ASSIGNMENT: 08

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
void SSTF(int requests[], int n, int head) {
    int visited[n];
    for (int i = 0; i < n; i++) visited[i] = 0;
    int seek_count = 0;
    int current_head = head;
    printf("\nSSTF Algorithm:\nOrder of servicing requests: %d ", current_head);
    for (int i = 0; i < n; i++) {
        int minDistance = 1e9, idx = -1;
        for (int j = 0; j < n; j++) {
            if (!visited[j]) {
                int dist = abs(requests[j] - current_head);
                if (dist < minDistance) {
                    minDistance = dist;
                    idx = j;
                }
            }
        }
        visited[idx] = 1;
        seek_count += minDistance;
        current_head = requests[idx];
        printf("-> %d ", current_head);
    }
    printf("\nTotal seek operations: %d\n", seek_count);
}
void SCAN(int requests[], int n, int head, int disk_size, int direction) {
    int seek_count = 0;
    int distance, cur_track;
    int left[n+1], right[n+1];  // +1 for adding end values
    int l = 0, r = 0;
    for (int i = 0; i < n; i++) {
        if (requests[i] < head)
            left[l++] = requests[i];
        else
            right[r++] = requests[i];
    }
    if (direction == 0)
        left[l++] = 0;
    else
        right[r++] = disk_size - 1;
    // Sort left
    for (int i = 0; i < l - 1; i++) {
        for (int j = i + 1; j < l; j++) {
            if (left[i] > left[j]) {
                int temp = left[i];
                left[i] = left[j];
                left[j] = temp;Disk Scheduling Algorithms
1. SSTF
```

```
2. SCAN
3. C-LOOK
4. Exit
Enter choice: 1
SSTF Algorithm:
Order of servicing requests: 50 -> 43 -> 24 -> 16 -> 82 -> 140 -> 170 -> 190
Total seek operations: 208
Disk Scheduling Algorithms
1. SSTF
2. SCAN
3. C-LOOK
4. Exit
Enter choice: 2
Enter direction (0 for left, 1 for right): 1
SCAN Algorithm:
Order of servicing requests: 50 -> 82 -> 140 -> 170 -> 190 -> 199 -> 43 -> 24 -> 16
Total seek operations: 332
Disk Scheduling Algorithms
1. SSTF
2. SCAN
3. C-LOOK
4. Exit
Enter choice: 3
C-LOOK Algorithm:
Order of servicing requests: 50 -> 82 -> 140 -> 170 -> 190 -> 16 -> 24 -> 43
Total seek operations: 341
Disk Scheduling Algorithms
1. SSTF
2. SCAN
3. C-LOOK
4. Exit
Enter choice: 4
Exiting...
```

```c
            }
        }
    }
    // Sort right
    for (int i = 0; i < r - 1; i++) {
        for (int j = i + 1; j < r; j++) {
            if (right[i] > right[j]) {
                int temp = right[i];
                right[i] = right[j];
                right[j] = temp;
            }
        }
    }
    int cur_pos = head;
    printf("\nSCAN Algorithm:\nOrder of servicing requests: %d ", cur_pos);
    if (direction == 0) { // left first
        for (int i = l - 1; i >= 0; i--) {
            distance = abs(cur_pos - left[i]);
            seek_count += distance;
            cur_pos = left[i];
            printf("-> %d ", cur_pos);
        }
        for (int i = 0; i < r; i++) {
            distance = abs(cur_pos - right[i]);
            seek_count += distance;
            cur_pos = right[i];
            printf("-> %d ", cur_pos);
        }
    } else { // right first
        for (int i = 0; i < r; i++) {
            distance = abs(cur_pos - right[i]);
            seek_count += distance;
            cur_pos = right[i];
            printf("-> %d ", cur_pos);
        }
        for (int i = l - 1; i >= 0; i--) {
            distance = abs(cur_pos - left[i]);
            seek_count += distance;
            cur_pos = left[i];
            printf("-> %d ", cur_pos);
        }
    }
    printf("\nTotal seek operations: %d\n", seek_count);
}
void C_LOOK(int requests[], int n, int head) {
    int seek_count = 0;
    int distance, cur_pos = head;
    int left[n], right[n];
    int l = 0, r = 0;
    for (int i = 0; i < n; i++) {
        if (requests[i] < head)
            left[l++] = requests[i];
        else
            right[r++] = requests[i];
    }
    // Sort left
    for (int i = 0; i < l - 1; i++) {
        for (int j = i + 1; j < l; j++) {
            if (left[i] > left[j]) {
                int temp = left[i];
                left[i] = left[j];
                left[j] = temp;
            }
        }
    }
    // Sort right
    for (int i = 0; i < r - 1; i++) {
        for (int j = i + 1; j < r; j++) {
            if (right[i] > right[j]) {
                int temp = right[i];
                right[i] = right[j];
                right[j] = temp;
            }
        }
    }
    printf("\nC-LOOK Algorithm:\nOrder of servicing requests: %d ", cur_pos);
    // Service right side requests
    for (int i = 0; i < r; i++) {
        distance = abs(right[i] - cur_pos);
        seek_count += distance;
        cur_pos = right[i];
        printf("-> %d ", cur_pos);
    }
    // Jump to leftmost request on left side
    if (l > 0) {
        distance = abs(cur_pos - left[0]);
        seek_count += distance;
        cur_pos = left[0];
        printf("-> %d ", cur_pos);
        for (int i = 1; i < l; i++) {
            distance = abs(left[i] - cur_pos);
            seek_count += distance;
            cur_pos = left[i];
            printf("-> %d ", cur_pos);
        }
    }
    printf("\nTotal seek operations: %d\n", seek_count);
}
int main() {
    int requests[] = {82, 170, 43, 140, 24, 16, 190};
    int n = sizeof(requests) / sizeof(requests[0]);
    int head = 50;
    int disk_size = 200;
    int choice, direction;
    while (1) {
        printf("\nDisk Scheduling Algorithms\n");
        printf("1. SSTF\n2. SCAN\n3. C-LOOK\n4. Exit\nEnter choice: ");
        scanf("%d", &choice);
        switch(choice) {
            case 1:
                SSTF(requests, n, head);
                break;
            case 2:
                printf("Enter direction (0 for left, 1 for right): ");
                scanf("%d", &direction);
                SCAN(requests, n, head, disk_size, direction);
                break;
            case 3:
                C_LOOK(requests, n, head);
                break;
            case 4:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice! Try again.\n");
        }
    }
    return 0;
}
```

OUTPUT:-

```
*******************************************************************************
Disk Scheduling Algorithms
1. SSTF
2. SCAN
3. C-LOOK
4. Exit
Enter choice: 1
SSTF Algorithm:
Order of servicing requests: 50 -> 43 -> 24 -> 16 -> 82 -> 140 -> 170 -> 190
Total seek operations: 208
Disk Scheduling Algorithms
1. SSTF
2. SCAN
3. C-LOOK
4. Exit
Enter choice: 2
Enter direction (0 for left, 1 for right): 1
SCAN Algorithm:
Order of servicing requests: 50 -> 82 -> 140 -> 170 -> 190 -> 199 -> 43 -> 24 -> 16
Total seek operations: 332
Disk Scheduling Algorithms
1. SSTF
2. SCAN
3. C-LOOK
4. Exit
Enter choice: 3
C-LOOK Algorithm:
Order of servicing requests: 50 -> 82 -> 140 -> 170 -> 190 -> 16 -> 24 -> 43
Total seek operations: 341
Disk Scheduling Algorithms
1. SSTF
2. SCAN
3. C-LOOK
4. Exit
Enter choice: 4
Exiting...
```