Mechi Multiple Campus

(Tribhuvan University)

Bhadrapur, Jhapa

Lab Report of Operating System(CACS-251)

Faculty of Humanities & Social Sciences

Tribhuvan University

Kritipur, Nepal

Submitted By

Name: Santosh Bhandari

Roll No: 58

Submitted To

Mechi Multiple Campus

Department of Bachelor in Computer

Bhadrapur, Jhapa, Nepal

Certificate from the supervisor

This is to certify that the Lab Report entitled **"Operating System"** is an academic work done by **"Santosh Bhandari"** submitted in the partial fulfillment of the requirements for the degree of **Bachelor of Computer Application** at Faculty of Humanities and Social Science, Tribhuvan University under my guidance and supervision. To the best of my knowledge, the worked performed by him in the Lab report is his own creation.

Signature of the Supervisor:-
Name:-
Designation:-
Date:-

Acknowledgement

I would like to express my special thanks of gratitude to my Operating System teacher "Mr. Raju Poudel" and to the whole faculty member of humanities and social science who has provided me this opportunity. I am really grateful to sir Raju Poudel for this kind of support in my project.

We are making this report as it compulsory required by Mechi Multiple Campus. It is compulsory report that should be submitted to the college in order to get practical marks. For this report our Scripting Language teacher has motivated us. We are also guided by him to score good marks in practical. He has suggested how to make the report. He was the main source for us to make the report ready for Mechi Multiple Campus. In this report, we the student of BCA were engaged and I would like to express my deep thankful particularly to all of them. We are doing this report to be very helpful for coming days.

Thank You.

Table Of Content

LAB SHEET - 1.

1. Write a program to create a process and display process id.

```
#include<stdio.h>
#include<unistd.h>
void main(){
    pid_t pid;
    pid=fork();
    if(pid<0){
        printf("Failed to Create a New process.");
    }else if(pid==0){
        printf("Child process is created.\n");
        printf("Process ID = %d\n",getpid());
    }else{
        printf("Parent Process is Created.\n");
        printf("Process ID = %d\n",getpid());
    }
}
```

Output

```
Parent Process is created.
Process ID = 16422
Child Process is created.
Process ID = 16423
```

LABSHEET-2

1. Write Syntax, Example and Description of Basic Linux Commands.

a. pwd
→ Show the location of the currently working directory.

Syntax :-
pwd

Example :-
santosh@santosh :~$ pwd
/home/santosh

b. clear
→ Used to clear the screen

Syntax :-
clear

Example :-
santosh@santosh :~$ clear

c. mkdir
→ used to create a new directory

Syntax :-
mkdir <directory_name>

Example :
santosh@santosh :~$ mkdir santoshTest

d. ls
→ Used to list content of currently working directory.

Syntax :-
ls

Example :-
santosh@santosh :~$ ls
Desktop Documents Downloads Music Pictures Public Snap Templates Videos

e. rmdir
→ Used to remove a directory.

Syntax:-
rmdir <directory_name>

Example:-
santosh@santosh:~ rmdir santoshTest

f. Touch
→ Used to Create Empty files.

Syntax:-
touch <file_name>

Example:-
santosh@santosh:~$ touch santosh.txt

g. rm
→ used to remove file from currently working directory.

Syntax:-
rm <file_name>

Example:-
santosh@santosh:~$ rm santosh.txt

h. cat
→ used to create a file, display content of file and copy the content of one file to another.

Syntax:-
cat <file_name>

Example:-
santosh@santosh:~$ cat > test.txt
learning Linux command
Santosh@santosh:~$ cat test.txt
learning Linux command

i. cd
→ Used to change the working directory.

Syntax:-
cd <directory_name>

Example:-
santosh@santosh:~$ cd Desktop
santosh@santosh:~/Desktop$

j. cp
→ used to copy file or directory from one location to another location.

Syntax:-
cp <existing_file_name> <new_file_name>

Example:-
santosh@santosh:~$ cp test.txt Desktop/text.txt

k. MV
→ used to move file or directory from one to another location.

Syntax:-
MV <file_name> <location>

Example:-
santosh@santosh:~$ MV test.txt Desktop/santosh

l. rename
→ Used to rename files.

Syntax:-
rename 's/old/newname/' files

Example:-
rename 's/test1.png/test3.txt/' test1.png

LABSHEET-3

1. WAP to Simulate FCFS process scheduling algorithm.

```
#include<stdio.h>
void main(){
    int p[10],bt[10],n,i,j,wt[10],tat[10];
    float awt=0,atat=0;
    printf("Enter the Number of Process (Max 10): ");
    scanf("%d",&n);
    printf("Enter process burst time:\n");
    for(i=0;i<n;i++){
        p[i]=i+1;
        scanf("%d",&bt[i]);
    }
    wt[0]=0;
    for(i=1;i<n;i++){
        wt[i]=0;
        for(j=0;j<i;j++)
            wt[i]=wt[i]+bt[j];
        awt=awt+wt[i];
    }
    for(i=0;i<n;i++){
        tat[i]=wt[i]+bt[i];
        atat=atat+tat[i];
    }
    printf("\nProcess\tBT\tWT\tTAT\n");
    for(i=0;i<n;i++)
        printf("p%d\t%d\t%d\t%d\t\n",p[i],bt[i],wt[i],tat[i]);
    printf("\nAWT=%.2f",awt/n);
    printf("\nATAT=%.2f",atat/n);
}
```

Output

```
Enter the Number of Process (MAX 10): 4
Enter process burst time:
3
2
5
4
```

Process	BT	WT	TAT
P1	3	0	3
P2	2	3	5
P3	5	5	10
P4	4	10	14

2. WAP to Simulate SJF Process Scheduling algorithm.

```
#include<stdio.h>
void main(){
    int p[10],bt[10],wt[10],tat[10],n,i,j,k,t;
    float awt,atat;
    printf("Enter the Number of process\n");
    scanf("%d",&n);
    printf("Enter Burst Time\n");
    for(i=0;i<n;i++){
        p[i]=i+1;
        scanf("%d",&bt[i]);
    }
    for(i=0;i<n;i++){
        for(j=i+1;j<n;j++){
            if(bt[i]>bt[j]){
                t=bt[i];
                bt[i]=bt[j];
                bt[j]=t;
                t=p[i];
                p[i]=p[j];
                p[j]=t;
            }
        }
    }
    wt[0]=0;
    for(i=1;i<n;i++){
        wt[i]=0;
        for(j=0;j<i;j++)
            wt[i]=wt[i]+bt[j];
        awt=awt+wt[i];
    }
```

```
for(i=0; i<n;i++){
    tat[i]=wt[i]+bt[i];
    atat=atat+tat[i];
}
printf("\nProcess\tBT|t WT\tTAT\n");
for(i=0;i<n;i++)
    printf("p%d\t%d\t%d\t%d\n",p[i],bt[i],wt[i],tat[i]);
printf("Average Waiting Time = %.2f ms\n Average Turn Around Time =
    %.2f ms", awt/n, atat/n);
}
```

