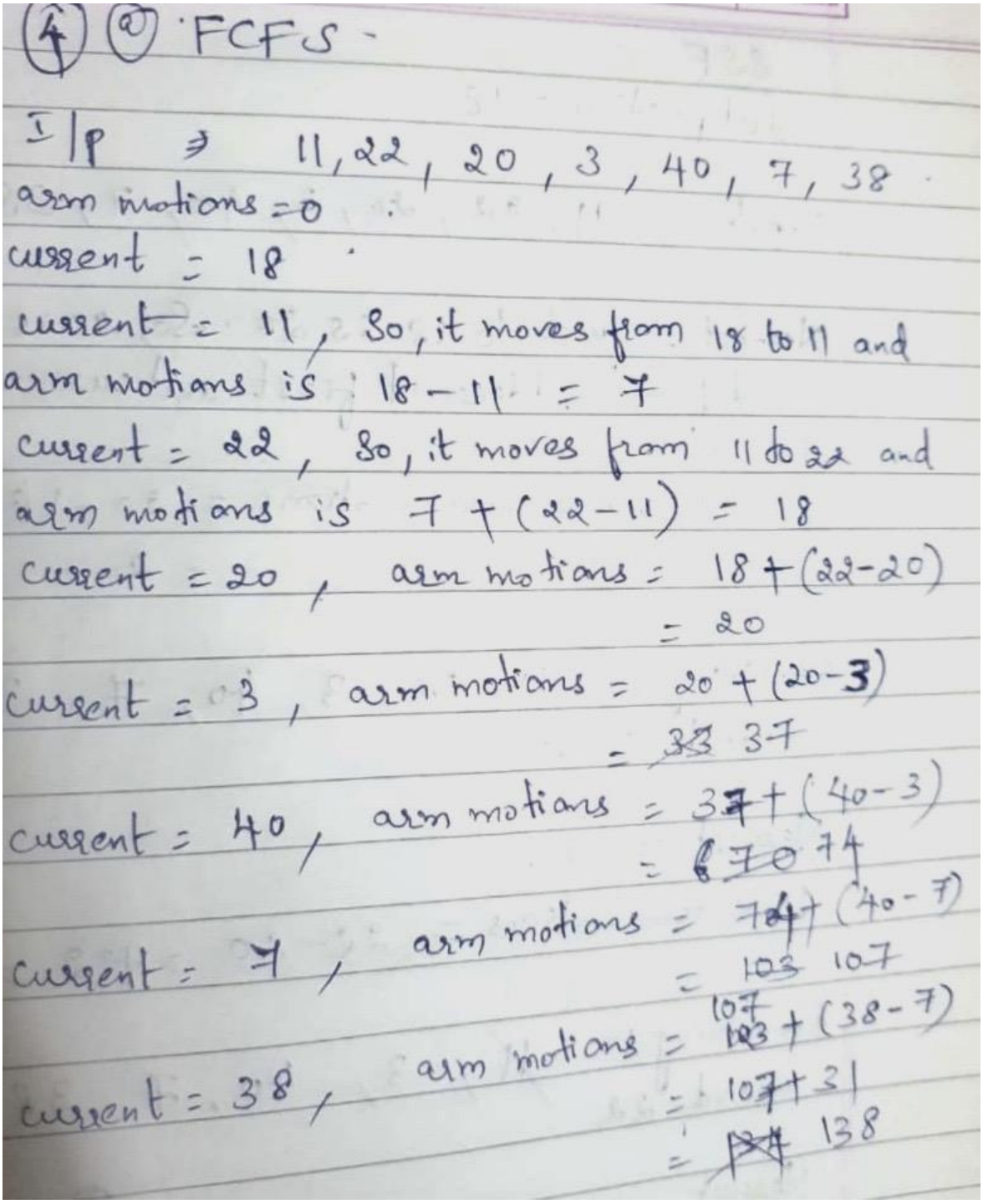
**Explanation**

Virtual Page number is 1 0 1 1 which is 1.

