Question 1

Let me solve this step by step:

1. Initial position: Cylinder 100

Queue: 30, 85, 90, 100, 105, 110, 135, 145

2. Using SSTF, we'll service the requests closest to current position first. Let's trace the order:

- Start at 100

- Look for closest request to 100:

\* 100 itself (0 distance) - 1st request

\* 105 (5 units away) - 2nd request

\* 110 (5 units from 105) - 3rd request

\* 90 (20 units from 110) - 4th request

3. So before getting to 90:

- 100 is serviced (1st)

- 105 is serviced (2nd)

- 110 is serviced (3rd)

Therefore, the request for cylinder 90 is serviced after servicing 3 requests.

To verify this is the shortest path each time:

\* At 100: Options are 30(-70), 85(-15), 90(-10), 100(0), 105(+5), 110(+10), 135(+35), 145(+45)

→ 100 is closest

\* At 100: Now choosing between rest - 105 is closest (+5)

\* At 105: 110 is closest (+5)

\* At 110: 90 is closest (-20)

The answer is 3 requests.

Question 2

Let me solve this step by step.

1) First, let's clarify the scenario:

- Disk has cylinders 0 to 50 (51 cylinders total)

- Starting position: 11

- Request queue in order: 1, 36, 16, 34, 9, 12

2) Let's calculate for each algorithm:

FCFS (First Come First Serve):

- Path: 11 → 1 → 36 → 16 → 34 → 9 → 12

- Distance: |11-1| + |1-36| + |36-16| + |16-34| + |34-9| + |9-12|

- = 10 + 35 + 20 + 18 + 25 + 3

- Total = 111 cylinders

SSTF (Shortest Seek Time First):

- At 11, closest is: 12 (distance 1)

- At 12, closest is: 9 (distance 3)

- At 9, closest is: 1 (distance 8)

- At 1, closest is: 16 (distance 15)

- At 16, closest is: 34 (distance 18)

- At 34, closest is: 36 (distance 2)

- Path: 11 → 12 → 9 → 1 → 16 → 34 → 36

- Total = 1 + 3 + 8 + 15 + 18 + 2 = 47 cylinders

SCAN (moves until end even if no requests):

- Starting at 11, going towards 50

- Path: 11 → 12 → 16 → 34 → 36 → 50 → 9 → 1

- Distance: |11-50| + |50-1|

- = 39 + 49

- Total = 88 cylinders

LOOK (like SCAN but reverses at last request):

- Starting at 11, going towards 50

- Path: 11 → 12 → 16 → 34 → 36 → 9 → 1

- Distance: |11-36| + |36-1|

- = 25 + 35

- Total = 60 cylinders

Therefore:

FCFS: 111 cylinders

SSTF: 47 cylinders

SCAN: 88 cylinders

LOOK: 60 cylinders

The most efficient algorithm in this case is SSTF with 47 cylinders of total movement.

Question 3

Let me solve this step by step.

1) First, let's calculate total distance for SSTF:

Starting position: 50

Queue: 45, 20, 90, 10, 50, 60, 80, 25, 70

SSTF path:

- At 50, closest is: 50 (distance 0)

- At 50, closest is: 45 (distance 5)

- At 45, closest is: 60 (distance 15)

- At 60, closest is: 70 (distance 10)

- At 70, closest is: 80 (distance 10)

- At 80, closest is: 90 (distance 10)

- At 90, closest is: 25 (distance 65)

- At 25, closest is: 20 (distance 5)

- At 20, closest is: 10 (distance 10)

SSTF total = 0 + 5 + 15 + 10 + 10 + 10 + 65 + 5 + 10 = 130 tracks

2) Now for SCAN (moving towards 100 first):

Starting position: 50

Path: 50 → 60 → 70 → 80 → 90 → 100 → 45 → 25 → 20 → 10

Distance calculation:

- 50 to 100: 50 tracks

- 100 to 10: 90 tracks

SCAN total = 50 + 90 = 140 tracks

3) Additional distance traversed by SSTF compared to SCAN:

130 - 140 = -10 tracks

Therefore, SSTF traverses 10 tracks LESS than SCAN. The answer is -10 tracks.

To verify:

- SSTF: 130 tracks

- SCAN: 140 tracks

- Difference (SSTF - SCAN): -10 tracks

The negative answer means SSTF is more efficient in this case by 10 tracks.

Question 4

\*\*Sequential organization\*\* is simple and efficient for reading data in order but becomes inefficient for searching large datasets, as it requires scanning records sequentially. It's best for tasks where data access follows a predictable pattern.

\*\*Indexed organization\*\* improves search efficiency by using an index to locate data quickly. It allows faster retrieval compared to sequential but may require additional storage and overhead for maintaining the index.

\*\*Hashed organization\*\* offers the fastest access for retrieval and insertion by using a hash function to determine the record's location. However, it struggles with collision handling and is less efficient for range queries.

Each method's effectiveness depends on the specific use case:

- \*\*Sequential\*\*: Efficient for ordered access.

- \*\*Indexed\*\*: Balanced for search and retrieval.

- \*\*Hashed\*\*: Best for direct lookups, poor for sequential access.

Question 5

A comprehensive file management strategy for a multinational corporation should incorporate the following:

1. \*\*Cloud-based storage\*\*: Use a cloud solution (e.g., AWS, Google Cloud) for scalable, centralized file management, ensuring high availability and accessibility across global offices.

2. \*\*Role-based access control (RBAC)\*\*: Implement RBAC to ensure that employees can access only the files they need, enhancing security and reducing data breaches.

3. \*\*Data encryption\*\*: Use end-to-end encryption (both at rest and in transit) to secure sensitive information, ensuring compliance with global data protection regulations (e.g., GDPR).

4. \*\*Backup and disaster recovery\*\*: Establish regular automated backups and a disaster recovery plan to safeguard data against loss or breaches.

5. \*\*Content Delivery Networks (CDN)\*\*: Employ CDNs to enhance file access speed for dispersed offices, reducing latency.

6. \*\*Audit trails\*\*: Maintain comprehensive logging for tracking file access and modifications to monitor compliance and detect suspicious activities.

This strategy balances security, accessibility, and scalability for global operations.