#include <stdio.h>

#include <stdlib.h>

#define MAX 100

// Function prototypes

void sstf(int requests[], int n, int head);

void scan(int requests[], int n, int head, int direction);

void clook(int requests[], int n, int head);

// Function to compare two integers (used in qsort)

int compare(const void \*a, const void \*b) {

return (\*(int\*)a - \*(int\*)b);

}

int main() {

int requests[MAX], n, head, i, direction;

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

scanf("%d", &n);

printf("Enter the disk requests:\n");

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

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

}

printf("Enter the initial head position: ");

scanf("%d", &head);

printf("Enter the direction of head movement (1 for right, 0 for left): ");

scanf("%d", &direction);

printf("\n--- SSTF Scheduling ---\n");

sstf(requests, n, head);

printf("\n--- SCAN Scheduling ---\n");

scan(requests, n, head, direction);

printf("\n--- C-LOOK Scheduling ---\n");

clook(requests, n, head);

return 0;

}

// Shortest Seek Time First (SSTF) algorithm

void sstf(int requests[], int n, int head) {

int visited[MAX] = {0};

int total\_seek\_time = 0, current\_head = head;

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

int min\_distance = 9999, index = -1;

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

if (!visited[j]) {

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

if (distance < min\_distance) {

min\_distance = distance;

index = j;

}

}

}

if (index != -1) {

visited[index] = 1;

total\_seek\_time += min\_distance;

current\_head = requests[index];

printf("Served Request: %d, Current Head Position: %d\n", requests[index], current\_head);

}

}

printf("Total Seek Time for SSTF: %d\n", total\_seek\_time);

}

// SCAN algorithm

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

int total\_seek\_time = 0, current\_head = head;

int sorted\_requests[MAX];

// Sort the requests

qsort(requests, n, sizeof(int), compare);

int start = 0, end = n - 1;

// Determine the next head movement

if (direction == 1) { // Moving right

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

if (requests[i] >= current\_head) {

start = i;

break;

}

}

// Serve requests to the right

for (int i = start; i < n; i++) {

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

current\_head = requests[i];

printf("Served Request: %d, Current Head Position: %d\n", requests[i], current\_head);

}

// Move to the end of the disk

total\_seek\_time += abs(current\_head - 199); // Assuming max cylinder is 199

current\_head = 199;

// Serve requests to the left

for (int i = n - 1; i >= start; i--) {

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

current\_head = requests[i];

printf("Served Request: %d, Current Head Position: %d\n", requests[i], current\_head);

}

} else { // Moving left

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

if (requests[i] <= current\_head) {

end = i;

break;

}

}

// Serve requests to the left

for (int i = end; i >= 0; i--) {

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

current\_head = requests[i];

printf("Served Request: %d, Current Head Position: %d\n", requests[i], current\_head);

}

// Move to the beginning of the disk

total\_seek\_time += abs(current\_head - 0);

current\_head = 0;

// Serve requests to the right

for (int i = 0; i <= end; i++) {

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

current\_head = requests[i];

printf("Served Request: %d, Current Head Position: %d\n", requests[i], current\_head);

}

}

printf("Total Seek Time for SCAN: %d\n", total\_seek\_time);

}

// C-LOOK algorithm

void clook(int requests[], int n, int head) {

int total\_seek\_time = 0, current\_head = head;

int sorted\_requests[MAX];

// Sort the requests

qsort(requests, n, sizeof(int), compare);

// Serve requests greater than or equal to head first

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

if (requests[i] >= current\_head) {

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

current\_head = requests[i];

printf("Served Request: %d, Current Head Position: %d\n", requests[i], current\_head);

}

}

// After reaching the end, wrap around to the smallest request

if (current\_head < 199) { // Assuming max cylinder is 199

total\_seek\_time += abs(current\_head - requests[0]);

current\_head = requests[0];

printf("Served Request: %d, Current Head Position: %d\n", requests[0], current\_head);

}

// Serve remaining requests

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

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

current\_head = requests[i];

printf("Served Request: %d, Current Head Position: %d\n", requests[i], current\_head);

}

printf("Total Seek Time for C-LOOK: %d\n", total\_seek\_time);

}



