Q1

- Indicate whether the following statements are true or false. Justify your answer.
  a) Buffering can be used to improve I/O efficiency for files that are being written and re-read rapidly.
  b) Process will be in waiting state after performing an I/O system call if non-blocking I/O is used.
  c) Device drivers are part of the kernel I/O subsystem.

Q1

- Indicate whether the following statements are true or false. Justify your answer.
  a) Buffering can be used to improve I/O efficiency for files that are being written and re-read rapidly.

     False: Buffering is mainly used to cope with device speed mismatch or transfer size mismatch.
  b) Process will be in waiting state after performing an I/O system call if non-blocking I/O is used.

     False: Using non-blocking I/O, the process will be able to continue to execute after the system call.
  c) Device drivers are part of the kernel I/O subsystem.

     False. Device drivers are not part of the kernel I/O subsystem.

Q2

Suppose that in a multiprogramming system, a process reads blocks of data from a file on disk for processing. As shown below, it reads one block of data at a time to a buffer using synchronous I/O and then processes the data.

```
while ( not end of file) {
        buffer <- read a block of data from disk using synchronous I/O;
        process data in buffer;
}
```

a) Discuss how the performance of the above process can be improved.

b) For a system running mainly with this type of processes, which file allocation scheme is best in terms of I/O performance?

Q2 (a)

- Discuss how the performance of the above process can be improved

→ Using double-buffer and asynchronous I/O

```
buffer2<- read a block of data from disk using asynchronous I/O;
while ( not end of file ) {
        while ( I/O not over ) do nothing;
        buffer1 <- buffer2;
        buffer2<- read next block of data from disk using asynchronous I/O;
        process data in buffer1;
}
```

Q2 (b)

For a system running mainly with this type of processes, which file allocation scheme is best in terms of I/O performance?

→ Contiguous file allocation is most suitable.

Q3

During his presentation, a salesman emphasized on the substantial effort his company has made to improve the performance of their UNIX version - one example he quoted was that the disk driver used the SCAN algorithm and also queued multiple requests within a cylinder in sector order. You bought a copy and wrote a program to randomly read 10,000 blocks spread across the disk. The performance measured was the same as what would be expected from FCFS algorithm. Was the salesman lying?

Q3 (a)

- Not necessarily.
  - if the requests are issued one at a time.
  - The disk driver has no opportunity for SCAN optimization (**SCAN=FCFS**)
- Solution: generates many concurrent I/Os

Q3 (b)

Under what circumstances could a disk scheduling discipline not improve the performance or even degrade performance of the system?

Q3 (b)

- Under light load conditions.
- If overhead for scheduling is significantly more than the average seek time.

Q4

Assume that a disk drive has 200 cylinders, numbered 0 to 199. The disk head starts at cylinder 0. A seek takes ($20 + 0.1 \times T$) milliseconds, where T is the number of cylinders to move. Rotational latency is 2 milliseconds and data transfer per request takes 8 milliseconds, assuming each request accesses the same amount of data. The following table shows the arrival time and destination cylinder number of requests:

Arrive Time (ms)	0	15	20	23	30	35	50	65	70	88
Cylinder Number	45	132	35	4	23	50	70	40	10	35

Compute the average time to service a request using the Shortest Seek Time First (SSTF) disk head scheduling algorithm.

What is the time for one request?

- Service time

= seek time + rotational latency + data transfer time

= 20+0.1*T +2 +8

Average Service Time

Arrive Time (ms)	0	15	20	23	30	35	50	65	70	88
Cylinder Number	45	132	35	4	23	50	70	40	10	35

Cylinder No.	Time Taken for Servicing the Request	Accumulated Time
45	$20 + 0.1 \times 45 + 2 + 8 = 34.5$	
35	$20 + 0.1 \times 10 + 2 + 8 = 31$	65.5
40	$20 + 0.1 \times 5 + 2 + 8 = 30.5$	96.0
35	$20 + 0.1 \times 5 + 2 + 8 = 30.5$	126.5
23	$20 + 0.1 \times 12 + 2 + 8 = 31.2$	157.7
10	$20 + 0.1 \times 13 + 2 + 8 = 31.3$	189.0
4	$20 + 0.1 \times 6 + 2 + 8 = 30.6$	219.6
50	$20 + 0.1 \times 46 + 2 + 8 = 34.6$	254.2
70	$20 + 0.1 \times 20 + 2 + 8 = 32$	286.2
132	$20 + 0.1 \times 62 + 2 + 8 = 36.2$	322.4

Average=322.4/10=32.24 milliseconds.