```
//TIA 19
//sujal jagtap
//B batch
//SSTF

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

#define MAX_REQUESTS 100

void sstf(int requests[], int n, int initial_head) {
    int total_seek_time = 0;
    int current_head = initial_head;

    // Create an array to keep track of visited requests
    int visited[MAX_REQUESTS] = {0};

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

        // Find the next request with the shortest seek time
        for (int j = 0; j < n; j++) {
            if (!visited[j]) {
                int seek = abs(current_head - requests[j]); // Fixed the abs() function call
                if (seek < min_seek) {
                    min_seek = seek;
                    next_request = j;
                }
            }
        }

        // Mark the selected request as visited
        visited[next_request] = 1;

        // Update the seek time and current head position
        total_seek_time += min_seek;
        printf("Move from %d to %d (Seek Time: %d)\n", current_head, requests[next_request],
min_seek);
        current_head = requests[next_request];
    }

    // Print the total seek time
    printf("Total Seek Time: %d\n", total_seek_time);
}

int main() {
    int n, initial_head;

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

```
    if (n <= 0 || n > MAX_REQUESTS) {
        printf("Invalid number of requests. \n");
        return 1;
    }

    int requests[MAX_REQUESTS];

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

    printf("Enter the track positions for %d requests:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &requests[i]);
    }

    sstf(requests, n, initial_head);

    return 0;
}
```

OUTPUT-:

```
student@student:~$ gcc OS_Lab12(SSTF).c
student@student:~$ ./a.out
Enter the number of requests: 7
Enter the initial head position: 50
Enter the track positions for 7 requests:
82
170
43
140
24
16
190
Move from 50 to 43 (Seek Time: 7)
Move from 43 to 24 (Seek Time: 19)
Move from 24 to 16 (Seek Time: 8)
Move from 16 to 82 (Seek Time: 66)
Move from 82 to 140 (Seek Time: 58)
Move from 140 to 170 (Seek Time: 30)
Move from 170 to 190 (Seek Time: 20)
Total Seek Time: 208
```

```
//DISK-SCAN
#include <iostream>
#include <cstdlib>
#include <algorithm> // For std::sort

#define MAX_REQUESTS 100

// Function to simulate the SCAN algorithm
void scan(int requests[], int n, int initial_head, int max_track) {
    int total_seek_time = 0;
    int current_head = initial_head;
    std::sort(requests, requests + n);
    // Move to the right first
    total_seek_time += abs(current_head - max_track);
    current_head = max_track;
    std::cout << "Move to right end: " << max_track << " (Seek Time: " << abs(initial_head -
max_track) << ")\n";

    // Process requests to the right
    for (int i = 0; i < n; i++) {
        if (requests[i] >= initial_head) {
            total_seek_time += abs(current_head - requests[i]);
            current_head = requests[i];
            std::cout << "Move to: " << current_head << " (Seek Time: " << abs(current_head -
requests[i]) << ")\n";
        }
    }
    // Move to the left end
    total_seek_time += abs(current_head - 0);
    current_head = 0;
    std::cout << "Move to left end: 0 (Seek Time: " << abs(current_head) << ")\n";

    // Process requests to the left
    for (int i = n - 1; i >= 0; i--) {
        if (requests[i] < initial_head) {
            total_seek_time += abs(current_head - requests[i]);
            current_head = requests[i];
            std::cout << "Move to: " << current_head << " (Seek Time: " << abs(current_head -
requests[i]) << ")\n";
        }
    }
    std::cout << "Total Seek Time: " << total_seek_time << "\n";
}

int main() {
    int n, initial_head, max_track;

    std::cout << "Enter number of requests: ";
    std::cin >> n;

    if (n <= 0 || n > MAX_REQUESTS) {
        std::cout << "Invalid number of requests.\n";
```

