# Bansilal Ramnath Agarwal Charitable Trust’s

Vishwakarma Institute of Technology, Pune-37

*(Anautonomous Institute of Savitribai Phule Pune University)*



**Department of Computer Engineering**

|  |  |
| --- | --- |
| **Division** | **CS** |
| **Batch** | **B1** |
| **Roll no.** | **90** |
| **Name** | **Aditya Shrinivas Kurapati** |
| **PRN No** | **12320184** |

* Implementing FCFS, SCAN, C-SCAN, SSTF:-

#include <limits.h>

#include <stdbool.h>

#include <stdio.h>

#include <stdlib.h>

// Function to simulate FCFS disk scheduling

void fcfs(int requests[], int num\_requests, int head) {

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

printf("Head movement order: %d", head);

for (int i = 0; i < num\_requests; ++i) {

printf(" -> %d", requests[i]);

}

printf("\n");

}

// Function to simulate SCAN disk scheduling

void scan(int requests[], int num\_requests, int head, int max\_cylinder) {

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

printf("Head movement order: ");

bool direction = true; // true for moving right, false for moving left

int current = head;

while (true) {

printf("%d ", current);

if (direction) {

bool found = false;

for (int i = 0; i < num\_requests; ++i) {

if (requests[i] == current) {

printf("-> %d ", requests[i]);

found = true;

break;

}

}

if (found) {

direction = false;

}

if (current == max\_cylinder) {

direction = false;

}

current++;

} else {

bool found = false;

for (int i = 0; i < num\_requests; ++i) {

if (requests[i] == current) {

printf("-> %d ", requests[i]);

found = true;

break;

}

}

if (found) {

direction = true;

}

if (current == 0) {

direction = true;

}

current--;

}

if (current > max\_cylinder) {

current = max\_cylinder;

}

if (current < 0) {

current = 0;

}

if (current == head) {

break;

}

}

printf("\n");

}

// Function to simulate C-SCAN disk scheduling

void cscan(int requests[], int num\_requests, int head, int max\_cylinder) {

printf("C-SCAN Disk Scheduling:\n");

printf("Head movement order: ");

int current = head;

printf("%d ", current);

bool direction = true; // true for moving right, false for moving left

while (true) {

printf("-> ");

if (direction) {

bool found = false;

for (int i = 0; i < num\_requests; ++i) {

if (requests[i] == current) {

printf("%d ", requests[i]);

found = true;

break;

}

}

if (found) {

current++;

}

if (current == max\_cylinder) {

printf("%d ", max\_cylinder);

current = 0;

}

if (current > max\_cylinder) {

current = 0;

}

} else {

bool found = false;

for (int i = num\_requests - 1; i >= 0; --i) {

if (requests[i] == current) {

printf("%d ", requests[i]);

found = true;

break;

}

}

if (found) {

current--;

}

if (current == 0) {

printf("0 ");

current = max\_cylinder;

}

if (current < 0) {

current = max\_cylinder;

}

}

if (current == head) {

break;

}

}

printf("\n");

}

// Function to simulate SSTF disk scheduling

void sstf(int requests[], int num\_requests, int head) {

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

printf("Head movement order: ");

int current = head;

bool processed[num\_requests];

for (int i = 0; i < num\_requests; ++i) {

processed[i] = false;

}

for (int i = 0; i < num\_requests; ++i) {

int min\_distance = INT\_MAX;

int next\_index = -1;

for (int j = 0; j < num\_requests; ++j) {

if (!processed[j]) {

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

if (distance < min\_distance) {

min\_distance = distance;

next\_index = j;

}

}

}

printf("%d ", current);

current = requests[next\_index];

processed[next\_index] = true;

}

printf("\n");

}

int main() {

int requests[] = {98, 183, 37, 122, 14, 124, 65, 67}; // Sample disk requests

int num\_requests = sizeof(requests) / sizeof(requests[0]);

int head = 53; // Sample initial head position

int max\_cylinder = 199; // Sample maximum cylinder

fcfs(requests, num\_requests, head);

scan(requests, num\_requests, head, max\_cylinder);

cscan(requests, num\_requests, head, max\_cylinder);

sstf(requests, num\_requests, head);

return 0;

}

OUTPUT:-   
