PRACTICAL NO: 08

Name: Riyan Shaikh

Roll no: T512051 [B]

CODE:

#include <stdio.h>

#include <stdlib.h>

#include <limits.h> // <-- Include this for INT\_MAX

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

void SCAN(int requests[], int n, int head, int disk\_size);

void C\_Look(int requests[], int n, int head);

int main() {

int n, head, disk\_size, choice;

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

scanf("%d", &n);

int \*requests = (int \*)malloc(n \* sizeof(int));

if (requests == NULL) {

printf("Memory allocation failed!\n");

return -1;

}

printf("Enter the request queue: ");

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 (total number of cylinders): ");

scanf("%d", &disk\_size);

printf("\nChoose Disk Scheduling Algorithm:\n");

printf("1. SSTF\n2. SCAN\n3. C-Look\n");

scanf("%d", &choice);

switch(choice) {

case 1:

SSTF(requests, n, head);

break;

case 2:

SCAN(requests, n, head, disk\_size);

break;

case 3:

C\_Look(requests, n, head);

break;

default:

printf("Invalid choice\n");

}

free(requests);

return 0;

}

int findClosest(int requests[], int n, int head, int visited[]) {

int minDistance = INT\_MAX; // Use INT\_MAX from limits.h

int index = -1;

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

if (!visited[i] && abs(requests[i] - head) < minDistance) {

minDistance = abs(requests[i] - head);

index = i;

}

}

return index;

}

// SSTF Disk Scheduling

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

int visited[n], totalSeekTime = 0;

for (int i = 0; i < n; i++) visited[i] = 0;

printf("\nSSTF Disk Scheduling:\n");

printf("Seek sequence: %d", head);

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

int closestIndex = findClosest(requests, n, head, visited);

if (closestIndex == -1) break;

visited[closestIndex] = 1;

totalSeekTime += abs(requests[closestIndex] - head);

head = requests[closestIndex];

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

}

printf("\nTotal Seek Time (SSTF): %d\n", totalSeekTime);

}

// SCAN Disk Scheduling

void SCAN(int requests[], int n, int head, int disk\_size) {

int totalSeekTime = 0, i;

requests = (int \*)realloc(requests, (n + 2) \* sizeof(int));

requests[n] = 0;

requests[n+1] = disk\_size - 1;

n += 2;

for (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 headIndex = 0;

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

if (requests[i] >= head) {

headIndex = i;

break;

}

}

printf("\nSCAN Disk Scheduling:\n");

printf("Seek sequence: %d", head);

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

totalSeekTime += abs(requests[i] - head);

head = requests[i];

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

}

for (i = headIndex - 1; i >= 0; i--) {

totalSeekTime += abs(requests[i] - head);

head = requests[i];

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

}

printf("\nTotal Seek Time (SCAN): %d\n", totalSeekTime);

}

// C-Look Disk Scheduling

void C\_Look(int requests[], int n, int head) {

int totalSeekTime = 0, i;

for (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 headIndex = 0;

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

if (requests[i] >= head) {

headIndex = i;

break;

}

}

printf("\nC-Look Disk Scheduling:\n");

printf("Seek sequence: %d", head);

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

totalSeekTime += abs(requests[i] - head);

head = requests[i];

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

}

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

totalSeekTime += abs(requests[i] - head);

head = requests[i];

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

}

printf("\nTotal Seek Time (C-Look): %d\n", totalSeekTime);

}

OUTPUT:

