JUNAID GIRKAR
60004190057
SE COMPS A-3

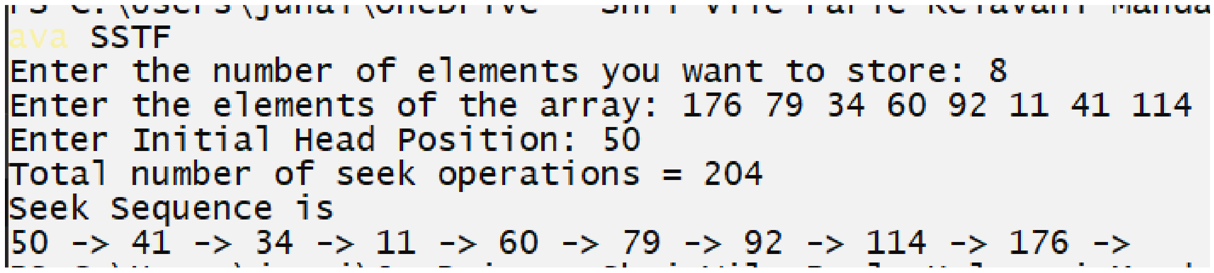
OPERATING SYSTEMS
EXPERIMENT 10
CODE & OUTPUT

SSTF ALGORITHM:

CODE:

```
import java.util.Scanner;

class node {

    int distance = 0;
    boolean accessed = false;
}

public class SSTF {

    // Calculates difference of each track number with the head
position
    public static void calculateDifference(int queue[],
                                        int head, node diff[])

    {
        for (int i = 0; i < diff.length; i++)
            diff[i].distance = Math.abs(queue[i] - head);
    }

    public static int findMin(node diff[])
    {
        int index = -1, minimum = Integer.MAX_VALUE;

        for (int i = 0; i < diff.length; i++) {
            if (!diff[i].accessed && minimum > diff[i].distance) {

                minimum = diff[i].distance;
                index = i;
            }
        }
        return index;
    }

    public static void shortestSeekTimeFirst(int request[],int head)

    {
        if (request.length == 0)
            return;

        // create array of objects of class node
        node diff[] = new node[request.length];

        for (int i = 0; i < diff.length; i++)

            diff[i] = new node();

        int seek_count = 0;

        // stores sequence in which disk access is done
        int[] seek_sequence = new int[request.length + 1];

        for (int i = 0; i < request.length; i++) {

            seek_sequence[i] = head;
            calculateDifference(request, head, diff);

            int index = findMin(diff);

            diff[index].accessed = true;
            seek_count += diff[index].distance;
            head = request[index];
        }

        // for last accessed track
        seek_sequence[seek_sequence.length - 1] = head;

        System.out.println("Total number of seek operations = "
                                                     + seek_count);

        System.out.println("Seek Sequence is");

        // print the sequence
        for (int i = 0; i < seek_sequence.length; i++)
            System.out.print(seek_sequence[i] + " -> ");

    }

    public static void main(String[] args)
    {
        int n;
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the number of elements you want to
store: ");
        n=sc.nextInt();

        int[] arr = new int[n];
        System.out.print("Enter the elements of the array: ");

        for(int i=0; i<n; i++)
        {
            arr[i]=sc.nextInt();
        }

        System.out.print("Enter Initial Head Position: ");
        int head_pos = sc.nextInt();
        shortestSeekTimeFirst(arr, head_pos);
    }
}
```

OUTPUT:



```
ava SSTF
Enter the number of elements you want to store: 8
Enter the elements of the array: 176 79 34 60 92 11 41 114
Enter Initial Head Position: 50
Total number of seek operations = 204
Seek Sequence is
50 -> 41 -> 34 -> 11 -> 60 -> 79 -> 92 -> 114 -> 176 ->
```

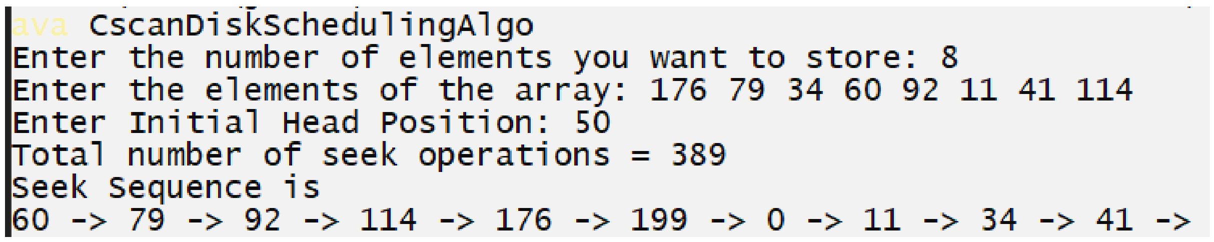
CSCAN ALGORITHM:

CODE:

```
import java.util.*;

class CscanDiskSchedulingAlgo {

    static int size = 8;
    static int disk_size = 200;

    public static void CSCAN(int arr[], int head)
    {
        int seek_count = 0;
        int distance, cur_track;

        Vector<Integer> left = new Vector<Integer>();
        Vector<Integer> right = new Vector<Integer>();
        Vector<Integer> seek_sequence
            = new Vector<Integer>();


        left.add(0);
        right.add(disk_size - 1);


        for (int i = 0; i < size; i++) {
            if (arr[i] < head)
                left.add(arr[i]);
            if (arr[i] > head)
```

```
                right.add(arr[i]);
        }

        // Sorting left and right vectors
        Collections.sort(left);
        Collections.sort(right);

        // First service the requests
        // on the right side of the
        // head.
        for (int i = 0; i < right.size(); i++) {
            cur_track = right.get(i);

            seek_sequence.add(cur_track);

            distance = Math.abs(cur_track - head);

            seek_count += distance;

            head = cur_track;
        }

        // Once reached the right end
        // jump to the beggining.
        head = 0;

        // adding seek count for head returning from 199 to
        // 0
        seek_count += (disk_size - 1);

        // Now service the requests again
        // which are left.
        for (int i = 0; i < left.size(); i++) {
            cur_track = left.get(i);

            seek_sequence.add(cur_track);

            distance = Math.abs(cur_track - head);

            seek_count += distance;
            head = cur_track;
        }

        System.out.println("Total number of seek "+ "operations =
" + seek_count);

        System.out.println("Seek Sequence is");

        for (int i = 0; i < seek_sequence.size(); i++) {
            System.out.print(seek_sequence.get(i) + " -> ");
        }
    }

    public static void main(String[] args) throws Exception
    {

        int n;
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the number of elements you want to
store: ");
        n=sc.nextInt();

        int[] arr = new int[n];
        System.out.print("Enter the elements of the array: ");

        for(int i=0; i<n; i++)
        {
            arr[i]=sc.nextInt();
        }

        System.out.print("Enter Initial Head Position: ");
        int head_pos = sc.nextInt();

        CSCAN(arr, head_pos);
    }
}
```

OUTPUT:



```
ava CscanDiskSchedulingAlgo
Enter the number of elements you want to store: 8
Enter the elements of the array: 176 79 34 60 92 11 41 114
Enter Initial Head Position: 50
Total number of seek operations = 389
Seek Sequence is
60 -> 79 -> 92 -> 114 -> 176 -> 199 -> 0 -> 11 -> 34 -> 41 ->
```

CONCLUSION: We learnt about different Disk Scheduling Algorithms, the advantages and disadvantages of each and implemented SSTF and CSCAN in a java program.