Output

```
Enter the Number of process
4
Enter Burst time
5
2
4
3
```

Process	BT	WT	TAT
P2	2	0	2
P4	3	2	5
P3	4	5	9
P1	5	9	14

Average Waiting Time = 16.00ms
Average Turn Around Time = 30.00

3. WAP to Simulate Priority Process Scheduling Algorithm.

```
#include<stdio.h>
void main(){
    int p[10],bt[10],pr[10],wt[10],tat[10],n,j,i,t;
    float awt=0,atat=0;
    printf("Enter the Number of process (Max 10): ");
    scanf("%d",&n);
```

```
printf("Enter Burst Time and Priority of Each process :\n");
for(i=0; i<n; i++){
    p[i]=i+1;
    printf("Burst Time of p[%d] = ", i+1);
    scanf("%d", &bt[i]);
    printf("priority of p[%d] : ", i+1);
    scanf("%d", &pr[i]);
}
for(i=0; i<n; i++){
    for(j=i+1; j<n; j++){
        if(pr[i] > pr[j]){
            t=pr[i];
            pr[i]=pr[j];
            pr[j]=t;
            t=bt[i];
            bt[i]=bt[j];
            bt[j]=t;
            t=p[i];
            p[i]=p[j];
            p[j]=t;
        }
    }
}
wt[0]=0;
for(i=1; i<n; i++){
    wt[i]=0;
    for(j=0; j<i; j++)
        wt[i]=wt[i]+bt[j];
    awt=awt+wt[i];
}
for(i=0; i<n; i++){
    tat[i]=wt[i]+bt[i];
    atat=atat+tat[i];
}
```

```
printf("\nprocess\t BT\tPR\tWT\tTAT\n");
for(i=0;i<n;i++)
    printf("p%d\t%d\t%d\t%d\t%d\n",p[i],bt[i],pr[i],wt[i],
        tat[i]);
printf("Average Waiting Time = %.2f ms\n",awt/n);
printf("Average Turn Around Time = %.2f ms\n",atat\n);
}
```

Output

```
Enter the Number of process (max 10) : 3
Enter Burst Time and priority of Each process :
Burst Time of P[1] : 5
priority of P[1] : 1
Burst Time of P[2] : 4
Priority of P[2] : 2
Burst Time of P[3] : 5
priority of P[3] : 3
```

process	BT	PR	WT	TAT
P1	5	1	0	5
P2	4	2	5	9
P3	5	3	9	14

Average Waiting Time = 4.67 ms
Average Turn Around Time = 9.33 ms

4. WAP to Simulate Round Robin Process Scheduling Algorithm.

```
#include<stdio.h>
#include<stdbool.h>
void main(){
    int n, p[10],bt[10],wt[10],tat[10],i,j,quantum,rem_bt[10];
    float awt=0,atat=0;
    printf("Enter total Number of process(Max 10): ");
    scanf("%d",&n);
    printf("Enter process Burst Time:\n");
    for(i=0;i<n;i++){
```

```
    p[i]=i+1;
    scanf("%d",&bt[i]);
    rem_bt[i]=bt[i];
}
printf("Enter Time Quantum : ");
scanf("%d",&quantum);
int t=0;
while(1){
        bool done=true;
        for(i=0;i<n;i++){
            if(rem_bt[i]>0){
                done=false;
                if(rem_bt[i]>quantum){
                    t=t+quantum;
                    rem_bt[i]=rem_bt-quantum;
                }else{
                    t=t+rem_bt[i];
                    wt[i]=t-bt[i];
                    rem_bt[i]=0;
                }
            }
        }
        if(done==true)
            break;
}
printf("\nProcess\tB \tWT\t TAT\n");
for(i=0;i<n;i++){
    tat[i]=bt[i]+wt[i];
    awt=awt+wt[i];
    atat=atat+tat[i];
    printf("p%d\t%d\t%d\t%d\n",p[i],bt[i],wt[i],tat[i]);
}
printf("\nAverage Waiting Time = %.2f",awt/n);
printf("\nAverage Turnaround Time = %.2f",atat/n);
}
```

Output

Enter total Number of process (Max 10) : 3
Enter Process Burst Time
7
6
5
Enter Process Time Quantum : 3

