Add the following functionalities in your program
a) Accept Available
b) Display Allocation, Max
c) Display the contents of need matrix
d) Display Available

code:

```
#include<stdio.h>
int nop,nor,A[10][10],M[10][10],Av[10],N[10][10],finish[10];
void acceptdata(int x[10][10])
{
int i,j;
for(i=0;i<nop;i++)
{
printf("P%d\n",i);
for(j=0;j<nor;j++)
{
printf("%c: ",65+j);
scanf("%d",&x[i][j]);
}
}
}
void acceptav()
{
int i;
for(i=0;i<nor;i++)
{
printf("%c: ",65+i);
scanf("%d",&Av[i]);
}
}
void calcneed()
{
int i,j;
for(i=0;i<nop;i++)
for(j=0;j<nor;j++)
N[i][j]=M[i][j]-A[i][j];
}
void displaydata()
{
int i,j;
printf("\n\tAllocation \t\tMax\t\tNeed\n\t");
for(i=0;i<3;i++)
{
for(j=0;j<nor;j++)
printf("%4c",65+j);
printf("\t");
}
for(i=0;i<nop;i++)
{
printf("\nP%d\t",i);
for(j=0;j<nor;j++)
printf("%4d",A[i][j]);
printf("\t");
for(j=0;j<nor;j++)
printf("%4d",M[i][j]);
printf("\t");
for(j=0;j<nor;j++)
printf("%4d",N[i][j]);
}
printf("\navailable");
for(i=0;i<nor;i++)
printf("%4d",Av[i]);
}
int checkneed(int pno)
{
int i;
for(i=0;i<nor;i++)
if(N[pno][i]>Av[i])
return 0;
return 1;
}
main()
{
printf("\nEnter No of Processes: ");
scanf("%d",&nop);
printf("\nEnter No. of Resources: ");
scanf("%d",&nor);
printf("\nEnter Allocation Matrix: ");
acceptdata(A);
printf("\nEnter Max Matrix: ");
acceptdata(M);
printf("\nEnter Availability:");
acceptav();
calcneed();
displaydata();
}
```

Add the following functionalities in your program
a) Accept Available
b) Display Allocation, Max
c) Display the contents of need matrix
d) Display Available
Implement Bankers Algorithm
code:

```
#include<stdio.h>
int nop,nor,A[10][10],M[10][10],Av[10],N[10][10],finish[10];
void acceptdata(int x[10][10])
{
int i,j;
for(i=0;i<nop;i++)
{
printf("P%d\n",i);
for(j=0;j<nor;j++)
{
printf("%c: ",65+j);
scanf("%d",&x[i][j]);
}
}
}
void acceptav()
{
int i;
for(i=0;i<nor;i++)
{
```

```
printf("%c: ",65+i);
scanf("%d",&Av[i]);
}
}
void calcneed()
{
int i,j;
for(i=0;i<nop;i++)
for(j=0;j<nor;j++)
N[i][j]=M[i][j]-A[i][j];
}
void displaydata()
{
int i,j;
printf("\n\tAllocation \t\tMax\t\tNeed\n\t");
for(i=0;i<3;i++)
{
for(j=0;j<nor;j++)
printf("%4c",65+j);
printf("\t");
}
for(i=0;i<nop;i++)
{
printf("\nP%d\t",i);
for(j=0;j<nor;j++)
printf("%4d",A[i][j]);
printf("\t");
for(j=0;j<nor;j++)
printf("%4d",M[i][j]);
printf("\t");
for(j=0;j<nor;j++)
printf("%4d",N[i][j]);
}
printf("\navailable");
for(i=0;i<nor;i++)
printf("%4d",Av[i]);
}
int checkneed(int pno)
{
int i;
for(i=0;i<nor;i++)
if(N[pno][i]>Av[i])
return 0;
return 1;
}
void banker()
{
int p=0,j=0,k=0,flag=0,safe[10];
while(flag<2)
{
if(!finish[p])
{
printf("\n\nNeed of process P%d (,",p);
for(j=0;j<nor;j++)
printf("%d,",N[p][j]);
if(checkneed(p))
{
```

```
printf(") <= available (");
for(j=0;j<nor;j++)
printf("%d,",Av[j]);
printf(")");
printf("\nNeed is Satsified, So process P%d can
be granted requiered resources.\n After P%d finishes, it will realease
all the resources.",p,p);
for(j=0;j<nor;j++)
Av[j]=Av[j]+A[p][j];
printf("New Availble=");
for(j=0;j<nor;j++)
printf("%d ",Av[j]);
finish[p]=1;
safe[k++]=p;
}
else
{
printf(") > available (");
for(j=0;j<nor;j++)
printf("%d,",Av[j]);
printf(")");
printf("\nNeed is not Satsified, So process P%d
cannot be granted required resources.\n process P%d has to
wait.",p,p);
}
}
if((p+1)%nop==0)
flag++;
p=(p+1)%nop;
}//while
if(k==nop)
{
printf("\nSystem is in safe state...");
printf("\nSafe Sequence: ");
for(j=0;j<k;j++)
printf("P%d->",safe[j]);
}
else
printf("\nSystem is not in safe state....");
}
main()
{
printf("\nEnter No of Processes: ");
scanf("%d",&nop);
printf("\nEnter No. of Resources: ");
scanf("%d",&nor);
printf("\nEnter Allocation Matrix: ");
acceptdata(A);
printf("\nEnter Max Matrix: ");
acceptdata(M);
printf("\nEnter Availability:");
acceptav();
calcneed();
displaydata();
banker();
}
```

Modify above program so as to include the following:
a) Accept Request for a process
b) Resource request algorithm
c) Safety algorithm Consider a system with ‘n’ processes and ‘m’ resource types.
