SCAN Disk Scheduling Algorithm Report

This report describes the implementation and execution of the SCAN Disk Scheduling Algorithm in C. The SCAN algorithm moves the disk arm towards one end of the disk, servicing requests in that direction, then reverses direction and services the remaining requests.

C Program Code:

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

#define SIZE 8
#define DISK_SIZE 200

void SCAN(int arr[], int head, char direction) {
    int seek_count = 0;
    int distance, cur_track;
    int left[SIZE], right[SIZE];
    int seek_sequence[SIZE * 2];
    int left_index = 0, right_index = 0, sequence_index = 0;

    if (direction == 'l') {
        left[left_index++] = 0;
    } else if (direction == 'r') {
        right[right_index++] = DISK_SIZE - 1;
    }

    for (int i = 0; i < SIZE; i++) {
        if (arr[i] < head) left[left_index++] = arr[i];
        if (arr[i] > head) right[right_index++] = arr[i];
    }

    for (int i = 0; i < left_index - 1; i++)
        for (int j = i + 1; j < left_index; j++)
            if (left[i] > left[j]) { int temp = left[i]; left[i] = left[j]; left[j] = temp; }

    for (int i = 0; i < right_index - 1; i++)
        for (int j = i + 1; j < right_index; j++)
            if (right[i] > right[j]) { int temp = right[i]; right[i] = right[j]; right[j] = temp; }

    int run = 2;
    while (run--) {
        if (direction == 'l') {
            for (int i = left_index - 1; i >= 0; i--) {
                cur_track = left[i];
                seek_sequence[sequence_index++] = cur_track;
                distance = abs(cur_track - head);
                seek_count += distance;
                head = cur_track;
            }
            direction = 'r';
        } else if (direction == 'r') {
            for (int i = 0; i < right_index; i++) {
                cur_track = right[i];
                seek_sequence[sequence_index++] = cur_track;
                distance = abs(cur_track - head);
                seek_count += distance;
                head = cur_track;
            }
            direction = 'l';
        }
    }

    printf("Total number of seek operations = %d\n", seek_count);
```

```
    printf("Seek Sequence is\n");
    for (int i = 0; i < sequence_index; i++) printf("%d\n", seek_sequence[i]);
}

int main() {
    int arr[SIZE] = {176, 79, 34, 60, 92, 11, 41, 114};
    int head = 50;
    char direction = 'l';
    SCAN(arr, head, direction);
    return 0;
}
```

Execution Example:

Input:

Requests: 176, 79, 34, 60, 92, 11, 41, 114

Initial Head Position: 50

Direction: Left

Output:

Total number of seek operations = 226

Seek Sequence:

41, 34, 11, 0, 60, 79, 92, 114, 176

Explanation:

1. The head moves left servicing all requests until it reaches track 0.
2. It then reverses direction and services requests on the right.
3. Total seek time is the sum of all head movements between consecutive requests.