Process	BT	WT	TAT
P1	7	11	18
P2	6	9	15
P3	5	12	17

Average Waiting Time = 10.67
Average Turn Around Time = 16.67

LABSHEET-4

1. WAP to simulate Bankers Algorithm.

```
#include<stdio.h>
void main(){
    int allocation[10][5], max[10][5], need[10][5], available[10], finish[10], safe[10];
    int n,m,i,j,k,count, count1=0;
    printf("Enter the no. of process : ");
    scanf("%d",&n);
    printf("Enter the no. of resource type : ");
    scanf("%d",&m);
    printf("\n Enter the MAX Matrix for each process : ");
    for(i=0;i<n;i++){
        printf("\n For Process %d :\n",i+1);
        for(j=0;j<m;j++)
            scanf("%d",&max[i][j]);
    }
    printf("\n Enter the allocation for each process : ");
    for(i=0;i<n;i++){
        printf("\n For Process %d :\n",i+1);
        for(j=0;j<m;j++)
            scanf("%d",&allocation[i][j]);
    }
    printf("\n Enter the Available Resource : \n");
    for(j=0;j<m;j++)
        scanf("%d",&available[j]);
    printf("\n The need Matrix is as follow :\n");
    for(i=0;i<n;i++){
        for(j=0;j<m;j++){
            need[i][j]=max[i][j]-allocation[i][j];
            printf("\t%d",need[i][j]);
        }
    }
    for(i=0;i<10;i++)
        finish[i]=0;
    do{
```

```
for(k=0; k<n; k++){
    for(i=0; i<n; i++){
        if(finish[i]==0){
            count=0;
            for(j=0; j<m; j++){
                if(available[j]>=need[i][j])
                    count++;
            }
            if(count==m){
                count1++;
                finish[i]=1;
                sq[count1-1]=i+1;
                for(j=0; j<m; j++)
                    available[j]=available[j]+allocation[i][j];
                break;
            }
        }
    }
    if(count1!=n){
        printf("\n IT'S AN UNSAFE STATE");
        break;
    }
}while(count1!=n);
if(count1!=n){
    printf("\n IT'S A SAFE STATE");
    printf("\n The Safe Sequence is : \n");
    for(i=0; i<n; i++)
        printf("p%d\t", sq[i]);
    printf("\n The Available array is now: ");
    for(j=0; j<m; j++)
        printf("%d\t", available[j]);
```

3

3

3

Output

Enter the no. of process : 3
Enter the no. of resource type : 1
Enter the MAX Matrix for each process :
For process 1 :
9
For process 2 :
4
For process 3 :
7
Enter the allocation for each process :
For process 1 :
3
For process 2 :
2
For process 3 :
2
Enter the Available Resource :
3
The Need Matrix is as follows :
6
2
5

It's A SAFE STATE
The Safe Sequence is
P2 P3 P1

The available array is now : 10

LABSHEET-5

1. WAP to Simulate FIFO Page replacement Algorithm.

```
#include<stdio.h>
void main(){
    int page[20],frame[20],pfcount=0,np,nf;
    int i,j,k=0,available;
    float pfrate;
    printf("Enter no. of frames: ");
    scanf("%d",&nf);
    printf("Enter no. of Pages: ");
    scanf("%d",&np);
    printf("Enter reference String:\n");
    for(i=0;i<np;i++)
        scanf("%d",&page[i]);
    for(i=0;i<nf;i++)
        frame[i]=-1;
    printf("Ref. String\t Frames\n");
    for(i=0;i<np;i++){
        printf("%d\t",page[i]);
        available=0;
        for(j=0;j<nf;j++){
            if(page[i]==frame[j]){
                available=1;
                printf("No page Replacement\n");
                break;
            }
        }
        if(available==0){
            frame[k]=page[i];
            k=(k+1)%nf;
            pfcount++;
            for(j=0;j<nf;j++)
                printf("%d\t",frame[j]);
            printf("\n");
        }
    }
```

```
        }
    }
    printf("\n Total Page fault = %d", pfcount);
    pfrate = (float) pfcount / (float) np;
    printf("\n Page fault Ratio = %.2f", pfrate * 100);
}
```

Output

```
Enter no. of frames: 3
Enter no. of Pages: 5
Enter reference String:
7
0
1
7
3
```

Ref. String	Frames		
7	7	-1	-1
0	7	0	-1
1	7	0	1
7	No Replacement		
3	3	0	1

```
Total Page Fault = 4
Page Fault Ratio = 80.00
```

