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

// Function to sort the array
int i,j;
void sort(int arr[], int n) {
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

int main() {
    int n, head, disk_size, direction;
    int requests[100];
    int left[100], right[100];
    int l = 0, r = 0;
    int total = 0;

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

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

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

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

    printf("Enter head movement direction (1 for high, 0 for low): ");
    scanf("%d", &direction);

    // Separate requests into left and right of head
    for (i = 0; i < n; i++) {
        if (requests[i] < head)
            left[l++] = requests[i];
        else
            right[r++] = requests[i];
    }

    // Sort both sides
    sort(left, l);
    sort(right, r);

    printf("\nSCAN Disk Scheduling Algorithm:\n");
    printf("Initial head position: %d\n", head);
    printf("Seek Sequence: %d", head);

    if (direction == 1) {
        // Move towards higher end first
        for (i = 0; i < r; i++) {
            total += abs(head - right[i]);
            head = right[i];
            printf(" -> %d", head);
        }
```

```
        // At end, move to the max (end of disk)
        if (l > 0) {
            total += abs(head - (disk_size - 1));
            head = disk_size - 1;

            // Then move toward lower end
            total += abs(head - left[l - 1]);
            head = left[l - 1];
            printf(" -> %d", head);

            for (i = l - 2; i >= 0; i--) {
                total += abs(head - left[i]);
                head = left[i];
                printf(" -> %d", head);
            }
        }
    } else {
        // Move towards lower end first
        for (i = l - 1; i >= 0; i--) {
            total += abs(head - left[i]);
            head = left[i];
            printf(" -> %d", head);
        }

        // At beginning, move to 0
        if (r > 0) {
            total += abs(head - 0);
            head = 0;

            // Then move toward higher end
            total += abs(head - right[0]);
            head = right[0];
            printf(" -> %d", head);

            for (i = 1; i < r; i++) {
                total += abs(head - right[i]);
                head = right[i];
                printf(" -> %d", head);
            }
        }
    }

    printf("\nTotal head movement: %d\n", total);
    return 0;
}
```

```
OUTPUT :
Enter number of disk requests: 8
Enter disk requests: 55 58 39 18 90 160 150 184
Enter initial head position: 100
Enter total disk size: 200
Enter head movement direction (1 for high, 0 for low): 1

SCAN Disk Scheduling Algorithm:
Initial head position: 100
Seek Sequence: 100 -> 150 -> 160 -> 184 -> 90 -> 58 -> 55 -> 39 -> 18
Total head movement: 280
```