NAME: G. GANESH

REG NO: 192373008

EXERICSE-38

Design a C program to simulate SCAN disk scheduling algorithm.

AIM:

To design a C program to simulate the SCAN (Elevator) disk scheduling algorithm.

Algorithm:

1. Input the number of disk requests, the disk size (range), and the initial head position.
2. Take the direction of head movement (towards higher or lower numbered tracks).
3. Sort the disk requests in ascending order.
4. Divide the requests into two parts:

- Requests less than the initial head position.
- Requests greater than or equal to the initial head position.

5. Process the requests in the direction of head movement, first servicing the requests in the current direction and then reversing to service the remaining requests.
6. Calculate the total seek time.
7. Output the seek sequence and total seek time.

Procedure:

1. Sort the requests to ensure proper traversal order.
2. Traverse in the specified direction, servicing requests until the end of the disk or the beginning.
3. Reverse the direction and process the remaining requests.
4. Sum the absolute differences between consecutive requests for total seek time.
5. Print the seek sequence and total seek time.

Code:

```
#include <stdio.h>

#include <stdlib.h>

void scan(int requests[], int n, int head, int diskSize, char direction) {

    int totalSeekTime = 0, current = head;

    int seekSequence[n + 2], index = 0;

    for (int i = 0; i < n - 1; i++) {

        for (int j = 0; j < n - i - 1; j++) {

            if (requests[j] > requests[j + 1]) {
```

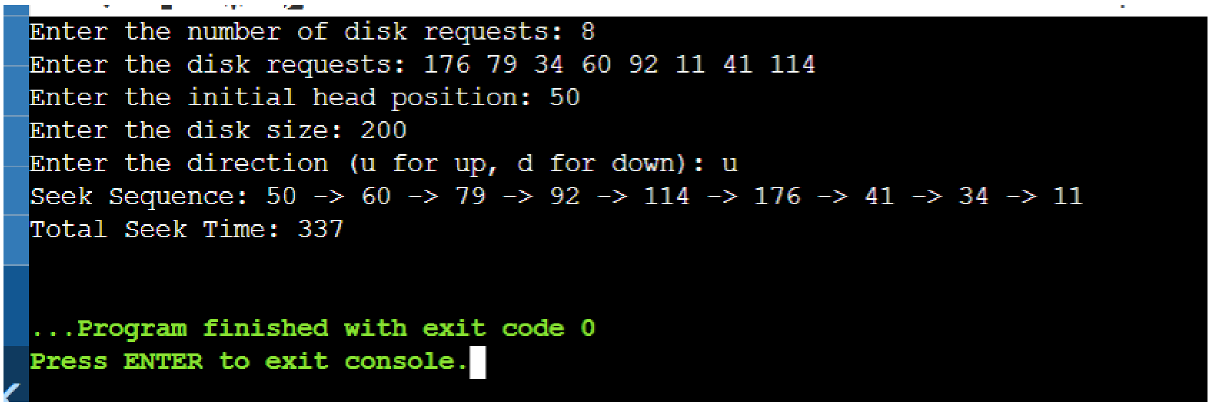
```
            int temp = requests[j];
            requests[j] = requests[j + 1];
            requests[j + 1] = temp;
        }
    }
}
int lower[n], higher[n], lowerCount = 0, higherCount = 0;
for (int i = 0; i < n; i++) {
    if (requests[i] < head) lower[lowerCount++] = requests[i];
    else higher[higherCount++] = requests[i];
}
if (direction == 'u') {  // Upwards
    for (int i = 0; i < higherCount; i++) {
        totalSeekTime += abs(current - higher[i]);
        seekSequence[index++] = higher[i];
        current = higher[i];
    }
    totalSeekTime += abs(current - (diskSize - 1));
    current = diskSize - 1;
    for (int i = lowerCount - 1; i >= 0; i--) {
        totalSeekTime += abs(current - lower[i]);
        seekSequence[index++] = lower[i];
        current = lower[i];
    }
} else {
    for (int i = lowerCount - 1; i >= 0; i--) {
        totalSeekTime += abs(current - lower[i]);
        seekSequence[index++] = lower[i];
        current = lower[i];
    }
```

```
        totalSeekTime += abs(current - 0);
        current = 0;
        for (int i = 0; i < higherCount; i++) {
            totalSeekTime += abs(current - higher[i]);
            seekSequence[index++] = higher[i];
            current = higher[i];
        }
    }
    printf("Seek Sequence: %d -> ", head);
    for (int i = 0; i < index; i++) {
        printf("%d", seekSequence[i]);
        if (i != index - 1) printf(" -> ");
    }
    printf("\nTotal Seek Time: %d\n", totalSeekTime);
}
int main() {
    int n, head, diskSize;
    char direction;
    printf("Enter the 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 the initial head position: ");
    scanf("%d", &head);
    printf("Enter the disk size: ");
    scanf("%d", &diskSize);
    printf("Enter the direction (u for up, d for down): ");
    scanf(" %c", &direction);

    scan(requests, n, head, diskSize, direction);

    return 0;

}
```

Result:

The program successfully simulates the SCAN disk scheduling algorithm, displaying the seek sequence and total seek time.

Output:



```
Enter the number of disk requests: 8
Enter the disk requests: 176 79 34 60 92 11 41 114
Enter the initial head position: 50
Enter the disk size: 200
Enter the direction (u for up, d for down): u
Seek Sequence: 50 -> 60 -> 79 -> 92 -> 114 -> 176 -> 41 -> 34 -> 11
Total Seek Time: 337


...Program finished with exit code 0
Press ENTER to exit console.
```