Accept number of instances for every resource type. For each process accept the allocation and maximum requirement matrices. Write a program to display the contents of need matrix and to check if the given request of a process can be granted immediately or not.
Code:

```
#include<stdio.h>
int
nop,nor,Rprocess,A[10][10],M[10][10],Av[10],N[10][10],R[10],finish[10]
;
void acceptdata(int x[10][10])
{
int i,j;
for(i=0;i<nop;i++)
{
printf("P%d\n",i);
for(j=0;j<nor;j++)
{
printf("%c: ",65+j);
scanf("%d",&x[i][j]);
}
}
}
void acceptav()
{
int i;
for(i=0;i<nor;i++)
{
printf("%c: ",65+i);
scanf("%d",&Av[i]);
}
}
void acceptrequest()
{
int i;
printf("\nEnter the Process for which request has arrived :P");
scanf("%d",&Rprocess);
printf("\nEnter the request for process: ");
for(i=0;i<nor;i++)
{
printf("%c: ",65+i);
scanf("%d",&R[i]);
}
}
void calcneed()
{
int i,j;
for(i=0;i<nop;i++)
for(j=0;j<nor;j++)
N[i][j]=M[i][j]-A[i][j];
}
void displaydata()
```

```
{
int i,j;
printf("\n\tAllocation \t\tMax\t\tNeed\n\t");
for(i=0;i<3;i++)
{
for(j=0;j<nor;j++)
printf("%4c",65+j);
printf("\t");
}
for(i=0;i<nop;i++)
{
printf("\nP%d\t",i);
for(j=0;j<nor;j++)
printf("%4d",A[i][j]);
printf("\t");
for(j=0;j<nor;j++)
printf("%4d",M[i][j]);
printf("\t");
for(j=0;j<nor;j++)
printf("%4d",N[i][j]);
}
printf("\navailable");
for(i=0;i<nor;i++)
printf("%4d",Av[i]);
}
int checkneed(int pno)
{
int i;
for(i=0;i<nor;i++)
if(N[pno][i]>Av[i])
return 0;
return 1;
}
void resourcerequest()
{
int i;
for(i=0;i<nor;i++)
{
if(R[i]>N[Rprocess][i])
break;
}
if(i==nor)
{
for(i=0;i<nor;i++)
{
if(R[i]>Av[i])
break;
}
}
if(i==nor)
{
printf("\nRequest<=Need \n Request<=Available \n Both
Condition are true");
printf("\nThen system Pretends to fulfill request , then
modify resourse allocation state");
for(i=0;i<nor;i++)
{
```

```
Av[i]=Av[i]-R[i];
A[Rprocess][i]=A[Rprocess][i]+R[i];
N[Rprocess][i]=N[Rprocess][i]-R[i];
}
displaydata();
}
else
{
printf("\nRequest<=Need \n Request<=Available \n Condition
is not true");
printf("\nSo request cannot be satisfied!");
}
}
main()
{
printf("\nEnter No of Processes: ");
scanf("%d",&nop);
printf("\nEnter No. of Resources: ");
scanf("%d",&nor);
printf("\nEnter Allocation Matrix: ");
acceptdata(A);
printf("\nEnter Max Matrix: ");
acceptdata(M);
printf("\nEnter Availability:");
acceptav();
calcneed();
displaydata();
acceptrequest();
resourcerequest();
banker();
}
```

Resource Request and banker

```
#include<stdio.h>
int
nop,nor,Rprocess,A[10][10],M[10][10],Av[10],N[10][10],R[10],finish[10]
;
void acceptdata(int x[10][10])
{
int i,j;
for(i=0;i<nop;i++)
{
printf("P%d\n",i);
for(j=0;j<nor;j++)
{
printf("%c: ",65+j);
scanf("%d",&x[i][j]);
}
}
}
