import java.util.\*;

public class DiskScheduling {

    static int[] requests;

    static int head;

    static int diskSize;

    // FCFS (First Come First Serve)

    public static void FCFS() {

        int totalSeekCount = 0;

        int curTrack;

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

            curTrack = requests[i];

            totalSeekCount += Math.abs(curTrack - head);

            head = curTrack;

        }

        System.out.println("FCFS Total Seek Count: " + totalSeekCount);

    }

    // SSTF (Shortest Seek Time First)

    public static void SSTF() {

        int totalSeekCount = 0;

        boolean[] visited = new boolean[requests.length];

        int count = 0;

        int curTrack;

        while (count < requests.length) {

            int min = Integer.MAX\_VALUE;

            int index = -1;

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

                if (!visited[i] && Math.abs(requests[i] - head) < min) {

                    min = Math.abs(requests[i] - head);

                    index = i;

                }

            }

            visited[index] = true;

            curTrack = requests[index];

            totalSeekCount += Math.abs(curTrack - head);

            head = curTrack;

            count++;

        }

        System.out.println("SSTF Total Seek Count: " + totalSeekCount);

    }

    // SCAN

    public static void SCAN() {

        int totalSeekCount = 0;

        ArrayList<Integer> left = new ArrayList<>();

        ArrayList<Integer> right = new ArrayList<>();

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

            if (requests[i] < head)

                left.add(requests[i]);

            else

                right.add(requests[i]);

        }

        Collections.sort(left, Collections.reverseOrder());

        Collections.sort(right);

        for (int i = 0; i < right.size(); i++) {

            totalSeekCount += Math.abs(right.get(i) - head);

            head = right.get(i);

        }

        totalSeekCount += Math.abs(head - (diskSize - 1));

        head = diskSize - 1;

        for (int i = 0; i < left.size(); i++) {

            totalSeekCount += Math.abs(left.get(i) - head);

            head = left.get(i);

        }

        System.out.println("SCAN Total Seek Count: " + totalSeekCount);

    }

    // C-SCAN (Circular SCAN)

    public static void C\_Scan() {

        int totalSeekCount = 0;

        ArrayList<Integer> left = new ArrayList<>();

        ArrayList<Integer> right = new ArrayList<>();

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

            if (requests[i] < head)

                left.add(requests[i]);

            else

                right.add(requests[i]);

        }

        Collections.sort(left);

        Collections.sort(right);

        for (int i = 0; i < right.size(); i++) {

            totalSeekCount += Math.abs(right.get(i) - head);

            head = right.get(i);

        }

        totalSeekCount += Math.abs(head - (diskSize - 1));

        head = 0;

        for (int i = 0; i < left.size(); i++) {

            totalSeekCount += Math.abs(left.get(i) - head);

            head = left.get(i);

        }

        System.out.println("C-SCAN Total Seek Count: " + totalSeekCount);

    }

    // LOOK

    public static void LOOK() {

        int totalSeekCount = 0;

        ArrayList<Integer> left = new ArrayList<>();

        ArrayList<Integer> right = new ArrayList<>();

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

            if (requests[i] < head)

                left.add(requests[i]);

            else

                right.add(requests[i]);

        }

        Collections.sort(left, Collections.reverseOrder());

        Collections.sort(right);

        for (int i = 0; i < right.size(); i++) {

            totalSeekCount += Math.abs(right.get(i) - head);

            head = right.get(i);

        }

        for (int i = 0; i < left.size(); i++) {

            totalSeekCount += Math.abs(left.get(i) - head);

            head = left.get(i);

        }

        System.out.println("LOOK Total Seek Count: " + totalSeekCount);

    }

    // C-LOOK (Circular LOOK)

    public static void C\_Look() {

        int totalSeekCount = 0;

        ArrayList<Integer> left = new ArrayList<>();

        ArrayList<Integer> right = new ArrayList<>();

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

            if (requests[i] < head)

                left.add(requests[i]);

            else

                right.add(requests[i]);

        }

        Collections.sort(left);

        Collections.sort(right);

        for (int i = 0; i < right.size(); i++) {

            totalSeekCount += Math.abs(right.get(i) - head);

            head = right.get(i);

        }

        totalSeekCount += Math.abs(head - 0);

        head = 0;

        for (int i = 0; i < left.size(); i++) {

            totalSeekCount += Math.abs(left.get(i) - head);

            head = left.get(i);

        }

        System.out.println("C-LOOK Total Seek Count: " + totalSeekCount);

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the total number of disk requests: ");

        int n = scanner.nextInt();

        requests = new int[n];

        System.out.println("Enter the disk requests: ");

        for (int i = 0; i < n; i++) {

            requests[i] = scanner.nextInt();

        }

        System.out.println("Enter the initial head position: ");

        head = scanner.nextInt();

        System.out.println("Enter the total disk size: ");

        diskSize = scanner.nextInt();

        System.out.println("\nDisk Scheduling Algorithms:");

        System.out.println("1. FCFS");

        System.out.println("2. SSTF");

        System.out.println("3. SCAN");

        System.out.println("4. C-SCAN");

        System.out.println("5. LOOK");

        System.out.println("6. C-LOOK");

        System.out.print("Enter your choice: ");

        int choice = scanner.nextInt();

        switch (choice) {

            case 1:

                FCFS();

                break;

            case 2:

                SSTF();

                break;

            case 3:

                SCAN();

                break;

            case 4:

                C\_Scan();

                break;

            case 5:

                LOOK();

                break;

            case 6:

                C\_Look();

                break;

            default:

                System.out.println("Invalid choice.");

        }

        scanner.close();

    }

}