

Scan

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

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

```
int main() {
```

```
    int queue[20], n, head, i, j, temp, total = 0, disk_size;
```

```
    int seek_sequence[20], count = 0; // to store sequence of head movement
```

```
    float avg;
```

```
    // Input number of requests
```

```
    printf("Enter the number of disk requests: ");
```

```
    scanf("%d", &n);
```

```
    // Input the requests (track numbers)
```

```
    printf("Enter the disk requests (track numbers): ");
```

```
    for (i = 0; i < n; i++) {
```

```
        scanf("%d", &queue[i]);
```

```
    }
```

```
    // Input disk size
```

```
    printf("Enter total disk size (e.g., 200): ");
```

```
    scanf("%d", &disk_size);
```

```
    // Input initial head position
```

```
    printf("Enter initial head position: ");
```

```
    scanf("%d", &head);
```

```
    // Add head position to the queue (for sorting purpose)
```

```
    queue[n] = head;
```

```
    n++;
```

```
// Sort the queue (Ascending order)
```

```
for (i = 0; i < n; i++) {  
    for (j = i + 1; j < n; j++) {  
        if (queue[i] > queue[j]) {  
            temp = queue[i];  
            queue[i] = queue[j];  
            queue[j] = temp;  
        }  
    }  
}
```

```
// Find the index of current head position after sorting
```

```
int pos;  
for (i = 0; i < n; i++) {  
    if (queue[i] == head) {  
        pos = i;  
        break;  
    }  
}
```

```
// Moving away from spindle means towards higher tracks (right side first)
```

```
printf("\nHead movement order:\n");
```

```
// Move right side (towards higher track numbers)
```

```
for (i = pos; i < n; i++) {  
    seek_sequence[count++] = queue[i];  
}
```

```
// Then move to the end (maximum track)
```

```
if (seek_sequence[count - 1] != disk_size - 1) {  
    seek_sequence[count++] = disk_size - 1;
```

```

}

// Then move back to the start (lower track numbers)
for (i = pos - 1; i >= 0; i--) {
    seek_sequence[count++] = queue[i];
}

// Display head movement sequence
for (i = 0; i < count; i++) {
    printf("%d ", seek_sequence[i]);
}

// Calculate total head movement (seek time)
for (i = 0; i < count - 1; i++) {
    total += abs(seek_sequence[i + 1] - seek_sequence[i]);
}

avg = (float)total / (count - 1); // Average seek time

printf("\n\nTotal head movement: %d", total);
printf("\nAverage seek time: %.2f\n", avg);

return 0;
}

```

Enter the number of disk requests: 7

Enter the disk requests (track numbers): 82 170 43 140 24 16 190

Enter total disk size (e.g., 200): 200

Enter initial head position: 50

Head movement order:

50 82 140 170 190 199 43 24 16

Total head movement: 382

Average seek time: 47.75

explanation

Step 1: Sort all requests (and include head)

We include the head position in the list to make movement calculation easier.

So we first have:

[82, 170, 43, 140, 24, 16, 190, 50]

After sorting in ascending order:

[16, 24, 43, 50, 82, 140, 170, 190]

Now, our **head = 50** is at **index 3**.

Step 2: Determine direction — “away from spindle”

→ Means **towards higher track numbers** (right side).

So from position 50, we move **right first**, then **come back left** like an elevator.

Step 3: Move right (away from spindle)

We move from 50 → all higher numbers on the right side of 50:

Right side tracks = [82, 140, 170, 190]

So our movement goes like this:

50 → 82 → 140 → 170 → 190

Now the head is at **190** (the highest request).

But since the **disk size = 200**, the last track is **199** (the end of the disk).

So before turning back, the head goes till **199**:

... → 199

Step 4: Change direction (like elevator coming back down)

Now the head moves **back towards the lower end** to serve remaining tracks on the left side of 50:

Left side (in reverse order) = [43, 24, 16]

So the next sequence is:

199 → 43 → 24 → 16

Step 5: Combine full movement order

Putting it all together:

50 → 82 → 140 → 170 → 190 → 199 → 43 → 24 → 16

That's your **head movement order** 

Step 6: Calculate total head movement

Now, we calculate **seek distance** (absolute difference between consecutive positions):

Movement Calculation Distance

50 → 82	32
82 → 140	58
140 → 170	30
170 → 190	20
190 → 199	9
199 → 43	156
43 → 24	19
24 → 16	8

Total = 32 + 58 + 30 + 20 + 9 + 156 + 19 + 8 = 382

 **Total Head Movement = 382 tracks**

Step 7: Calculate Average Seek Time

There are **8 movements**, so:

Average Seek Time = $382 / 8 = 47.75$