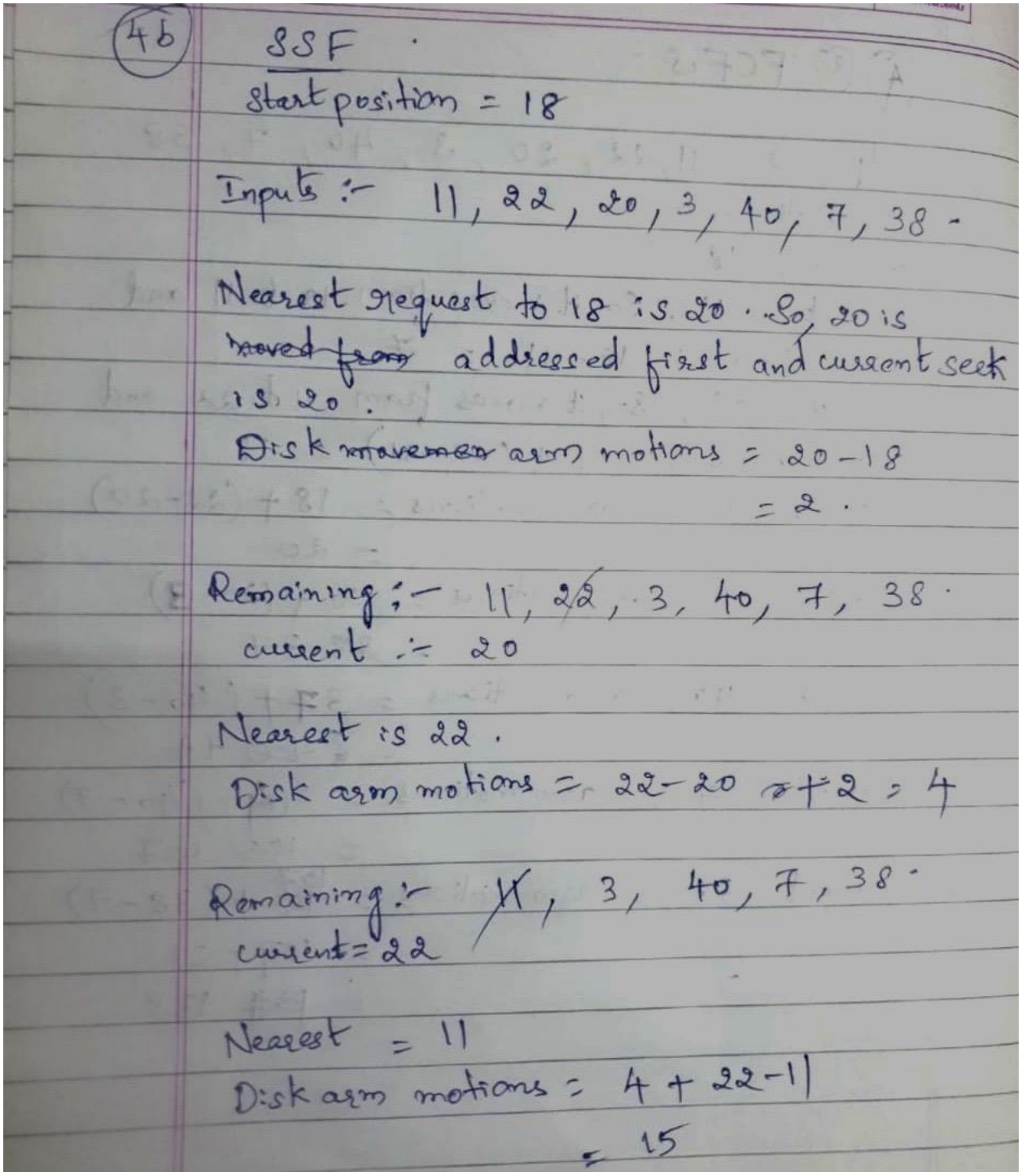
Page Index present index in the index is 1 1 1.

Rest of the values are copied from input to output.

**4. (a). FCFS: Required Arm Motions = 138**



**4.(b). Shortest Seek First (SSF): Required Arm Motions = 60**



Remaining :- 3, 40, 7, 38.
Current = 11
Nearest = 7.
Disk arm motions = 15 + (11 - 7)
= 19

Remaining :- 3, 40, 38.
current = 7
Nearest = 3
Disk arm motions = 19 + (7 - 3)
= 23

Remaining = 40, 38.
current = 3
Nearest = 38.
Disk arm motions = 23 + (38 - 3)
= 23 + 35
= 58

Remaining = 40
Disk arm motions = 58 + (40 - 38)
= 60

**4.(c). Elevator algorithm (initially moving upwards): Required Arm Motions = 59**

Elevator algorithm :-

current :- 18.

Inputs :- 11, 22, 20, 3, 40, 7, 38.

Sorted order of requests from 18 :-

20, 22, 38, 40

Remaining requests :- 11, 3, 7.

Sorted order of requests :- 11, 7, 3

No. of arm rotations in upward movement,

~~= (18 - 11) + (~~

= (20 - 18) + (22 - 20) + (38 - 22) +
(40 - 38)

= 22

No. of rotations in downward movement

= (40 - 11) + (11 - 7) + (7 - 3)

= 37

Total no. of movements = 22 + 37

= 59

**5.(a). Number of blocks in use on a disk:**

```
void tfs_debug()
{
	int i,j,k;
	int b_in_use = 0;
	int inode_in_use = 0;
	union tfs_block block,inblock,onblock;
	disk_read(0,block.data);
	for(i=0; i<NUM_BLOCKS; i++)
	  if(block.super.block_in_use[i/BITS_PER_ULONG] & (1 <<(i%BITS_PER_ULONG)))
		b_in_use++ ;
	printf("	%d blocks in use \n", b_in_use);
}
```

**6.(a).** Has Max Required = Max-Has

P0 2 8 6

P1 1 5 4

P2 2 6 4

P3 4 7 3

Free: 3

**No request cannot be granted to P2.**

If requested is granted to P2, free resources will be only 2 and P0 requires 6 units, P2 requires 3 units and P3 requires 4 units and P4 requires 3 units. So, requests cannot be granted to any one of the processes to complete execution which results in deadlock.

**6.(b). Yes. Request can be granted to P3.**

If request is granted to P3, free resources will be 2 and P3 requires 2 resources to complete execution which is possible to allocate which results in safe state. One P3 is completed, Free resources will be 10. So, P0, P1, P3 requires 6, 3, 4 units respectively which can also be addressed as there are 10 free resources.