Bachelor of Science in Computer Science and Information Technology

Laboratory Report On

Operating System

Submitted By :

Name : Basanta Rai
Semester : 4th
Section : B
Rollno : 23473

Submitted To :

Department Of Computer Science

S.N	Topic	Signature
1.	Creating a single process and displaying the id.	
2.	Wap to demonstrate thread creation and termination in linux	
3.	Simulating first come first served (FCFS) scheduling algorithm	
4.	Simulating Shortest job first (SJF) scheduling algorithm	
5.	Simulating Round Robin (RR) scheduling algorithm	
6.	Implementing Deadlock detection algorithm	
7.	Simulating FIFO page replacement algorithm	
8.	Simulating Second chance page replacement algorithm	
9.	Simulating Optimal page replacement algorithm	
10.	Simulating LRU page replacement algorithm	
11.	Simulating Clock page replacement algorithm	
12.	Simulating Best fit disk scheduling algorithm	
13.	Simulating Shortest seek time first disk scheduling algorithm	
14.	FCFS disk scheduling algorithm in c	

S.N	Topic	Signature
15.	SSTF disk scheduling algorithm	
16.	Simulating SCAN disk scheduling algorithm	
17.	Wap to implement C-SCAN disk scheduling algorithm in c	
18.	Wap to implement Look disk scheduling algorithm in c	
19.	Wap to implement C-Look disk scheduling algorithm in c	
20.	Implementing contiguous file allocation	
21.	Implementing linked list file allocation	

WAP to create a single process and display the id.

```
#include<stdio.h>
#include<unistd.h>

int main(){
        printf("Hello World!\n");
        fork();
        printf("I am after forking");
        printf("\t I am process %d. \n" , getpid());
        return 0;
}
```

OUTPUT:

```
Hello World!
I am after forking       I am process 4277.


...Program finished with exit code 0
Press ENTER to exit console.
```

Wap to demonstrate thread creation and termination in linux

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<pthread.h>

void * ThreadFun(void * tid){
        long * ThreadID= (long *) tid;
        printf("Hello World! This is Thread %ld\n",*ThreadID);
}

int main (){

        pthread_t tid0;
        pthread_t tid1;
        pthread_t tid2;
        pthread_t *pthread[]={&tid0,&tid1,&tid2};

        for (int i=0; i<3; i++){
                pthread_create(pthread[i],NULL,ThreadFun,(void *)pthread[i]);
        }
        pthread_exit(NULL);
        return 0;
}
```

Output:

```
Basanta@basanta:~$ gcc MThread.c -lpthread
Basanta@basanta:~$ ./a.out
Hello World! This is Thread 139962092205632
Hello World! This is Thread 139962083812928
Hello World! This is Thread 139962075420224
Basanta@basanta:~$
```

WAP for simulating First Come First Serve (FCFS) FIFO scheduling algorithm in C.

```
#include<stdio.h>
int main() {
        int n,bt[20],wt[20],tat[20],avwt=0,avtat=0,i,j;
        printf("Enter total number of processes(maximum 20):");
        scanf("%d",&n);
        printf("\nEnter Process Burst Time\n");
        for(i=0;i<n;i++) {
                printf("P[%d]:",i+1);
                scanf("%d",&bt[i]);
        }
        wt[0]=0; //waiting time for first process is 0
        //calculating waiting time
        for(i=1;i<n;i++) {
                wt[i]=0;
                for(j=0;j<i;j++)
                wt[i]+=bt[j];
        }
        printf("\nProcess\t\tBurst                    Time\tWaiting
Time\tTurnaround     Time");
 //calculating turnaround time
        for(i=0;i<n;i++)  {
                tat[i]=bt[i]+wt[i];
                avwt+=wt[i];
                avtat+=tat[i];
                printf("\nP[%d]\t\t%d\t\t%d\t\t%d",i+1,bt[i],wt[i],tat[i]);
        }
        avwt/=i;
        avtat/=i;
```

```
    printf("\n\nAverage Waiting Time:%d",avwt);
    printf("\nAverage Turnaround Time:%d",avtat);
    return 0;
}
```

OUTPUT:

```
Enter total number of processes(maximum 20):4

Enter Process Burst Time
P[1]:2
P[2]:2
P[3]:3
P[4]:5

Process         Burst Time      Waiting Time    Turnaround Time
P[1]            2               0               2
P[2]            2               2               4
P[3]            3               4               7
P[4]            5               7               12

Average Waiting Time:3
Average Turnaround Time:6
--------------------------------
Process exited after 5.909 seconds with return value 0
Press any key to continue . . .
```

WAP for simulating Shortest Job First (SJF) scheduling algorithm in C.

