Assignment No. 8

Name : Omkar Hotkar
Roll No : 30

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

void SSTF(int requests[], int n, int head) {
    int visited[n];
    int totalSeek = 0;
    int current = head;

    for (int i = 0; i < n; i++)
        visited[i] = 0;

    printf("\nSSTF Sequence: %d ", current);

    for (int i = 0; i < n; i++) {
        int minDist = 10000, index = -1;

        for (int j = 0; j < n; j++) {
            if (!visited[j] && abs(requests[j] - current) < minDist) {
                minDist = abs(requests[j] - current);
                index = j;
            }
        }

        visited[index] = 1;
        totalSeek += minDist;
        current = requests[index];
        printf("-> %d ", current);
    }

    printf("\nTotal Seek Time (SSTF): %d\n", totalSeek);
}

void SCAN(int requests[], int n, int head, int diskSize) {
    int totalSeek = 0;
    int direction = 1; // 1 means moving away from spindle, 0 means towards spindle
    int temp, i, j;

    // Sort requests
    for (i = 0; i < n-1; i++) {
        for (j = i+1; j < n; j++) {
            if (requests[i] > requests[j]) {
```

```
            temp = requests[i];
            requests[i] = requests[j];
            requests[j] = temp;
        }
    }
}

printf("\nSCAN Sequence: %d ", head);

// Find the division point
int idx;
for (i = 0; i < n; i++) {
    if (requests[i] > head) {
        idx = i;
        break;
    }
}

// Moving away from spindle (towards higher track)
for (i = idx; i < n; i++) {
    totalSeek += abs(requests[i] - head);
    head = requests[i];
    printf("-> %d ", head);
}

// Go to the end of disk
if (head != diskSize-1) {
    totalSeek += abs((diskSize-1) - head);
    head = diskSize-1;
    printf("-> %d ", head);
}

// Then move towards spindle (track 0)
for (i = idx-1; i >= 0; i--) {
    totalSeek += abs(requests[i] - head);
    head = requests[i];
    printf("-> %d ", head);
}

printf("\nTotal Seek Time (SCAN): %d\n", totalSeek);
}

void CLOOK(int requests[], int n, int head) {
    int totalSeek = 0, temp, i, j;

    // Sort requests
    for (i = 0; i < n-1; i++) {
        for (j = i+1; j < n; j++) {
```

```
            if (requests[i] > requests[j]) {
                temp = requests[i];
                requests[i] = requests[j];
                requests[j] = temp;
            }
        }
    }

    printf("\nC-LOOK Sequence: %d ", head);

    int idx;
    for (i = 0; i < n; i++) {
        if (requests[i] > head) {
            idx = i;
            break;
        }
    }

    // Move towards higher track first
    for (i = idx; i < n; i++) {
        totalSeek += abs(requests[i] - head);
        head = requests[i];
        printf("-> %d ", head);
    }

    // Jump to the smallest request
    if (idx > 0) {
        totalSeek += abs(head - requests[0]);
        head = requests[0];
        printf("-> %d ", head);

        for (i = 1; i < idx; i++) {
            totalSeek += abs(requests[i] - head);
            head = requests[i];
            printf("-> %d ", head);
        }
    }

    printf("\nTotal Seek Time (C-LOOK): %d\n", totalSeek);
}

int main() {
    int n, head, diskSize;

    printf("Enter number of disk requests: ");
    scanf("%d", &n);

    int requests[n];
```

```
    printf("Enter the disk requests: ");
    for (int i = 0; i < n; i++)
        scanf("%d", &requests[i]);

    printf("Enter initial head position: ");
    scanf("%d", &head);

    printf("Enter disk size: ");
    scanf("%d", &diskSize);

    SSTF(requests, n, head);
    SCAN(requests, n, head, diskSize);
    CLOOK(requests, n, head);

    return 0;
}
```

Output :

```
Number of disk requests: 8
Disk requests: 95 180 34 119 11 123 62 64
Initial head position: 50
Disk size: 200
```