Sidharth

2K18/MC/114

Experiment 4

Aim: Write a program to implement following Disk Scheduling Algorithms.

Algorithm:

FCFS

Elevator's Algorithm

Input - Head Start Position - 101, Total number of cylinders = 300

1. Queue 1 - 22, 43, 55, 21, 220, 2, 33, 55, 221, 231,121, 157, 189
2. Queue 2 - 31, 157, 242, 134, 79, 145, 178, 244, 267, 288

Generate the output as

Number of cylinders traversed for FCFS and Elevators algorithm for both the input queues

The order of cylinders traversed for FCFS and Elevators algorithm for both the input queues

Code:

```
#include <bits/stdc++.h>
using namespace std;

int disk_size = 300;

void FCFS(int arr[], int head, int size)
{
    cout<<"Queue: ";
    for(int i=0; i<size; i++){
        cout<<arr[i]<<"-> ";
    }
```

```
    cout<<"\n";

    int seek_count = 0;
    int distance, cur_track;

    cout<<"Seek Operations: ";
    for (int i=0; i<size; i++){
        cur_track = arr[i];
        distance = abs(cur_track - head);
        cout<<distance<<" ";
        seek_count += distance;
        head = cur_track;
    }

    cout<<"\nTotal number of seek operations = "<<seek_count<<endl;
    cout<<"\n";
}

void SCAN(int arr[], int head, string direction, int size)
{
    int seek_count = 0;
    int distance, cur_track;
    vector<int> left, right;
    vector<int> seek_sequence;

    if (direction == "left")
        left.push_back(0);
    else if (direction == "right")
        right.push_back(disk_size - 1);
    for (int i = 0; i < size; i++) {
        if (arr[i] < head)
        left.push_back(arr[i]);
        if (arr[i] > head)
        right.push_back(arr[i]);
    }

    sort(left.begin(), left.end());
    sort(right.begin(), right.end());

    int run = 2;
    while (run--) {
    if (direction == "left") {
        for (int i = left.size() - 1; i >= 0; i--) {
            cur_track = left[i];
            // appending current track to seek sequence
```

```
            seek_sequence.push_back(cur_track);
            // calculate absolute distance
            distance = abs(cur_track - head);
            // increase the total count
            seek_count += distance;
            // accessed track is now the new head
            head = cur_track;
        }
        direction = "right";
    }
    else if (direction == "right") {
        for (int i = 0; i < right.size(); i++) {
        cur_track = right[i];
        // appending current track to seek sequence
        seek_sequence.push_back(cur_track);
        // calculate absolute distance
        distance = abs(cur_track - head);
        // increase the total count
        seek_count += distance;
        // accessed track is now new head
        head = cur_track;
        }
    direction = "left";
    }
    }
    cout<<"\nQueue: ";
    for(int i=0; i<size; i++) cout<<arr[i]<<"-> ";
    cout <<"\nSeek Sequence: ";
    for(int i=0; i<seek_sequence.size(); i++){
    cout <<seek_sequence[i]<<" ";
    }
    cout <<"\nTotal number of seek operations = "<<seek_count<< endl;
}


int main()
{
    int arr1[] = {22, 43, 55, 21, 220, 2, 33, 55, 221, 231,121, 157, 189 };
    int arr2[] = {31, 157, 242, 134, 79, 145, 178, 244, 267, 288 };
    int head = 101;

    int size1 = sizeof(arr1)/sizeof(arr1[0]);
    int size2 = sizeof(arr2)/sizeof(arr2[0]);

    cout<<"\nFirst Come First Serve Algo\n";
```

```
    FCFS(arr1, head, size1);
    FCFS(arr2, head, size2);

    string direction;
    cout<<"\nElevator Algo\nEnter direction: ";
    cin>>direction;

    SCAN(arr1, head, direction, size1);
    SCAN(arr2, head, direction, size2);

    return 0;
}
```

Output: