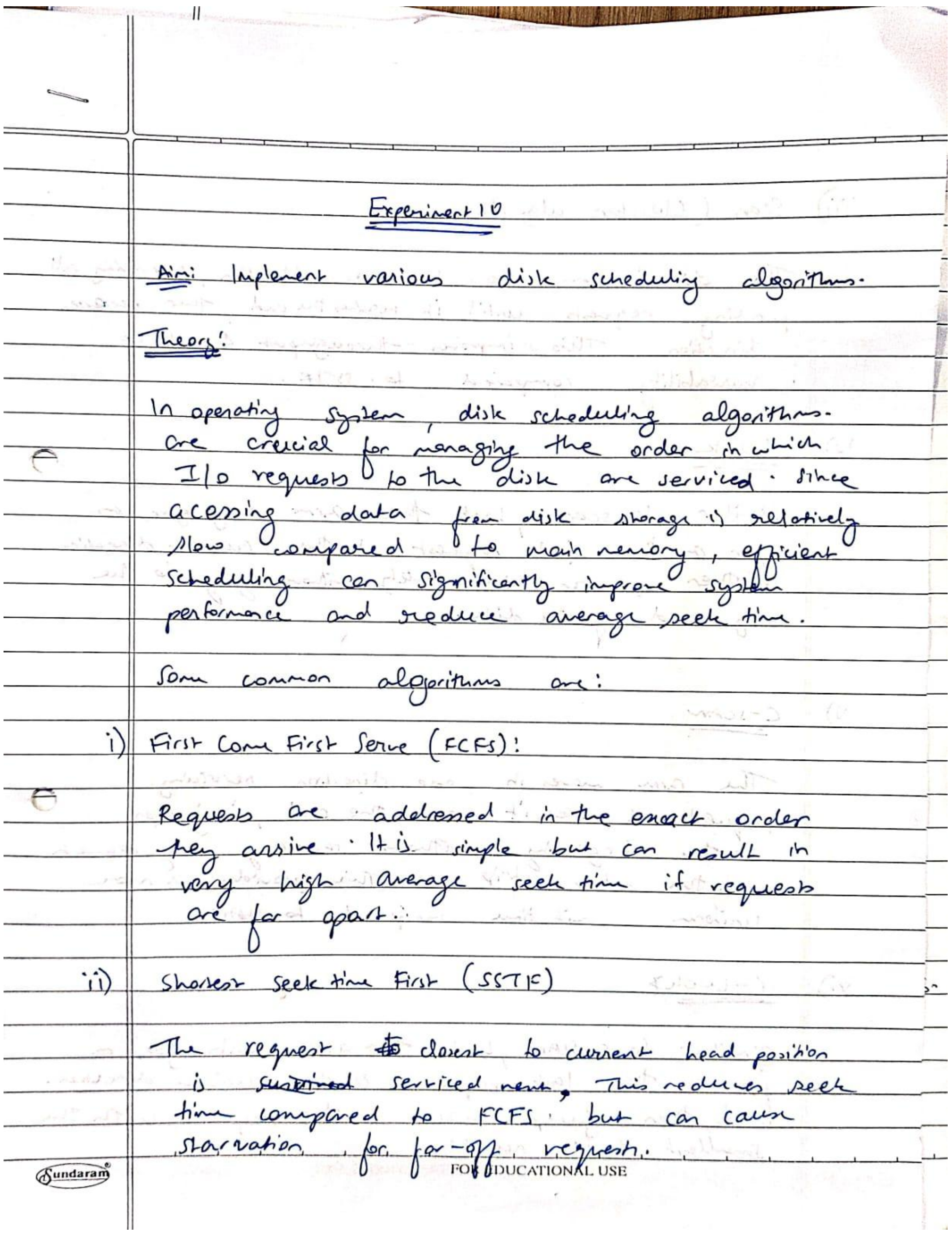
EXPERIMENT 10



Experiment 10

Aim: Implement various disk scheduling algorithms.