2. WAP to Simulate LRU Page replacement algorithm.

```
#include<stdio.h>
int findLRU(int duration[], int n){
    int i, pos=0, min=duration[0];
    for(i=1; i<n; i++)
        if(duration[i]<min){
            min=duration[i];
            pos=i;
        }
    return pos;
}
```

```
void main(){
    int page[20], frame[10], pfcount=0, np, nf, i, j, flag1, flag2;
    float pfrate;
    int count=0, duration[20], pos;
    printf("Enter no. of frames : ");
    scanf("%d", &nf);
    printf("Enter no. of Pages : ");
    scanf("%d", &np);
    printf("Enter reference String : \n");
    for(i=0; i<np; i++)
        scanf("%d", &page[i]);
    for(i=0; i<nf; i++)
        frame[i]=-1;
    printf("Ref. String \t frames \n");
    for(i=0; i<np; i++){
        printf("%d\t", page[i]);
        flag1=flag2=0;
        for(j=0; j<nf; j++){
            if(frame[j]==page[i]){
                count++;
                duration[j]=count;
                flag1=flag2=1;
                break;
            }
            if(flag1==0 && frame[j]==-1){
                count++;
                pfcount++;
                frame[j]=page[i];
                duration[j]=count;
                flag2=1;
                break;
            }
        }
        if(flag2==0){
            pos=findLRU(duration, nf);
            count++;
```

```
            pfcount++;
            frame[pos] = page[i];
            duration[pos] = count;
        }
        for(j=0; j<nf; j++)
            printf("%d\t", frame[j]);
        printf("\n");
    }
    printf("\nTotal Page Fault = %d", pfcount);
    pfrate = (float)pfcount/(float)np;
    printf("\nPage Fault Ratio = %.2f", pfrate*100);
}
```

Output

```
Enter no. of frames: 3
Enter no. of pages: 7
Enter reference string:
7
0
1
2
3
1
7
```

Ref. String	Frames		
7	7	-1	-1
0	7	0	-1
1	7	0	1
2	2	0	1
3	2	3	1
1	2	3	1
7	7	3	1

Total Page Fault = 6

Page Fault Ratio = 85.71

3. WAP to Simulate Optimal Page replacement Algorithms.

```
#include<stdio.h>
Void main(){
    int nf,np,frame[10],pages[10],temp[10];
    int flag1,flag2,flag3,i,j,k,pos,max,pfcount=0;
    printf("Enter number of frames: ");
    Scanf("%d",&nf);
    printf("Enter number of pages:");
    scanf("%d",&np);
    printf("Enter reference String:\n");
    for(i=0;i<np;i++)
        scanf("%d",&page[i]);
    for(i=0;i<nf;i++)
        frame[i]=-1;
    for(i=0;i<np;i++){
        flag1=flag2=0;
        for(j=0;j<nf;j++)
            if(frame[j]==page[i]){
                flag1=flag2=1;
                break;
            }
        if(flag1==0){
            for(j=0;j<nf;j++)
                if(frame[j]==-1){
                    pfcount++;
                    frame[j]=page[i];
                    flag2=1;
                    break;
                }
        }
        if(flag2==0){
            flag3=0;
            for(j=0;j<nf;j++){
                temp[j]=-1;
                for(k=i+1;k<np;k++){
                    if(frame[j]==page[k]){
```

```
                temp[j]=k;
                break;
            }
        }
    }
    for(j=0;j<nf;j++)
        if(temp[j]==-1){
            pos=j;
            flag3=1;
            break;
        }
    if(flag3==0){
        max=temp[0];
        pos=0;
        for(j=1;j<nf;j++)
            if(temp[j]>max){
                max=temp[j];
                pos=j;
            }
    }
    frame[pos]=page[i];
    pfcount++;
}
printf("\n");
for(j=0;j<nf;j++){
    printf("%d\t",frame[j]);
}
}
printf("\nTotal Page Faults = %d",pfcount);
}
```

