Operating System Practicals

Assignment 7

Krunal Rank
U18CO081

1. To implement Shortest Seek Time First (SSTF) Disk Scheduling Algorithm.
2. To implement SCAN algorithm for Disk Scheduling.

```
#include <bits/stdc++.h>
#define N 10
#define MAX_TRACKS 200
using namespace std;

int get_next_track(vector<int>&tracks,int head,vector<int> &mask){
   int idx = -1;
   int seek_time = MAX_TRACKS + 2;
   for(int i = 0;i<tracks.size();i++){
       if(mask[i]) continue;
       if(abs(head-tracks[i])<seek_time){
           idx = i;
           seek_time = abs(head-tracks[i]);
       }
   }
   return idx;
}

void sstf(vector<int> tracks,int initial_head){

cout<<"----------------------------------------------------------------------"<<endl;
   cout<<"Shortest Seek Time First Algorithm"<<endl;
   int seek_time = 0;
   vector<int> mask(tracks.size(),0);
   int completed = 0;
   while(completed<tracks.size()){
       int next_track = get_next_track(tracks,initial_head,mask);
       seek_time += abs(tracks[next_track]-initial_head);
       cout<<"Seeked Track "<<next_track<<" at Position: "<<tracks[next_track]<<endl;
       cout<<"Seek Time: "<<abs(tracks[next_track]-initial_head)<<endl;
       initial_head = tracks[next_track];
       mask[next_track] = 1;
       completed++;
   }
   cout<<"Total Seek Time: "<<seek_time<<endl;

}
```

```
void scan(vector<int> tracks,int initial_head){

cout<<"--------------------------------------------------------------------------------"<<endl;
   cout<<"Scan Algorithm"<<endl;
   tracks.push_back(0);
   tracks.push_back(MAX_TRACKS-1);
   sort(tracks.begin(),tracks.end());
   int seek_time = 0;
   int idx = lower_bound(tracks.begin(),tracks.end(),initial_head + 1) -
tracks.begin();
   idx--;
   for(int i = idx;i>=0;i--){
       cout<<"Seeked Track "<<i<<" at Position: "<<tracks[i]<<endl;
       cout<<"Seek Time: "<<abs(tracks[i]-initial_head)<<endl;
       seek_time += abs(tracks[i]-initial_head);
       initial_head = tracks[i];
   }
   for(int i = idx+1;i<tracks.size();i++){
       cout<<"Seeked Track "<<i<<" at Position: "<<tracks[i]<<endl;
       cout<<"Seek Time: "<<abs(tracks[i]-initial_head)<<endl;
       seek_time += abs(tracks[i]-initial_head);
       initial_head = tracks[i];
   }
   cout<<"Total Seek Time: "<<seek_time<<endl;
}

int main(){
   srand(time(NULL));
   vector<int> tracks;
   for(int i = 0;i<N;i++){
       tracks.push_back(rand()%MAX_TRACKS);
   }
   vector<int> displayTracks = tracks;
   sort(displayTracks.begin(),displayTracks.end());
   cout<<"Tracks: ";
   for(auto i: displayTracks) cout<<i<<" ";
   cout<<endl;
   sstf(tracks,100);
   scan(tracks,100);
}
```

Output:

```
Tracks: 2 11 46 58 67 75 81 148 169 191
-----------------------------------------------------------------------
Shortest Seek Time First Algorithm
Seeked Track 1 at Position: 81
Seek Time: 19
Seeked Track 0 at Position: 75
Seek Time: 6
Seeked Track 8 at Position: 67
Seek Time: 8
Seeked Track 9 at Position: 58
Seek Time: 9
Seeked Track 3 at Position: 46
Seek Time: 12
Seeked Track 2 at Position: 11
Seek Time: 35
Seeked Track 6 at Position: 2
Seek Time: 9
Seeked Track 4 at Position: 148
Seek Time: 146
Seeked Track 5 at Position: 169
Seek Time: 21
Seeked Track 7 at Position: 191
Seek Time: 22
Total Seek Time: 287
-----------------------------------------------------------------------
Scan Algorithm
Seeked Track 7 at Position: 81
Seek Time: 19
Seeked Track 6 at Position: 75
Seek Time: 6
Seeked Track 5 at Position: 67
Seek Time: 8
Seeked Track 4 at Position: 58
Seek Time: 9
Seeked Track 3 at Position: 46
Seek Time: 12
Seeked Track 2 at Position: 11
Seek Time: 35
Seeked Track 1 at Position: 2
Seek Time: 9
Seeked Track 0 at Position: 0
Seek Time: 2
```

```
Seeked Track 8 at Position: 148
Seek Time: 148
Seeked Track 9 at Position: 169
Seek Time: 21
Seeked Track 10 at Position: 191
Seek Time: 22
Seeked Track 11 at Position: 199
Seek Time: 8
Total Seek Time: 299
```