**Program 14:**

1. Write a program to implement the SCAN elevator disk scheduling algorithm. The program should give detail about each disk movement from starting head position (input from the user) and calculate average head movement.

# Write a program to implement the LOOK elevator disk scheduling algorithm. The program should give detail about each disk movement from starting head position (input from the user) and calculate average head movement.

# Answer:

# Source code:

# #include <iostream>

# #include <vector>

# #include <cmath>

# #include <algorithm>

# using namespace std;

# int search\_(vector<int> programs,int lo, int hi, int x){// find point where arr[mid]<=head && arr[mid+1]>head

# if(lo<hi){

# int mid=(hi-lo)/2;

# if(programs[mid]==x){

# return mid;

# }else if(programs[mid]<x){

# if((mid+1)==(int)programs.size() || programs[mid+1]>x) return mid;

# else search\_(programs,lo,mid-1,x);

# }else{

# if((mid)==0 || programs[mid-1]<x) return mid;

# else search\_(programs,mid+1,hi,x);

# }

# }

# return -1;

# }

# int piv(vector<int> &arr, int lo, int hi){

# int i=lo, p=arr[hi];

# for (int j=lo; j<=hi;j++){

# if(arr[j]<p){

# swap(arr[j],arr[i]);

# i+=1;

# }

# }

# swap(arr[hi],arr[i]);

# return i;

# }

# void sort\_(vector<int> &arr,int lo, int hi){

# if(lo<hi){

# int p=piv(arr,lo,hi);

# sort\_(arr,lo,p-1);

# sort\_(arr,p+1,hi);

# }

# }

# int left\_move(vector<int> programs,int ind,int pos){

# int sum=0;

# for(int i=ind;i>=0;i--){

# sum+=abs(programs[i]-pos);

# cout<<pos<<"\t"<<programs[i]<<"\t"<<abs(programs[i]-pos)<<endl;

# pos=programs[i];

# }

# return sum;

# }

# int right\_move(vector<int> programs, int ind, int pos){

# int sum=0;

# for(int i=ind+1;i<programs.size();i++){

# sum+=abs(programs[i]-pos);

# cout<<pos<<"\t"<<programs[i]<<"\t"<<abs(programs[i]-pos)<<endl;

# pos=programs[i];

# }

# return sum;

# }

# int algo(vector<int> programs,int pos,int dir,int disk){

# int sum=0,diff, n=(int)programs.size();

# cout<<"Disk Movement:-"<<endl;

# cout<<"From\tto\tDisk Movement"<<endl;

# sort\_(programs,0,n-1); //sorting

# int ind=search\_(programs,0,n-1,pos); //searching nearest index(0 based)

# if(programs[ind]>pos) ind-=1;

# if(dir==0){

# // for left side

# sum+=left\_move(programs,ind, pos);

# //to zero

# sum+=programs[0];

# cout<<programs[0]<<"\t"<<0<<"\t"<<programs[0]<<endl;

# //for right

# sum+=right\_move(programs,ind,0);

# }else{

# // for left side

# sum+=right\_move(programs,ind, pos);

# //to end

# sum+=(disk-1-programs[n-1]);

# cout<<programs[n-1]<<"\t"<<disk-1<<"\t"<<disk-1-programs[n-1]<<endl;

# //for right

# sum+=left\_move(programs,ind,disk-1);

# }

# return sum;

# }

# int main()

# {

# int n,pos,dir,disk;

# cout << "Enter number of programs\tInitial position of Head\tTotal number of disks"<<endl;

# cin>>n>>pos>>disk;

# cout << "Enter direction of head movement \*\*1 = Right and 0 = Left\*\*"<<endl;

# cin>>dir;

# vector<int> programs(n);

# cout<<"Enter programs"<<endl;

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

# cin>>programs[i];

# }

# int total\_movements=algo(programs,pos,dir,disk);

# total\_movements=(float)total\_movements;

# cout<<"Average disk movement: "<<total\_movements/(float)n<<endl;

# return 0;

# }

# Output:

# 

# 

# Source code:

# #include <iostream>

# #include <vector>

# #include <cmath>

# #include <algorithm>

# using namespace std;

# int search\_(vector<int> programs,int lo, int hi, int x){// find point where arr[mid]<=head && arr[mid+1]>head

# if(lo<hi){

# int mid=(hi-lo)/2;

# if(programs[mid]==x){

# return mid;

# }else if(programs[mid]<x){

# if((mid+1)==(int)programs.size() || programs[mid+1]>x) return mid;

# else search\_(programs,lo,mid-1,x);

# }else{

# if((mid)==0 || programs[mid-1]<x) return mid;

# else search\_(programs,mid+1,hi,x);

# }

# }

# return -1;

# }

# int piv(vector<int> &arr, int lo, int hi){

# int i=lo, p=arr[hi];

# for (int j=lo; j<=hi;j++){

# if(arr[j]<p){

# swap(arr[j],arr[i]);

# i+=1;

# }

# }

# swap(arr[hi],arr[i]);

# return i;

# }

# void sort\_(vector<int> &arr,int lo, int hi){

# if(lo<hi){

# int p=piv(arr,lo,hi);

# sort\_(arr,lo,p-1);

# sort\_(arr,p+1,hi);

# }

# }

# int left\_move(vector<int> programs,int ind,int pos){

# int sum=0;

# for(int i=ind;i>=0;i--){

# sum+=abs(programs[i]-pos);

# cout<<pos<<"\t"<<programs[i]<<"\t"<<abs(programs[i]-pos)<<endl;

# pos=programs[i];

# }

# return sum;

# }

# int right\_move(vector<int> programs, int ind, int pos){

# int sum=0;

# for(int i=ind+1;i<programs.size();i++){

# sum+=abs(programs[i]-pos);

# cout<<pos<<"\t"<<programs[i]<<"\t"<<abs(programs[i]-pos)<<endl;

# pos=programs[i];

# }

# return sum;

# }

# int algo(vector<int> programs,int pos, int dir){

# int sum=0,diff, n=(int)programs.size();

# cout<<"Disk Movement:-"<<endl;

# cout<<"From\tto\tDisk Movement"<<endl;

# sort\_(programs,0,n-1); //sorting

# int ind=search\_(programs,0,n-1,pos); //searching nearest index(0 based)

# if(programs[ind]>pos) ind-=1;

# if(dir==0){

# // for left side

# sum += left\_move(programs,ind,pos);

# //for right

# sum+= right\_move(programs,ind,programs[0]);

# }else{

# //for right

# sum+= right\_move(programs,ind,pos);

# // for left side

# sum += left\_move(programs,ind,programs[n-1]);

# }

# return sum;

# }

# int main()

# {

# int n,pos,dir;

# cout << "Enter number of programs and Initial position of Head"<<endl;

# cin>>n>>pos;

# cout << "Enter direction of head movement \*\*1 = Right and 0 = Left\*\*"<<endl;

# cin>>dir;

# vector<int> programs(n);

# cout<<"Enter programs"<<endl;

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

# cin>>programs[i];

# }

# int total\_movements=algo(programs,pos,dir);

# total\_movements=(float)total\_movements;

# cout<<"Average disk movement: "<<total\_movements/(float)n<<endl;

# return 0;

# }

# Output:

# 

# 