**Name: Atharva Salitri Division: CSAI B**

**Roll No.: 37 PRN: 12310120**

**Subject: OS Lab Assignment 9**

**Title: Implementation of Disk Scheduling**

**1) FCFS 2) SSTF 3) SCAN 4) C-SCAN**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define MAX\_REQUESTS 100

void bubble\_sort(int arr[], int n)

{

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

for (int 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;

}

}

void print\_moves(int start, int requests[], int n, bool visited[], int \*total\_head\_movement)

{

int current = start;

printf("%d", current); // Start position

for (int i = 0; i < n; i++)

{

if (visited[i])

continue; // Skip if already visited

\*total\_head\_movement += abs(current - requests[i]);

current = requests[i];

printf(" -> %d", current);

}

}

void fcfs(int requests[], int n, int start)

{

int total\_head\_movement = 0;

int current = start;

printf("FCFS Scheduling: %d", current); // Start position

for (int i = 0; i < n; i++)

{

total\_head\_movement += abs(current - requests[i]);

current = requests[i];

printf(" -> %d", current); // Append move

}

printf("\nTotal Head Movement: %d\n", total\_head\_movement);

}

void sstf(int requests[], int n, int start)

{

bool visited[MAX\_REQUESTS] = {false};

int total\_head\_movement = 0, current = start;

printf("SSTF Scheduling: %d", current); // Start position

for (int i = 0; i < n; i++)

{

int min\_distance = 1000, min\_index = -1;

for (int j = 0; j < n; j++)

{

if (!visited[j])

{

int distance = abs(current - requests[j]);

if (distance < min\_distance)

{

min\_distance = distance;

min\_index = j;

}

}

}

visited[min\_index] = true;

total\_head\_movement += min\_distance;

current = requests[min\_index];

printf(" -> %d", current);

}

printf("\nTotal Head Movement: %d\n", total\_head\_movement);

}

void scan(int requests[], int n, int start, int total\_tracks)

{

int total\_head\_movement = 0;

int sorted\_requests[MAX\_REQUESTS];

for (int i = 0; i < n; i++)

sorted\_requests[i] = requests[i];

bubble\_sort(sorted\_requests, n);

printf("SCAN Scheduling: %d", start); // Start position

int idx = 0;

while (idx < n && sorted\_requests[idx] < start)

idx++;

for (int i = idx; i < n; i++)

{

total\_head\_movement += abs(start - sorted\_requests[i]);

start = sorted\_requests[i];

printf(" -> %d", start);

}

if (start != total\_tracks - 1)

{

total\_head\_movement += abs(start - (total\_tracks - 1));

start = total\_tracks - 1;

printf(" -> %d", start);

}

for (int i = idx - 1; i >= 0; i--)

{

total\_head\_movement += abs(start - sorted\_requests[i]);

start = sorted\_requests[i];

printf(" -> %d", start);

}

printf("\nTotal Head Movement: %d\n", total\_head\_movement);

}

void cscan(int requests[], int n, int start, int total\_tracks)

{

int total\_head\_movement = 0;

int sorted\_requests[MAX\_REQUESTS];

for (int i = 0; i < n; i++)

sorted\_requests[i] = requests[i];

bubble\_sort(sorted\_requests, n);

printf("C-SCAN Scheduling: %d", start); // Start position

int idx = 0;

while (idx < n && sorted\_requests[idx] < start)

idx++;

for (int i = idx; i < n; i++)

{

total\_head\_movement += abs(start - sorted\_requests[i]);

start = sorted\_requests[i];

printf(" -> %d", start);

}

if (start != total\_tracks - 1)

{

total\_head\_movement += abs(start - (total\_tracks - 1));

start = 0;

total\_head\_movement += (total\_tracks - 1);

printf(" -> %d", start);

}

for (int i = 0; i < idx; i++)

{

total\_head\_movement += abs(start - sorted\_requests[i]);

start = sorted\_requests[i];

printf(" -> %d", start);

}

printf("\nTotal Head Movement: %d\n", total\_head\_movement);

}

int main()

{

int requests[MAX\_REQUESTS], n, start, total\_tracks;

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

scanf("%d", &n);

printf("Enter the requests: ");

for (int i = 0; i < n; i++)

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

printf("Enter the starting position of the head: ");

scanf("%d", &start);

printf("Enter the total number of tracks: ");

scanf("%d", &total\_tracks);

printf("\n");

fcfs(requests, n, start);

printf("\n");

sstf(requests, n, start);

printf("\n");

scan(requests, n, start, total\_tracks);

printf("\n");

cscan(requests, n, start, total\_tracks);

return 0;

}

**Output:**