```
#include<stdio.h>
int main(){
        int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
        float avg_wt,avg_tat;
         printf("Enter number of process:");
        scanf("%d",&n);

        printf("\nEnter Burst Time:\n");
        for(i=0;i<n;i++) {
                printf("p%d:",i+1);
                scanf("%d",&bt[i]);
                p[i]=i+1; //contains process number
        }

         //sorting burst time in ascending order using selection sort
        for(i=0;i<n;i++)  {
                pos=i;
                for(j=i+1;j<n;j++) {
                        if(bt[j]<bt[pos])
                        pos=j;
                }
                temp=bt[i];
                 bt[i]=bt[pos];
                 bt[pos]=temp;
                 temp=p[i];
                p[i]=p[pos];
                p[pos]=temp;
        }
```

```
wt[0]=0; //waiting time for first process will be zero

 //calculate waiting time
 for(i=1;i<n;i++) {
            wt[i]=0;
            for(j=0;j<i;j++)
            wt[i]+=bt[j];

            total+=wt[i];
 }
 avg_wt=(float)total/n; //average waiting time
 total=0;

        printf("\nProcess\t        Burst        Time        \tWaiting
Time\tTurnaround Time");

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

            tat[i]=bt[i]+wt[i]; //calculate turnaround time
            total+=tat[i];
                        printf("\np%d\t\t                %d\t\t
           %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);

 }
 avg_tat=(float)total/n; //average turnaround time
 printf("\n\nAverage Waiting Time=%f",avg_wt);
 printf("\nAverage Turnaround Time=%f\n",avg_tat);
 return 0;
}
```

OUTPUT:

```
Enter number of process:4

Enter Burst Time:
p1:3
p2:2
p3:5
p4:3

Process     Burst Time          Waiting Time    Turnaround Time
p2                2                   0                   2
p1                3                   2                   5
p4                3                   5                   8
p3                5                   8                   13

Average Waiting Time=3.750000
Average Turnaround Time=7.000000

--------------------------------
Process exited after 8.366 seconds with return value 0
Press any key to continue . . .
```

WAP to simulate Round Robin Algorithm in C.

```
#include<stdio.h>
int main()
{
        int count,j,n,time,remain,flag=0,time_quantum;
        int wait_time=0,turnaround_time=0,at[10],bt[10],rt[10];
        printf("Enter Total Process:\t ");
        scanf("%d",&n);
        remain=n;
        for(count=0;count<n;count++)  {
                printf("Enter Arrival Time and Burst Time for Process Process
               Number %d :",count+1);   scanf("%d",&at[count]);
                scanf("%d",&bt[count]);
                rt[count]=bt[count];
        }
        printf("Enter Time Quantum:\t");
        scanf("%d",&time_quantum);
        printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
        for(time=0,count=0;remain!=0;)  {
                if(rt[count]<=time_quantum && rt[count]>0)   {
                        time+=rt[count];
                        rt[count]=0;
                        flag=1;
                }  else if(rt[count]>0)  {
                        rt[count]-=time_quantum;
                        time+=time_quantum;
                }
                if(rt[count]==0 && flag==1)   {
                                                                        remain--;
                        printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-bt[coun
                        t]);   wait_time+=time-at[count]-bt[count];
```

```
                turnaround_time+=time-at[count];
                flag=0;
            }
            if(count==n-1)
                count=0;
            else if(at[count+1]<=time)
                count++;
            else
                count=0;
        }
        printf("\nAverage Waiting Time= %f\n",wait_time*1.0/n);
        printf("Avg Turnaround Time = %f",turnaround_time*1.0/n);

        return 0;
}
```

OUTPUT:

```
Enter Total Process:     4
Enter Arrival Time and Burst Time for Process Process Number 1 :0 5
Enter Arrival Time and Burst Time for Process Process Number 2 :1 4
Enter Arrival Time and Burst Time for Process Process Number 3 :3 2
Enter Arrival Time and Burst Time for Process Process Number 4 :4 2
Enter Time Quantum:     2


Process |Turnaround Time|Waiting Time

P[3]    |       3       |       1
P[4]    |       4       |       2
P[2]    |       11      |       7
P[1]    |       13      |       8

Average Waiting Time= 4.500000
Avg Turnaround Time = 7.750000
--------------------------------
Process exited after 30.73 seconds with return value 0
Press any key to continue . . .
```

WAP to implement a deadlock detection algorithm.

```
#include<stdio.h>
static int mark[20];
int i,j,np,nr;

int main()
{
int alloc[10][10],request[10][10],avail[10],r[10],w[10];

printf("\nEnter the no of process: ");
scanf("%d",&np);
printf("\nEnter the no of resources: ");
scanf("%d",&nr);
for(i=0;i<nr;i++)
{
printf("\nTotal Amount of the Resource R%d:
",i+1); scanf("%d",&r[i]);
}
printf("\nEnter the request matrix:");
for(i=0;i<np;i++)
for(j=0;j<nr;j++)
scanf("%d",&request[i][j]);

printf("\nEnter the allocation matrix:");
for(i=0;i<np;i++)
for(j=0;j<nr;j++)
scanf("%d",&alloc[i][j]);
/*Available Resource calculation*/
```

```
for(j=0;j<nr;j++)
{
avail[j]=r[j];
for(i=0;i<np;i++)
{
avail[j]-=alloc[i][j];
}
}
//marking processes with zero
allocation for(i=0;i<np;i++)
{
int count=0;
 for(j=0;j<nr;j++)
 {
 if(alloc[i][j]==0)
 count++;
 else
 break;
 }
 if(count==nr)
 mark[i]=1;
}
// initialize W with avail
for(j=0;j<nr;j++)
 w[j]=avail[j];
//mark processes with request less than or equal
to W for(i=0;i<np;i++)
{
int canbeprocessed=0;
 if(mark[i]!=1){
```

```
for(j=0;j<nr;j++)
{
if(request[i][j]<=w[j])
canbeprocessed=1;
else
{
canbeprocessed=0;
break;
}
}
if(canbeprocessed)
{
mark[i]=1;
for(j=0;j<nr;j++)
w[j]+=alloc[i][j];
}
}
}
//checking for unmarked
processes int deadlock=0;
for(i=0;i<np;i++)
if(mark[i]!=1)
deadlock=1;
if(deadlock)
printf("\n Deadlock
detected"); else
printf("\n No Deadlock
possible"); }
```

OUTPUT:

```
Enter the no of process: 4

Enter the no of resources: 5

Total Amount of the Resource R1: 2

Total Amount of the Resource R2: 1

Total Amount of the Resource R3: 1

Total Amount of the Resource R4: 2

Total Amount of the Resource R5: 1

Enter the request matrix:0 1 0 0 1
0 0 1 0 1
0 0 0 0 1
1 0 1 0 1

Enter the allocation matrix:1 0 1 1 0
1 1 0 0 0
0 0 0 1 0
0 0 0 0 0

 Deadlock detected
```

WAP for simulating FIFO page replacement .

```
#include<stdio.h>
int main()
{
int i,j,n,a[50],frame[10],no,k,avail,count=0;
printf("\n ENTER THE NUMBER OF PAGES:\n");
scanf("%d",&n);
printf("\n ENTER THE PAGE NUMBER :\n");
for(i=1;i<=n;i++)
scanf("%d",&a[i]);
printf("\n ENTER THE NUMBER OF FRAMES :");
scanf("%d",&no);
for(i=0;i<no;i++)
frame[i]= -1;
 j=0;
printf("\tref string\t page frames\n");
for(i=1;i<=n;i++)
{
printf("%d\t\t",a[i]);
avail=0;
for(k=0;k<no;k++)
if(frame[k]==a[i])
avail=1;
if (avail==0)
 {
frame[j]=a[i];
j=(j+1)%no;
count++;
for(k=0;k<no;k++)
```

```
printf("%d\t",frame[k]);

}

printf("\n");

}

printf("Page Fault Is %d",count);

return 0;

}
```

OUTPUT:

```
 ENTER THE PAGE NUMBER :
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

 ENTER THE NUMBER OF FRAMES :3
        ref string        page frames
7                7       -1      -1
0                7       0       -1
1                7       0       1
2                2       0       1
0
3                2       3       1
0                2       3       0
4                4       3       0
2                4       2       0
3                4       2       3
0                0       2       3
3
2
1                0       1       3
2                0       1       2
0
1
7                7       1       2
0                7       0       2
1                7       0       1
Page Fault Is 15
--------------------------------
```

WAP for simulating second chance page replacement .

```
#include<stdio.h>
#include<conio.h>
#define SIZE 3
int full = 0;
int a[21];
int ref[SIZE];
int frame[SIZE];
int repptr=0;
int count=0;
int display(){
        int i;
        printf("\n The elements in the frame are \n");
        for(i=0;i<full;i++)
        printf("%d\n", frame[i]);

}
int Pagerep(int ele){
        int temp;
        while(ref[repptr]!=0){
                ref[repptr++]=0;
                if(repptr==SIZE)
                repptr=0;
        }
        temp = frame[repptr];
        frame[repptr]=ele;
        ref[repptr]=1;
        return temp;
}
```

```
int Pagefault(int ele){
        if(full!=SIZE){
                ref[full]=1;
                frame[full++]=ele;
        }
        else
        printf("The page replaced is %d",Pagerep(ele));
}
int Search(int ele){
        int i, flag;
        flag = 0;
        if(full!=0){
                for(i=0;i<full;i++)
                if(ele==frame[i]){
                        flag=1;
                        ref[i]=1;
                        break;
                }
        }
        return flag;
        }

        int main(){
                int n, i;
                printf("The number of elements in the reference string
                are:"); scanf("%d", &n);
                printf("%d",n);
                printf("Enter elements");
                for(i=0;i<n;i++)
                scanf("%d",&a[i]);
```

```
        printf("\n The elements present in the string are\n");
        for(i=0;i<n;i++)
        printf("%d", a[i]);
        printf("\n\n");
        for(i=0;i<n;i++){
            if(Search(a[i])!=1){
                Pagefault(a[i]);
                display();
                count++;
            }
        }
        printf("\n The number of page faults are %d\n",
        count); getch();
        return 0;
    }
```

OUTPUT:

```
The number of elements in the reference string are: 12
12Enter elements 2 3 2 1 5 2 4 5 3 2 5 2

 The elements present in the string are
232152453252


 The elements in the frame are
2

 The elements in the frame are
2
3

 The elements in the frame are
2
3
1
The page replaced is 2
 The elements in the frame are
5
3
1
The page replaced is 3
 The elements in the frame are
5
2
1
```

```
The page replaced is 1
 The elements in the frame are
5
2
4
The page replaced is 2
 The elements in the frame are
5
3
4
The page replaced is 4
 The elements in the frame are
5
3
2

 The number of page faults are 8
```

WAP for simulating optimal page replacement .

```
#include<stdio.h>
int main()
{
 int no_of_frames, no_of_pages, frames[10], pages[30], temp[10], flag1, flag2, flag3, i, j, k, pos, max,  faults = 0;
 printf("Enter number of frames: ");
 scanf("%d", &no_of_frames);

 printf("Enter number of pages: ");
 scanf("%d", &no_of_pages);

 printf("Enter page reference string: ");

 for(i = 0; i < no_of_pages; ++i){
 scanf("%d", &pages[i]);
 }

 for(i = 0; i < no_of_frames; ++i){
 frames[i] = -1;
 }

 for(i = 0; i < no_of_pages; ++i){
 flag1 = flag2 = 0;

 for(j = 0; j < no_of_frames; ++j){
 if(frames[j] == pages[i]){
 flag1 = flag2 = 1;
 break;
 }
```

```
}

if(flag1 == 0){
 for(j = 0; j < no_of_frames; ++j){
if(frames[j] == -1){
faults++;
frames[j] = pages[i];
flag2 = 1;
break;
}
}
}

if(flag2 == 0){
flag3 =0;

 for(j = 0; j < no_of_frames; ++j){
temp[j] = -1;

 for(k = i + 1; k < no_of_pages; ++k){
if(frames[j] == pages[k]){   temp[j] = k;
break;
}
}
}

 for(j = 0; j < no_of_frames; ++j){
if(temp[j] == -1){
pos = j;
```

```
flag3 = 1;
break;
}
}

if(flag3 ==0){
max = temp[0];
pos = 0;

 for(j = 1; j < no_of_frames; ++j){
if(temp[j] > max){
max = temp[j];  pos = j;
}
}
}

                        frames[pos] = pages[i];
                        faults++;
}

printf("\n");

for(j = 0; j < no_of_frames; ++j){
printf("%d\t", frames[j]);
}
}

printf("\n\nTotal Page Faults = %d", faults);

return 0;
```

}

OUTPUT:

```
Enter number of frames: 3
Enter number of pages: 10
Enter page reference string: 2 3 4 2 1 3 7 5 4 3

2       -1      -1
2       3       -1
2       3       4
2       3       4
1       3       4
1       3       4
7       3       4
5       3       4
5       3       4
5       3       4

Total Page Faults = 6
--------------------------------
```

WAP for simulating LRU page replacement algorithm.

```
#include<stdio.h>
int main()
{
 int frames[10], temp[10], pages[10];
   int total_pages, m, n, position, k, l,
total_frames;  int a = 0, b = 0, page_fault = 0;
 printf("\nEnter Total Number of Frames:\t");
 scanf("%d", &total_frames);
 for(m = 0; m < total_frames; m++)
 {
 frames[m] = -1;
 }
 printf("Enter Total Number of Pages:\t");
 scanf("%d", &total_pages);
 printf("Enter Values for Reference String:\n");
 for(m = 0; m < total_pages; m++)
 {
 printf("Value No.[%d]:\t", m + 1);
 scanf("%d", &pages[m]);
 }
 for(n = 0; n < total_pages; n++)
 {
 a = 0, b = 0;
 for(m = 0; m < total_frames; m++)
 {
 if(frames[m] == pages[n])
 {
 a = 1;
 b = 1;
```

```
break;
}
}
if(a == 0)
{
for(m = 0; m < total_frames; m++)  {
if(frames[m] == -1)
{
frames[m] = pages[n];
b = 1;
break;
}
}
}
if(b == 0)
{
for(m = 0; m < total_frames; m++)  {
temp[m] = 0;
}
for(k = n - 1, l = 1; l <= total_frames - 1; l++, k--)  {
for(m = 0; m < total_frames; m++)  {
if(frames[m] == pages[k])
{
temp[m] = 1;
} } }
for(m = 0; m < total_frames; m++)
{
if(temp[m] == 0)
position = m;
}
```