```
        return 1;
    }

    int requests[MAX_REQUESTS];

    std::cout << "Enter initial head position: ";
    std::cin >> initial_head;

    std::cout << "Enter maximum track position: ";
    std::cin >> max_track;

    std::cout << "Enter track positions:\n";
    for (int i = 0; i < n; i++) {
        std::cin >> requests[i];
        if (requests[i] < 0 || requests[i] > max_track) {
            std::cout << "Invalid request position.\n";
            return 1;
        }
    }

    scan(requests, n, initial_head, max_track);
    return 0;
}
student@student:~$ g++ diskscan.cpp
student@student:~$ ./a.out
Enter number of requests: 7
Enter initial head position: 50
Enter maximum track position: 199
Enter track positions:
82
170
43
140
24
16
190
Move to right end: 199 (Seek Time: 149)
Move to: 82 (Seek Time: 0)
Move to: 140 (Seek Time: 0)
Move to: 170 (Seek Time: 0)
Move to: 190 (Seek Time: 0)
Move to left end: 0 (Seek Time: 0)
Move to: 43 (Seek Time: 0)
Move to: 24 (Seek Time: 0)
Move to: 16 (Seek Time: 0)
Total Seek Time: 634
student@student:~$
```

```
//CLOOK
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

#define MAX_REQUESTS 100

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

// Function to simulate C-Look algorithm
void clook(int requests[], int n, int initial_head, int max_track) {
    int total_seek_time = 0;
    int current_head = initial_head;

    // Sort the requests in ascending order
    sort(requests, n);

    // Find the index of the initial head position
    int initial_index = 0;
    while (initial_index < n && requests[initial_index] <= initial_head) {
        initial_index++;
    }

    // Move the head to the right end first
    if (current_head < max_track) {
        total_seek_time += abs(current_head - max_track);
        current_head = max_track;
        printf("Move from %d to %d (Seek Time: %d)\n", initial_head, max_track, abs(initial_head -
max_track));
    }

    // Traverse the requests to the right
    for (int i = initial_index; i < n; i++) {
        int seek = abs(current_head - requests[i]);
        total_seek_time += seek;
        printf("Move from %d to %d (Seek Time: %d)\n", current_head, requests[i], seek);
        current_head = requests[i];
    }

    // Move the head to the left end
```

```
    if (current_head > requests[0]) {
        total_seek_time += abs(current_head - requests[0]);
        printf("Move from %d to %d (Seek Time: %d)\n", current_head, requests[0], abs(current_head
- requests[0]));
        current_head = requests[0];
    }

    // Traverse the requests to the left
    for (int i = 0; i < initial_index; i++) {
        int seek = abs(current_head - requests[i]);
        total_seek_time += seek;
        printf("Move from %d to %d (Seek Time: %d)\n", current_head, requests[i], seek);
        current_head = requests[i];
    }

    printf("Total Seek Time: %d\n", total_seek_time);
}

int main() {
    int n, initial_head, max_track;

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

    if (n <= 0 || n > MAX_REQUESTS) {
        printf("Invalid number of requests.\n");
        return 1;
    }

    int requests[MAX_REQUESTS];

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

    printf("Enter the maximum track position: ");
    scanf("%d", &max_track);

    printf("Enter the track positions for %d requests:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &requests[i]);
        if (requests[i] < 0 || requests[i] > max_track) {
            printf("Invalid request position.\n");
            return 1;
        }
    }

    clook(requests, n, initial_head, max_track);

    return 0;
}
student@student:~$ ./a.out
Enter the number of requests: 7
```

Enter the initial head position: 50
Enter the maximum track position: 199
Enter the track positions for 7 requests:
82
170
43
140
24
16
190
Move from 50 to 199 (Seek Time: 149)
Move from 199 to 82 (Seek Time: 117)
Move from 82 to 140 (Seek Time: 58)
Move from 140 to 170 (Seek Time: 30)
Move from 170 to 190 (Seek Time: 20)
Move from 190 to 16 (Seek Time: 174)
Move from 16 to 16 (Seek Time: 0)
Move from 16 to 24 (Seek Time: 8)
Move from 24 to 43 (Seek Time: 19)
Total Seek Time: 575