Theory:

In operating system, disk scheduling algorithms are crucial for managing the order in which I/O requests to the disk are serviced. Since acessing data from disk storage is relatively slow compared to main memory, efficient scheduling can significantly improve system performance and reduce average seek time.

Some common algorithms are:

i) First Come First Serve (FCFS):

Requests are addressed in the exact order they arrive. It is simple but can result in very high average seek time if requests are far apart.

ii) Shortest Seek time First (SSTF)

The request ~~to~~ closest to current head position is ~~sustained~~ serviced next, This reduces seek time compared to FCFS, but can cause starvation for far-off request.

Sundaram®

FOR EDUCATIONAL USE

Conclusion

Disk scheduling ~~st~~ algorithms like FCFS, SSTF, SCAN & C-SCAN optimize disk I/O performance. Each algorithm has its ~~a~~ advantage with trade-off in efficiency & fairness.

Code

```
import java.util.*;
public class Main {
 static int sum(Vector<Integer> vc, int hp) {
  int sum = 0;
  sum += Math.abs(vc.get(0) - hp);
  for (int i = 1; i < vc.size(); i++)
   sum += Math.abs(vc.get(i) - vc.get(i - 1));
  return sum;
 }

 static void FCFS(Vector<Integer> vc, int hp) {
  int sum = 0;
  for (int i = 1; i < vc.size(); i++)
   sum += Math.abs(vc.get(i) - vc.get(i - 1));
  sum += Math.abs(vc.firstElement() - hp);
  System.out.println("Access Sequence: " + vc);
  System.out.println("Total number of disc movements: " + sum);
 }

 static void SSTF(Vector<Integer> queue, int hp) {
  Vector<Integer> vc = new Vector<>(queue);
  int now = hp;
  Vector<Integer> as = new Vector<>();
  while (!vc.isEmpty()) {
   int minDist = Integer.MAX_VALUE, chosen = -1;
   for (int i = 0; i < vc.size(); i++) {
    int dist = Math.abs(vc.get(i) - now);
    if (dist < minDist) {
     minDist = dist;
     chosen = i;
    }
  }
  now = vc.get(chosen);
  as.add(now);
  vc.remove(chosen);
 }
 int sum = sum(as, hp);
 System.out.println("Access Sequence: " + as);
 System.out.println("Total number of disc movements: " + sum);
}

static void Scan(Vector<Integer> queue, int hp, int start, int end, int d) {
 Vector<Integer> vc = new Vector<>(queue);
 vc.add(hp);
 if (hp != end) vc.add(end);
 if (hp != start) vc.add(start);
 vc.sort(null);
 Vector<Integer> as = new Vector<>();
 if (d == 1) {
  if (hp == start) vc.remove(end);
  for (int i = vc.indexOf(hp) + 1; i <= vc.size() - 1; i++) as.add(vc.get(i));
  for (int i = vc.indexOf(hp) - 1; i > 0; i--) as.add(vc.get(i));
 } else {
  if (hp == end) vc.remove(start);
  for (int i = vc.indexOf(hp) - 1; i >= 0; i--) as.add(vc.get(i));
  for (int i = vc.indexOf(hp) + 1; i < vc.size() - 1; i++) as.add(vc.get(i));
 }
 int sum = sum(as, hp);
 System.out.println("Access Sequence: " + as);
 System.out.println("Total number of disc movements: " + sum);
}

static void CScan(Vector<Integer> queue, int hp, int start, int end, int d) {
 Vector<Integer> vc = new Vector<>(queue);
 vc.add(hp);
```

```
  if (hp != end) vc.add(end);
  if (hp != start) vc.add(start);
  vc.sort(null);
  Vector<Integer> as = new Vector<>();
  if (d == 1) {
   if (hp == start) vc.remove(end);
   for (int i = vc.indexOf(hp) + 1; i <= vc.size() - 1; i++) as.add(vc.get(i));
   for (int i = 0; i < vc.indexOf(hp); i++) as.add(vc.get(i));
  } else {
   if (hp == end) vc.remove(start);
   for (int i = vc.indexOf(hp) - 1; i >= 0; i--) as.add(vc.get(i));
   for (int i = vc.size() - 1; i > vc.indexOf(hp); i--) as.add(vc.get(i));
  }
  int sum = sum(as, hp);
  System.out.println("Access Sequence: " + as);
  System.out.println("Total number of disc movements: " + sum);
 }

 static void Look(Vector<Integer> queue, int hp, int d) {
  Vector<Integer> vc = new Vector<>(queue);
  vc.add(hp);
  vc.sort(null);
  Vector<Integer> as = new Vector<>();
  if (d == 1) {
   for (int i = vc.indexOf(hp) + 1; i <= vc.indexOf(vc.lastElement()); i++)
as.add(vc.get(i));
   for (int i = vc.indexOf(hp) - 1; i >= 0; i--) as.add(vc.get(i));
  } else {
   for (int i = vc.indexOf(hp) - 1; i >= 0; i--) as.add(vc.get(i));
   for (int i = vc.indexOf(hp) + 1; i <= vc.size() - 1; i++) as.add(vc.get(i));
  }
  int sum = sum(as, hp);
  System.out.println("Access Sequence: " + as);
  System.out.println("Total number of disc movements: " + sum);
```

