\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Operating System Lab

CEN-493

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Program - 15

Code :-

#include <iostream>

#include <math.h>

#include <vector>

#include <algorithm>

using namespace std;

void printLines()

{

    for (int i = 0; i < 120; i++)

    {

        cout << "-";

    }

    cout << "\n";

}

void printTheInfo(string info, int noOfDiskTracks, vector<int> trackMovement, vector<int> headMovement)

{

    printLines();

    cout << info << "\n";

    printLines();

    int totalTrackMovement = 0;

    cout << "\nHead Movement\n";

    for (int i = 0; i < headMovement.size(); i++)

    {

        if (headMovement.size() - 1 == i)

            cout << headMovement[i] << " ";

        else

            cout << headMovement[i] << " -> ";

    }

    cout << "\n";

    cout << "\nTrack Movement\n";

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

    {

        totalTrackMovement += trackMovement[i];

        if (i == noOfDiskTracks - 1)

            cout << trackMovement[i];

        else

            cout << trackMovement[i] << " + ";

    }

    cout << " = " << totalTrackMovement << "\n";

    float avgHeadMovement = (totalTrackMovement / (float)noOfDiskTracks);

    cout << "\nAverage Head Movement : \n";

    cout << avgHeadMovement << "\n\n";

}

void fcfsDiskScheduling(int noOfDiskTracks, vector<int> diskTracks, int headPosition)

{

    vector<int> headMovement, trackMovement;

    int prevHeadPosition = headPosition;

    headMovement.push\_back(prevHeadPosition);

    for (int track = 0; track < noOfDiskTracks; track++)

    {

        headMovement.push\_back(diskTracks[track]);

        trackMovement.push\_back(abs(diskTracks[track] - prevHeadPosition));

        prevHeadPosition = diskTracks[track];

    }

    printTheInfo("Fcfs Disk Scheduling Algorithm", noOfDiskTracks, trackMovement, headMovement);

}

void sstfDiskScheduling(int noOfDiskTracks, vector<int> diskTracks, int headPosition)

{

    vector<int> headMovement, trackMovement;

    int prevHeadPosition = headPosition;

    headMovement.push\_back(prevHeadPosition);

    while (!diskTracks.empty())

    {

        int shortestSeekTime = 1e9, shortestSeekTimeIndex = 0;

        for (int i = 0; i < diskTracks.size(); i++)

        {

            if (shortestSeekTime > abs(diskTracks[i] - prevHeadPosition))

            {

                shortestSeekTime = abs(diskTracks[i] - prevHeadPosition);

                shortestSeekTimeIndex = i;

            }

        }

        headMovement.push\_back(diskTracks[shortestSeekTimeIndex]);

        trackMovement.push\_back(abs(diskTracks[shortestSeekTimeIndex] - prevHeadPosition));

        prevHeadPosition = diskTracks[shortestSeekTimeIndex];

        diskTracks.erase(diskTracks.begin() + shortestSeekTimeIndex);

    }

    printTheInfo("Sstf Disk Scheduling Algorithm", noOfDiskTracks, trackMovement, headMovement);

}

void scanDiskScheduling(int noOfDiskTracks, vector<int> diskTracks, int headPosition)

{

    vector<int> headMovement, trackMovement;

    int prevHeadPosition = headPosition;

    headMovement.push\_back(prevHeadPosition);

    sort(diskTracks.begin(), diskTracks.end());

    int strtTrack = lower\_bound(diskTracks.begin(), diskTracks.end(), prevHeadPosition) - diskTracks.begin();

    if (diskTracks[strtTrack] > prevHeadPosition)

        strtTrack--;

    for (int track = strtTrack; track >= 0; track--)

    {

        headMovement.push\_back(diskTracks[track]);

        trackMovement.push\_back(abs(diskTracks[track] - prevHeadPosition));

        prevHeadPosition = diskTracks[track];

    }

    for (int track = strtTrack + 1; track < noOfDiskTracks; track++)

    {

        headMovement.push\_back(diskTracks[track]);

        trackMovement.push\_back(abs(diskTracks[track] - prevHeadPosition));

        prevHeadPosition = diskTracks[track];

    }

    printTheInfo("Scan (Elevator) Disk Scheduling Algorithm", noOfDiskTracks, trackMovement, headMovement);

}

int main()

{

    system("cls");

    printLines();

    cout << "\_\_\_VickyGupta\_20BCS070\_\_\_\n";

    printLines();

    cout << "Disk Scheduling Alogrithms\n";

    printLines();

    int noOfDiskTracks;

    cout << "Enter The No Of Disk Tracks : \n";

    cin >> noOfDiskTracks;

    vector<int> diskTrack(noOfDiskTracks);

    cout << "\nEnter The  Disk Tracks :\n";

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

    {

        cin >> diskTrack[i];

    }

    int headPosition;

    cout << "\nEnter The Head Position : ";

    cin >> headPosition;

    printLines();

    printLines();

    fcfsDiskScheduling(noOfDiskTracks, diskTrack, headPosition);

    printLines();

    printLines();

    sstfDiskScheduling(noOfDiskTracks, diskTrack, headPosition);

    printLines();

    printLines();

    scanDiskScheduling(noOfDiskTracks, diskTrack, headPosition);

    printLines();

    printLines();

    return 0;

}

Output :-

Text

Description automatically generated

Text

Description automatically generated