Output

```
Enter number of frames: 3
Enter number of pages: 7
Enter page reference string:
```

7
0
1
2
3
7
1

7	-1	-1
7	0	-1
7	0	1
7	2	1
7	3	1
7	3	1

Total Page Faults = 5

LABSHEET-6

1. WAP to Simulate FCFS disk Scheduling algorithm.

```
#include<stdio.h>
#include<stdlib.h>
void main(){
    int RQ[100], i, n, TotalHeadMoment=0, initial;
    printf("Enter the number of Requests : ");
    scanf("%d", &n);
    printf("Enter the Request Sequence:\n");
    for(i=0; i<n; i++)
        scanf("%d", RQ[i]);
    printf("Enter initial head position : ");
    scanf("%d", &initial);
    for(i=0; i<n; i++){
        TotalHeadMoment += abs(RQ[i] - initial);
        initial = RQ[i];
        printf("%d -> ", initial);
    }
    printf("\nTotal Head Moment is %d", TotalHeadMoment);
}
```

Output

```
Enter the number of Requests: 8
Enter the Request Sequence:
98
183
37
122
14
124
65
67
Enter Initial head Position: 53
98 -> 183 -> 37 -> 122 -> 14 -> 124 -> 65 -> 67 ->
Total Head Moment is 640.
```

2. WAP to simulate SSTF disk scheduling algorithm.

```
#include<stdio.h>
#include<stdlib.h>
void main(){
    int RQ[100],i,n,TotalHeadMoment=0,initial,count=0;
    printf("Enter the number of Requests: ");
    scanf("%d",&n);
    printf("Enter the requests sequence:\n");
    for(i=0;i<n;i++)
        scanf("%d",&RQ[i]);
    printf("Enter initial head position: ");
    scanf("%d",&initial);
    while(count!=n){
        int min=1000,distance,index;
        for(i=0;i<n;i++){
            distance=abs(RQ[i]-initial);
            if(min>distance){
                min=distance;
                index=i;
            }
        }
        TotalHeadMoment+=min;
        initial=RQ[index];
        printf("%d->",initial);
        RQ[index]=1000;
        count++;
    }
    printf("\n Total head Movement is %d.",TotalHeadMoment);
}
```

Output

```
Enter the number of Requests: 8
Enter the Request Sequence:
98
183
37
```

```
122
14
124
65
67
Enter the initial Head position : 53
65 → 67 → 37 → 14 → 98 → 122 → 124 → 183 →
Total Head Movement is 236.
```

3. WAP to Simulate SCAN disk Scheduling algorithm.

```
#include<stdio.h>
#include<stdlib.h>
void printseq(int value){
    printf("%d→", value);
}
void main(){
    int RQ[100], i, j, n, TotalHeadMoment=0, initial, size, move;
    printf("Enter the number of Requests : ");
    scanf("%d", &n);
    printf("Enter the Request Sequence :\n");
    for(i=0; i<n; i++)
        scanf("%d", &RQ[i]);
    printf("Enter Initial Head position : ");
    scanf("%d", &initial);
    printf("Enter total disk size : ");
    scanf("%d", &size);
    printf("Enter the Head movement direction for High 1 and for
        Low 0 : ");
    scanf("%d", &move);
    for(i=0; i<n; i++){
        for(j=0; j<n-i-1; j++)
            if(RQ[j] > RQ[j+1]){
                int temp;
                temp = RQ[j];
                RQ[j] = RQ[j+1]
                RQ[j+1] = temp;
```


