1) a) i) F
There is only a ready queue and a waiting queue.

ii) F
All I/O instructions execute in kernel mode as it is a privileged instruction.

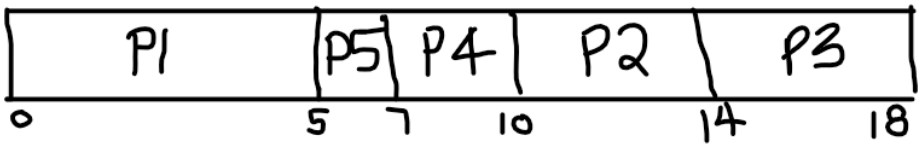
iii) F
The next process in the ready queue will begin execution.

iv) F
DMA still uses interrupt, just that they copy a large block of data per interrupt, while interrupt driven I/O has much more interrupts where it copies byte per interrupt

v) F
It is still possible within the processes of each core.

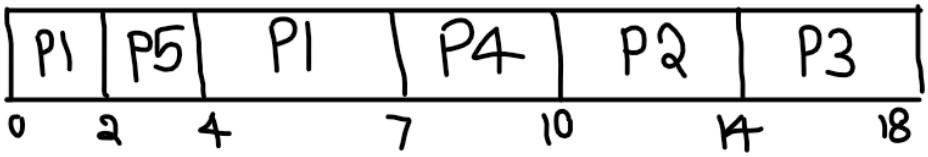
vi) T
When an interrupt occurs.

b) When non – pre-emptive shortest job first is used the following execution sequence is obtained: P1 -> P5 -> P4 -> P2 -> P3



The average waiting time in this case is = (0 + 8 + 10 + 2 + 3) / 5 = **4.6**

When Shortest Remaining Time First is used, the following Gantt Chart is obtained:



The average waiting time in this case is = (2 + 8 + 10 + 2 + 0)/5 = **4.4**

c) i) Different processes have different requirements:
- Foreground processes like those handling I/O operations need to be interactive and hence Round Robin Scheduling is preferred.
- Background processes do not have to be interactive and hence the scheduling overhead can be reduced. Therefore, FCFS is used.

ii) The two different scheduling algorithms that can be used for inter – queue scheduling are:
- Fixed Priority Scheduling: The queues are served in a priority order, for example, foreground before background. Can lead to starvation.
- Time – Slice based scheduling: Each queue gets a fixed quantum on the CPU.

d) i) Since no interrupt occurs, the scheduler will not run and the processes will be executed in the order they are currently in in the ready queue. Therefore, the order of execution is P1 -> P2 -> P3.

ii) Since it is a timer interrupt that has occurred, we can assume that a process, say P0 was currently running and completed its quantum. Therefore, after P1 is chosen for execution, the ready queue will be: PCB2 -> PCB3 -> PCB0.

2) a) i) T
This is because some other process waiting for the lock will continue waiting while the current process can re-enter the critical section.

ii) F
It needs to be atomic to avoid race condition.

iii) F
It is still possible to have a deadlock.

iv) T
Blocking can lead to a lot of overhead.

v) F
4 conditions must be satisfied for deadlock to occur: mutual exclusion, circular wait, hold and wait, no pre-emption.

b) i)
Rule: Available >= Need

				Available		
				A	B	C
				2	0	1
Process	Allocation	Need				
P2	3,1,2	1,0,0	+	3	1	2
				5	1	3
P1	1,3,2	3,1,0	+	1	3	2
				6	4	5
P3	2,0,1	2,1,3	+	2	0	1
				8	4	3

Therefore, one possible safe sequence is <P2,P1,P3>.
<P2,P3,P1> is also possible.

ii) No the request cannot be granted as there will be no safe sequence for completion.

c) i) Rd = -3 , wr = -9

ii) Since the reader waits for wr every single time even if there are readers already reading the document, only one reader will be allowed to the document which is not what is needed.

iii) We can utilize a variable called count and initialize it to 0. Let a semaphore r_count initialized to 1 protect this variable. We can then wait for wr only if the reader count is 0.

The reader code can be modified as follows:

```
Wait(rd);
Wait(r_count);
If(count == 0) {
    Count = count  + 1;
    Wait(wr);
}
Signal(r_count);
Do reading
Wait(r_count);
If(count == 1) {
  Signal(wr);
}
Count = count – 1
Signal(r_count);
Signal(rd);
```

3) a) i) **F**
Not always, example segmentation.

ii) **T**
The number of processes in the memory cannot be greater than the number of pre decided partitions in the memory.

iii) **F**
It will produce the largest hole.

iv) **F**
Not always the case. But sometimes, yes.

v) **T**
With virtual memory, we can have the size of the logical address space greater than that of the physical memory space.

b) i) Physical Address: | 4 bits | 9 bits |
Frame bits: 4, Address bits: 9

Logical Address: | 3 bits | 9 bits |
Page bits: 3, Address bits: 9

ii) Page 010_2 -> 2_{10}
Corresponds to – in the page table. Therefore, page fault.

Page 001_2 -> 1_{10}
Corresponds to 5_{10} in page table -> Frame number: 0101_2

Address: 101001001_2 (Keep the same)
Answer: Physical address **<u>0101</u> 101001001_2**

iii) There will be 2 page faults for page 2 and 3.

Page Number	Frame Number	Time of Use
0	0	11
1	5	8
2	2	12
3	1	7
4	12	10
5	-	-
6	3	9
7	-	-

c)
- If a process does not have enough pages in the memory for it to execute, it can lead to a high page fault rate known as thrashing. This can lead to low CPU utilization.
- Thrashing is when a process is busy bringing pages in and out of memory.
- The working – set model can be used to detect thrashing. If the total demand of the frames is greater than the total number of frames, thrashing has occurred.
- If this occurs, one of the processes is suspended.

4) a) i) **F**
It is used to cache file control blocks.

ii) **T**

iii) **T**
Due to symbolic and hard links.

iv) **F**
A cache is used to hold a copy of a data item that is stored somewhere.

v) **T**

b) The 5000th byte occurs in block 5. (5000/1024 = 4.88)
Contiguous: 1R (Support random access)
Linked: 5R (Has to read from Block 1 to Block 5)
Indexed: 1R (Supports random access via the inode)

c) When the block size increases, the file fits in a lesser number of blocks. In this case, it is faster to find and transfer blocks and hence there is a higher throughput. But this wastes space in the block and hence wastage of disk space.

--End of Answers--