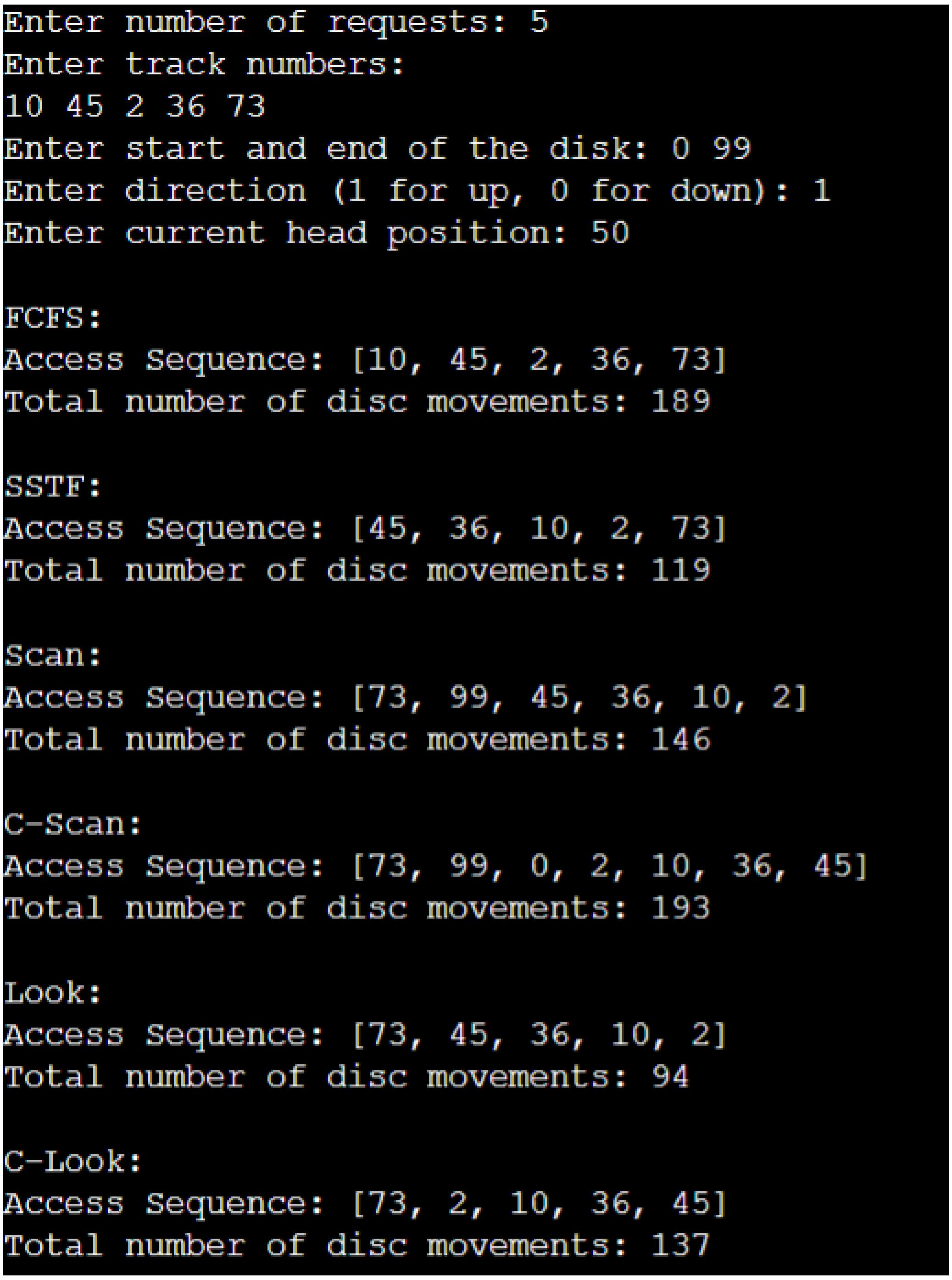
```java
}

static void CLook(Vector<Integer> queue, int hp, int d) {
 Vector<Integer> vc = new Vector<>(queue);
 vc.add(hp);
 vc.sort(null);
 Vector<Integer> as = new Vector<>();
 if (d == 1) {
  for (int i = vc.indexOf(hp) + 1; i <= vc.size() - 1; i++) as.add(vc.get(i));
  for (int i = 0; i < vc.indexOf(hp); i++) as.add(vc.get(i));
 } else {
  for (int i = vc.indexOf(hp) - 1; i >= 0; i--) as.add(vc.get(i));
  for (int i = vc.size() - 1; i > vc.indexOf(hp); i--) as.add(vc.get(i));
 }
 int sum = sum(as, hp);
 System.out.println("Access Sequence: " + as);
 System.out.println("Total number of disc movements: " + sum);
}

public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);
 Vector<Integer> queue = new Vector<>();
 System.out.print("Enter number of requests: ");
 int n = sc.nextInt();
 System.out.println("Enter track numbers:");
 for (int i = 0; i < n; i++) queue.add(sc.nextInt());
 System.out.print("Enter start and end of the disk: ");
 int start = sc.nextInt(), end = sc.nextInt();
 System.out.print("Enter direction (1 for up, 0 for down): ");
 int d = sc.nextInt();
 System.out.print("Enter current head position: ");
 int hp = sc.nextInt();