3

```
int index;
for(i=0;i<n;i++)
    if(initial<RQ[i]){
        index=i;
        break;
    }
if(move==1){
    for(i=index;i<n;i++){
        TotalHeadMoment+=(abs(RQ[i]-initial));
        initial=RQ[i];
        printseq(initial);
    }
    TotalHeadMoment+=abs(size-RQ[i-1]-1);
    initial=size-1;
    for(i=index-1;i>=0;i--){
        TotalHeadMoment+=abs(RQ[i]-initial);
        initial=RQ[i];
        printseq(initial);
    }
}else{
    for(i=index-1;i>=0;i--){
        TotalHeadMoment+=abs(RQ[i]-initial);
        initial=RQ[i];
        printseq(initial);
    }
    TotalHeadMoment+=abs(RQ[i]-initial);
    initial=0;
    for(i=index;i<n;i++){
        TotalHeadMoment+=abs(RQ[i]-initial);
        initial=RQ[i];
        printseq(initial);
    }
}
printf("\nTotal Head Movement is %d",TotalHeadMoment);
}
```

Output

```
Enter the number of requests: 8
Enter the requests Sequence:
98
183
37
122
14
124
65
67
Enter initial head position: 53
Enter total disk size: 200
Enter the head movement direction for high 1 and for low 0: 1
65→67→98→122→183→37→14→
Total Head Movement is 331.
```

4. WAP to Simulate C-SCAN disk Scheduling algorithm.

```
#include<stdio.h>
#include<stdlib.h>
void printseq(int value){
        printf("%d -> ", value);
}
void main(){
    int RQ[100], i, j, n, TotalHeadMovment=0, initial, size, move, temp;
    printf("Enter the number of Requests: ");
    scanf("%d", &n);
    printf("Enter the Request Sequence: ");
    for(i=0; i<n; i++)
        scanf("%d", &RQ[i]);
    printf("Enter Initial Head Position: ");
    scanf("%d", &initial);
    printf("Enter total disk size: ");
    scanf("%d", &size);
    printf("Enter the head movement direction for high 1 and for low 0: ");
    scanf("%d", &move);
    for(i=0; i<n; i++){
        for(j=0; j<n-i-1; j++){
```

```
            if(RQ[j]>RQ[j+1]){
                temp=RQ[j];
                RQ[j]=RQ[j+1];
                RQ[j+1]=temp;
            }
        }
    }
    int index;
    for(i=0;i<n;i++){
        if(initial<RQ[i]){
            index=i;
            break;
        }
    }
    if(move==1){
        for(i=index;i<n;i++){
            TotalHeadMoment+=abs(RQ[i]-initial);
            initial=RQ[i];
            printseq(initial);
        }
        TotalHeadMoment+=abs(size-RQ[i-1]-1);
        TotalHeadMoment+=abs(size-1-0);
        initial=0;
        for(i=0;i<index;i++){
            TotalHeadMoment+=abs(RQ[i]-initial);
            initial=RQ[i];
            printseq(initial);
        }
    }else{
        for(i=index-1;i>=0;i--){
            TotalHeadMoment+=abs(RQ[i]-initial);
            initial=RQ[i];
            printseq(initial);
        }
        TotalHeadMoment+=abs(RQ[i+1]-0);
        TotalHeadMoment+=abs(size-1-0);
        initial=size-1;
        for(i=n-1;i>=index;i--){
```

```
            TotalHeadMoment += abs(RQ[i] - initial);
            initial = RQ[i];
            printseq(initial);
        }
    }
    printf("\nTotal Head Moment is %d.", TotalHeadMoment);
}
```

Output

```
Enter the Number of Requests : 8
Enter the Request Sequence :
98
183
37
122
14
124
65
67
Enter initial Head Position : 53
Enter total disk size : 200
Enter the head moment direction for High 1 and For low 0 : 1
65 -> 67 -> 98 -> 122 -> 124 -> 183 -> 14 -> 37 ->
Total Head Movement is 382.
```