```
 frames[position] = pages[n];
 page_fault++;
 }
 printf("\n");
 for(m = 0; m < total_frames; m++)
 {
 printf("%d\t", frames[m]);
 } }
    printf("\nTotal Number of Page Faults:\t%d\n",
page_fault);  return 0;
}
```

OUTPUT:

```
Enter Total Number of Frames:   4
Enter Total Number of Pages:    5
Enter Values for Reference String:
Value No.[1]:   5
Value No.[2]:   3
Value No.[3]:   1
Value No.[4]:   2
Value No.[5]:   4

5       -1      -1      -1
5       3       -1      -1
5       3       1       -1
5       3       1       2
4       3       1       2
Total Number of Page Faults:    1
```

WAP for simulating CLOCK page replacement .

```
#include<stdio.h>
int main()
{
 int n,p[100],f[10],ava,hit=0,usebit[10],i,j;
 printf("enter the length of the Reference string: ");
 scanf("%d",&n);
 printf("enter the reference string: \n");
 for(i=0;i<n;i++)
 scanf("%d",&p[i]);
 for(i=0;i<n;i++)
 {
 ava=0;
 // found
 for(j=0;j<3;j++)
 {
 if(p[i]==f[j])
 {
 ava=1;
 hit++;
 usebit[j]=1;
 break;
 }
 }
 //search for usebit 0
 if(ava==0)
 {
 for(j=0;j<3;j++)
 {
```

```
if(usebit[j]==0)
{
f[j]=p[i];
usebit[j]=1;
ava=1;
break;
} } }
// fifo
if(ava==0)
{
for(j=0;j<3;j++)
usebit[j]=0;
}
f[0]=p[i];
usebit[0]=1;
}
printf("The number of Hits: %d",hit);
return 0;
}
```

OUTPUT:

```
enter the length of the Reference string: 10
enter the reference string:
1
2
3
4
3
1
2
1
5
4
The number of Hits: 2
--------------------------------
```

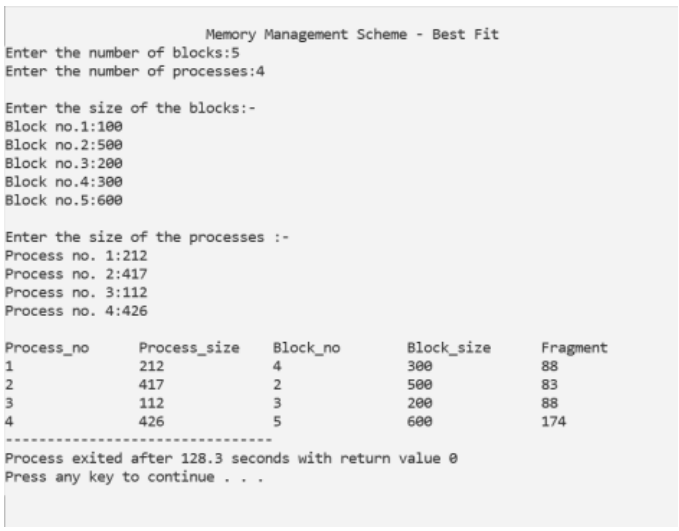
WAP to simulate Best Fit disk scheduling algorithm.

```
#include<iostream>
using namespace std;

int main()
{
int
fragment[20],b[20],p[20],i,j,nb,np,temp,lowest=999
9; static int barray[20],parray[20];
cout<<"\n\t\t\tMemory Management Scheme -
Best Fit"; cout<<"\nEnter the number of blocks:";
cin>>nb;
cout<<"Enter the number of processes:";
cin>>np;
cout<<"\nEnter the size of the blocks:-\n";
for(i=1;i<=nb;i++)
 {
 cout<<"Block no."<<i<<":";
 cin>>b[i];
 }
cout<<"\nEnter the size of the processes :-\n";
for(i=1;i<=np;i++)
 {
 cout<<"Process no. "<<i<<":";
 cin>>p[i];
 }
for(i=1;i<=np;i++)
{
for(j=1;j<=nb;j++)
```

```
{
if(barray[j]!=1)
{
temp=b[j]-p[i];
if(temp>=0)
if(lowest>temp)
{
parray[i]=j;
lowest=temp;
}
}
}
fragment[i]=lowest;
barray[parray[i]]=1;
lowest=10000;
}
cout<<"\nProcess_no\tProcess_size\tBlock_no\tBlock_size\tFrag
ment"; for(i=1;i<=np && parray[i]!=0;i++)
cout<<"\n"<<i<<"\t\t"<<p[i]<<"\t\t"<<parray[i]<<"\t\t"<<b[parray[i]]<<"\t\t"<<frag
ment[i]; return 0;
}
```

OUTPUT:



```
                    Memory Management Scheme - Best Fit
Enter the number of blocks:5
Enter the number of processes:4

Enter the size of the blocks:-
Block no.1:100
Block no.2:500
Block no.3:200
Block no.4:300
Block no.5:600

Enter the size of the processes :-
Process no. 1:212
Process no. 2:417
Process no. 3:112
Process no. 4:426

Process_no      Process_size    Block_no        Block_size      Fragment
1               212             4               300             88
2               417             2               500             83
3               112             3               200             88
4               426             5               600             174
--------------------------------
Process exited after 128.3 seconds with return value 0
Press any key to continue . . .
```

12: WAP to simulate Shortest Seek Time First disk scheduling algorithm.

```
#include<conio.h>
#include<stdio.h>
struct di
{
int num
;
int flag;
};
int main()
{
int i,j,sum=0,n,min,loc,x,y;
struct di d[20];
int disk;
int ar[20],a[20];
printf("enter number of location\t");
scanf("%d",&n);
printf("enter position of head\t");
scanf("%d",&disk);
printf("enter elements of disk queue\n");
for(i=0;i<n;i++)
{

scanf("%d",&d[i].num); d[i]. flag=0;
}
for(i=0;i<n;i++)
{ x=0; min=0;loc=0;
for(j=0;j<n;j++)
{
if(d[j].flag==0)
```

```
{
if(x==0)
{
ar[j]=disk-d[j].num;
if(ar[j]<0){ ar[j]=d[j].num-disk;}
min=ar[j];loc=j;x++; }
else
{
ar[j]=disk-d[j].num;
if(ar[j]<0){ ar[j]=d[j].num-disk;}
}
if(min>ar[j]){ min=ar[j]; loc=j;}
}
}
d[loc].flag=1;
a[i]=d[loc].num-disk;
if(a[i]<0){a[i]=disk-d[loc].num;}
disk=d[loc].num;
}
for(i=0;i<n;i++)
{
sum=sum+a[i];
}
printf("\nmovement of total cylinders %d",sum);
getch();
return 0;
}
```

OUTPUT:

```
enter number of location        9
enter position of head  143
enter elements of disk queue
86
1470
913
1774
948
1509
1022
1750
130

movement of total cylinders 1745
```

WAP to simulate SCAN disk scheduling algorithm.

```
#include<conio.h>
#include<stdio.h>
int main()
{
int i,j,sum=0,n;
int d[20];
int disk; //loc of head
int temp,max;
int dloc; //loc of disk in array
printf("enter number of location\t");
scanf("%d",&n);
printf("enter position of head\t");
scanf("%d",&disk);
printf("enter elements of disk queue\n");
for(i=0;i<n;i++)
{
scanf("%d",&d[i]);
}
d[n]=disk;
n=n+1;
for(i=0;i<n;i++) // sorting disk locations
{
for(j=i;j<n;j++)
{
if(d[i]>d[j])
{
temp=d[i];
```

```
d[i]=d[j];
d[j]=temp;
}
}

}
max=d[n];
for(i=0;i<n;i++) // to find loc of disc in array
{
if(disk==d[i]) { dloc=i; break; }
}
for(i=dloc;i>=0;i--)
{
printf("%d -->",d[i]);
}
printf("0 -->");
for(i=dloc+1;i<n;i++)
{
printf("%d-->",d[i]);
}
sum=disk+max;
printf("\nmovement of total cylinders %d",sum);
getch();
return 0;
}
```

OUTPUT:

```
enter number of location        9
enter position of head  143
enter elements of disk queue
86
1470
913
1774
948
1509
1022
1750
130
143 -->130 -->86 -->0 -->913-->948-->1022-->1470-->1509-->1750-->1774-->
movement of total cylinders 144
```

FCFS Disk scheduling algorithm in c

Source Code:

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

Output:

```
Enter the number of Requests
4
Enter the Requests sequence
70
30
10
40
Enter initial head position
55
Total head moment is 105
Name:Basanta Rai  Roll:23473 Sec:B 4th sem
```

SSTF DISK SCHEDULING ALGORITHM

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
 int RQ[100], i, n, TotalHeadMoment = 0, initial, count = 0;
printf("Enter the number of Requests\n");
 scanf("%d", &n);
 printf("Enter the Requests sequence\n");
 for (i = 0; i < n; i++)
 scanf("%d", &RQ[i]);
 printf("Enter initial head position\n");
 scanf("%d", &initial);

 // logic for sstf disk scheduling:

/* loop will execute until all process is
completed*/while (count != n)

 {
 int min = 1000, d, index;
 for (i = 0; i < n; i++)
 {
 d = abs(RQ[i] - initial);
 if (min > d)
 {
 min = d;
 index = i;
 }
 }
 TotalHeadMoment = TotalHeadMoment + min;  initial
= RQ[index];
 // 1000 is for max
 // you can use any number
 RQ[index] = 1000;
 count++;
 }

  printf("Total head movement is %d", TotalHeadMoment);
printf("Name:Basanta Rai Roll:23473 Sec:B 4th sem");  return
```