 System.out.println();
 System.out.println("FCFS: ");
```

```
  FCFS(queue, hp);
  System.out.println();
  System.out.println("SSTF: ");
  SSTF(queue, hp);
  System.out.println();
  System.out.println("Scan: ");
Scan(queue, hp, start, end, d);
System.out.println();
System.out.println("C-Scan: ");
CScan(queue, hp, start, end, d);
System.out.println();
System.out.println("Look: ");
Look(queue, hp, d);
System.out.println();
System.out.println("C-Look: ");
CLook(queue, hp, d);
sc.close();
}
}
```

Output



```
Enter number of requests: 5
Enter track numbers:
10 45 2 36 73
Enter start and end of the disk: 0 99
Enter direction (1 for up, 0 for down): 1
Enter current head position: 50

FCFS:
Access Sequence: [10, 45, 2, 36, 73]
Total number of disc movements: 189

SSTF:
Access Sequence: [45, 36, 10, 2, 73]
Total number of disc movements: 119

Scan:
Access Sequence: [73, 99, 45, 36, 10, 2]
Total number of disc movements: 146

C-Scan:
Access Sequence: [73, 99, 0, 2, 10, 36, 45]
Total number of disc movements: 193

Look:
Access Sequence: [73, 45, 36, 10, 2]
Total number of disc movements: 94

C-Look:
Access Sequence: [73, 2, 10, 36, 45]
Total number of disc movements: 137
```