5. WAP to simulate Look disk scheduling algorithm.

```
#include<stdio.h>
#include<stdlib.h>
void printseq(int value){
    printf("%d -> ", value);
}
void main(){
    int RQ[100], i, j, n, TotalHeadMoment=0, initial, size, move;
    printf("Enter the number of Requests : ");
    scanf("%d", &n);
    printf("Enter the Request Sequence :\n");
    for(i=0; i<n; i++)
        scanf("%d", &RQ[i]);
    printf("Enter Initial Head Position : ");
```

```
scanf("%d", &initial);
printf("Enter total disk size : ");
scanf("%d", &Size);
printf("Enter the Head Movement direction for High 1 and for low 0:");
scanf("%d", &move);
for(i=0;i<n;i++)
    for(j=0;j<n-1;j++)
        if(RQ[j] > RQ[j+1]) {
            int temp
            temp=RQ[j];
            RQ[j] = RQ[j+1];
            RQ[j+1] =temp;
        }
int index;
for(i=0;i<n;i++)
    if(initial<RQ[i]) {
        index=i;
        break;
    }
if(move==1){
    for(i=index;i<n;i++) {
        TotalHeadMoment += abs(RQ[i]-initial);
        initial = RQ[i];
        printseq(initial);
    }
    for(i=index-1;i>=0;i--){
        TotalHeadMoment += abs(RQ[i]-initial);
        initial = RQ[i];
        printseq(initial);
    }
}else{
    for(i=index-1;i>=0;i--){
        TotalHeadMoment += abs(RQ[i]-initial);
        initial = RQ[i];
        printseq(initial);
    }
    for(i=index;i<n;i++){
```

```
            TotalHeadMoment += abs(RQ[i] - initial);
            initial = RQ[i];
            printseq(initial);
        }
    }
    printf("\nTotal Head Moment is %d.", TotalHeadMoment);
}
```

Output

```
Enter the number of Requests : 8
Enter the request Sequence :
98
183
37
122
14
124
65
67
Enter Initial Head Position : 53
Enter total disk size : 200
Enter the Head movement direction for High 1 and For Low 0 : 1
65 -> 67 -> 98 -> 122 -> 183 -> 37 -> 14 ->
Total Head movement is 299.
```

6. WAP to Simulate C-Look disk scheduling algorithm.

```
#include<stdio.h>
#include<stdlib.h>
void printseq(int value){
    printf("%d->", value);
}
void main(){
    int RQ[100], i, j, n, TotalHeadMoment=0, initial, size, move, index, temp;
    printf("Enter the number of Requests : ");
    scanf("%d", &n);
    printf("Enter the Request Sequence :\n");
    for(i=0; i<n; i++)
        scanf("%d", &RQ[i]);
```

```
printf("Enter Initial Head Position : ");
scanf("%d",&Initial);
printf("Enter total disk Size : ");
scanf("%d",&Size);
printf("Enter the head Movement direction for High 1 and for low 0:");
scanf("%d",&move);
for(i=0;i<n;i++)
    for(j=0;j<n-i-1;j++)
        if(RQ[j]>RQ[j+1]){
            temp=RQ[j];
            RQ[j]=RQ[j+1];
            RQ[j+1]=temp;
        }
for(i=0;i<n;i++)
    if(Initial<RQ[i]){
        index=i;
        break;
    }
if(move==1){
    for(i=index;i<n;i++){
        TotalHeadMoment+=abs(RQ[i]-Initial);
        Initial=RQ[i];
        printseq(Initial);
    }
    for(i=0;i<index;i++){
        TotalHeadMoment+=abs(RQ[i]-Initial);
        Initial=RQ[i];
        printseq(Initial);
    }
}else{
    for(i=index-1;i>=0;i--){
        TotalHeadMoment+=abs(RQ[i]-Initial);
        Initial=RQ[i];
        printseq(Initial);
    }
```

```
        for(i=n-1; i>=index; i--){
            TotalHeadMoment += abs(RQ[i] - initial);
            initial = RQ[i];
            printseq(initial);
        }
    }
    printf("\nTotal Head Movement is %d.", TotalHeadMoment);
}
```

Output

```
Enter the number of Requests: 8
Enter the Requests Sequence:
98
183
37
122
14
124
65
67
Enter initial Head Position: 53
Enter total disk size: 199
Enter the head movement direction for High 1 and for Low 0: 1
65→67→98→122→124→183→14→37
Total Head Movement is 322.
```