0;
}

Output

```
Enter the number of Requests
4
Enter the Requests sequence
20
40
80
60
Enter initial head position
33
Total head movement is 87
Name:Basanta Rai  Roll:23473 Sec:B 4th sem
```

C-SCAN DISK SCHEDULING ALGORITHM

```
#include <stdio.h>

#include <stdlib.h>

int main()

{int queue[20], n, head, i, j, k, seek = 0, max, diff, temp, queue1[20], queue2[20], temp1 = 0, temp2
=  0;

 float avg;

  printf("Enter the max range of disk\n");

 scanf("%d", &max);

 printf("Enter the initial head position\n");

 scanf("%d", &head);

 printf("Enter the size of queue request\n");

 scanf("%d", &n);

 printf("Enter the queue of disk positions to be read\n");
```

```
for (i = 1; i <= n; i++)

{

scanf("%d", &temp);

if (temp >= head)

{

queue1[temp1] = temp;

temp1++;

}

else

{

queue2[temp2] = temp;

temp2++;

} }

for (i = 0; i < temp1 - 1; i++)
{ for (j = i + 1; j < temp1; j++)

{ if (queue1[i] > queue1[j])

{ temp = queue1[i];

queue1[i] = queue1[j];

queue1[j] = temp;

}} }

for (i = 0; i < temp2 - 1; i++)

{

for (j = i + 1; j < temp2; j++)

{

if (queue2[i] > queue2[j])

{

temp = queue2[i];
```

```
 queue2[i] = queue2[j];
 queue2[j] = temp;
 }} }
  for (i = 1, j = 0; j < temp1; i++, j++)
queue[i] = queue1[j];
 queue[i] = max;
 queue[i + 1] = 0;
  for (i = temp1 + 3, j = 0; j < temp2; i++, j++)
queue[i] = queue2[j];
 queue[0] = head;
 for (j = 0; j <= n + 1; j++)
 {
  diff = abs(queue[j + 1] - queue[j]);  seek
+= diff;
 printf("Disk head moves from %d to %d with seek %d\n", queue[j], queue[j + 1], diff);  }
 printf("Total seek time is %d\n", seek);
 avg = seek / (float)n;
 printf("Average seek time is %f\n", avg);
printf("Name:Basanta Rai Roll:23473 Sec:B 4th sem");
return 0;
}
```

Output:

```
Enter the max range of disk
200
Enter the initial head position
22
Enter the size of queue request
3
Enter the queue of disk positions to be read
55
66
21
Disk head moves from 22 to 55 with seek 33
Disk head moves from 55 to 66 with seek 11
Disk head moves from 66 to 200 with seek 134
Disk head moves from 200 to 0 with seek 200
Disk head moves from 0 to 21 with seek 21
Total seek time is 399
Average seek time is 133.000000
Name:Basanta Rai   Roll:23473 Sec:B 4th sem
```

Wap to implement look algorithm in c

```
#include<stdio.h>

#include<stdlib.h>

int main()

{

    int     RQ[100],i,j,n,TotalHeadMoment=0,initial,size,move;

printf("Enter the number of Requests\n");

 scanf("%d",&n);

 printf("Enter the Requests sequence\n");

 for(i=0;i<n;i++)

 scanf("%d",&RQ[i]);

 printf("Enter initial head position\n");
```

```
scanf("%d",&initial);
printf("Enter total disk size\n");
scanf("%d",&size);
printf("Enter the head movement direction for high 1 and for low 0\n");
scanf("%d",&move);

// logic for look disk scheduling

/*logic for sort the request array */
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;
}

}
}

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

```
if(initial<RQ[i])
{
index=i;
break;
}
}

// if movement is towards high value
if(move==1)
{
for(i=index;i<n;i++)
{
        TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
initial=RQ[i];
}

for(i=index-1;i>=0;i--)
{
        TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
initial=RQ[i];

}
}
// if movement is towards low value
else
{
```

```
 for(i=index-1;i>=0;i--)
 {
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
initial=RQ[i];
 }

 for(i=index;i<n;i++)
 {
             TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
initial=RQ[i];

 }
 }

  printf("Total head movement is %d \n",TotalHeadMoment);
printf("Name:Basanta Rai Roll:23473 Sec:B 4th sem");
 return 0;
}
```

Output:

```
Enter the number of Requests
4
Enter the Requests sequence
22
33
44
66
Enter initial head position
21
Enter total disk size
200
Enter the head movement direction for high 1 and for low 0
1
Total head movement is 108
Name:Basanta Rai   Roll:23473 Sec:B 4th sem
```

WAP to implement c-look algorithm in c

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
 int RQ[100], i, j, n, TotalHeadMoment = 0, initial, size, move;
printf("Enter the number of Requests\n");
 scanf("%d", &n);
 printf("Enter the Requests sequence\n");
 for (i = 0; i < n; i++)
 scanf("%d", &RQ[i]);
 printf("Enter initial head position\n");
 scanf("%d", &initial);
 printf("Enter total disk size\n");
 scanf("%d", &size);
 printf("Enter the head movement direction for high 1 and for low 0\n");
scanf("%d", &move);

 // logic for C-look disk scheduling

 /*logic for sort the request array */
 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;

}

}

}


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 = TotalHeadMoment + abs(RQ[i] - initial);  initial =

RQ[i];

}


for (i = 0; i < index; i++)

{
```

```
TotalHeadMoment = TotalHeadMoment + abs(RQ[i] - initial);
initial = RQ[i];

 }

}

else

{

for (i = index - 1; i >= 0; i--)

{

 TotalHeadMoment = TotalHeadMoment + abs(RQ[i] - initial);  initial =

RQ[i];

 }for (i = n - 1; i >= index; i--)

  {TotalHeadMoment  =  TotalHeadMoment  +  abs(RQ[i]  -  initial);

initial = RQ[i];}}

printf("Total head movement is %d \n", TotalHeadMoment);

printf("Name:Basanta Rai Roll:23473 Sec:B 4th sem");

 return 0;}
```

Output:

```
Enter the number of Requests
4
Enter the Requests sequence
44
67
12
34
Enter initial head position
22
Enter total disk size
450
Enter the head movement direction for high 1 and for low 0
0
Total head movement is 98
Name:Basanta Rai  Roll:23473 Sec:B 4th sem
```

Implementing Contiguous file allocation technique

Source Code:

```
#include<stdio.h>
#include<conio.h>
#include<windows.h>
int main()
{
        int f[50], i, st, len, j, c, k, count = 0;
        for(i=0;i<50;i++)
                f[i]=0;
        printf("Files Allocated are : \n");
        x: count=0;
        printf("Enter starting block and length of files:");
        scanf("%d%d", &st,&len);
        for(k=st;k<(st+len);k++)
        if(f[k]==0)
        count++;
        if(len==count)
{
for(j=st;j<(st+len);j++)
if(f[j]==0)
{
f[j]=1;
printf("%d\t%d\n",j,f[j]);
}
if(j!=(st+len-1))
printf(" The file is allocated to disk\n");
```

```
        }

        else

printf(" The file is not allocated \n");
        printf("Do you want to enter more file(Yes - 1/No - 0)");

        scanf("%d", &c);

        if(c==1)goto x;

        else

        exit(0);

printf("Name:Basanta  Rai  Roll:23473  Sec:B  4th  sem");

return 0;

}
```

Output:

```
Files Allocated are :
Enter starting block and length of files:4 5
4   1
5   1
6   1
7   1
8   1
 The file is allocated to disk
Do you want to enter more file(Yes - 1/No - 0)1
Enter starting block and length of files:3 4
The file is not allocated
Do you want to enter more file(Yes - 1/No - 0)
Name:Basanta Rai Roll:23473 sec :B 4th sem
```

Implementing Linked List file allocation technique

Source Code:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int main()
{
        int f[50], p,i, st, len, j, c, k, a;
        for(i=0;i<50;i++)
                f[i]=0;
        printf("Enter how many blocks already allocated: ");
        scanf("%d",&p);
        printf("Enter blocks already allocated: ");
        for(i=0;i<p;i++)
                {
                        scanf("%d",&a);
                        f[a]=1;
                }
        x: printf("Enter index starting block and length: ");
        scanf("%d%d", &st,&len);
        k=len;
        if(f[st]==0)
                {
                        for(j=st;j<(st+k);j++)
                        {
                        if(f[j]==0)
                        {
```

```
                    f[j]=1;
                    printf("%d-------->%d\n",j,f[j]);
                }
                else
                {
                    printf("%d Block is already allocated \n",j);
                    k++;
                }}}
        else
        printf("%d starting block is already allocated \n",st);
        printf("Do you want to enter more file(Yes - 1/No - 0)");
        scanf("%d", &c);
        if(c==1)
        goto x; else
        exit(0);
 printf("Name:Basanta Rai Roll:23473 sec:B 4th sem");

getch();

}
```

Output:

```
Enter how many blocks already allocated: 2
Enter blocks already allocated: 2 4
Enter index starting block and length: 3 4
3-------->1
4 Block is already allocated
5-------->1
6-------->1
7-------->1
Do you want to enter more file(Yes - 1/No - 0)0
Name:Basanta Rai Roll:23473 sec:B 4th sem
```