void acceptav()
{
int i;
for(i=0;i<nor;i++)
{
printf("%c: ",65+i);
scanf("%d",&Av[i]);
}
}
void acceptrequest()
{
int i;
printf("\nEnter the Process for which request has arrived :P");
scanf("%d",&Rprocess);
printf("\nEnter the request for process: ");
for(i=0;i<nor;i++)
{
printf("%c: ",65+i);
scanf("%d",&R[i]);
}
}
void calcneed()
{
int i,j;
for(i=0;i<nop;i++)
for(j=0;j<nor;j++)
N[i][j]=M[i][j]-A[i][j];
}
void displaydata()
{
int i,j;
printf("\n\tAllocation \t\tMax\t\tNeed\n\t");
for(i=0;i<3;i++)
{
for(j=0;j<nor;j++)
printf("%4c",65+j);
printf("\t");
}
for(i=0;i<nop;i++)
{
printf("\nP%d\t",i);
for(j=0;j<nor;j++)
printf("%4d",A[i][j]);
printf("\t");
for(j=0;j<nor;j++)
printf("%4d",M[i][j]);
printf("\t");
for(j=0;j<nor;j++)
printf("%4d",N[i][j]);
}
printf("\navailable");
for(i=0;i<nor;i++)
printf("%4d",Av[i]);
}
int checkneed(int pno)
{
int i;
for(i=0;i<nor;i++)
if(N[pno][i]>Av[i])
return 0;
return 1;
}
void resourcerequest()
```

```
{
int i;
for(i=0;i<nor;i++)
{
if(R[i]>N[Rprocess][i])
break;
}
if(i==nor)
{
for(i=0;i<nor;i++)
{
if(R[i]>Av[i])
break;
}
}
if(i==nor)
{
printf("\nRequest<=Need \n Request<=Available \n Both
Condition are true");
printf("\nThen system Pretends to fulfill request , then
modify resourse allocation state");
for(i=0;i<nor;i++)
{
Av[i]=Av[i]-R[i];
A[Rprocess][i]=A[Rprocess][i]+R[i];
N[Rprocess][i]=N[Rprocess][i]-R[i];
}
displaydata();
}
else
{
printf("\nRequest<=Need \n Request<=Available \n Condition
is not true");
printf("\nSo request cannot be satisfied!");
}
}
void banker()
{
int p=0,j=0,k=0,flag=0,safe[10];
while(flag<2)
{
if(!finish[p])
{
printf("\n\nNeed of process P%d (,",p);
for(j=0;j<nor;j++)
printf("%d,",N[p][j]);
if(checkneed(p))
{
printf(") <= available (");
for(j=0;j<nor;j++)
printf("%d,",Av[j]);
printf(")");
printf("\nNeed is Satsified, So process P%d can
be granted requiered resources.\n After P%d finishes, it will realease
all the resources.",p,p);
for(j=0;j<nor;j++)
Av[j]=Av[j]+A[p][j];
printf("New Availble=");
for(j=0;j<nor;j++)
printf("%d ",Av[j]);
finish[p]=1;
safe[k++]=p;
}
else
{
printf(") > available (");
for(j=0;j<nor;j++)
printf("%d,",Av[j]);
printf(")");
printf("\nNeed is not Satsified, So process P%d
cannot be granted required resources.\n process P%d has to
wait.",p,p);
}
}
if((p+1)%nop==0)
flag++;
p=(p+1)%nop;
}//while
if(k==nop)
{
printf("\nSystem is in safe state...");
printf("\nSafe Sequence: ");
for(j=0;j<k;j++)
printf("P%d->",safe[j]);
}
else
printf("\nSystem is not in safe state....");
}
main()
{
printf("\nEnter No of Processes: ");
scanf("%d",&nop);
printf("\nEnter No. of Resources: ");
scanf("%d",&nor);
printf("\nEnter Allocation Matrix: ");
acceptdata(A);
printf("\nEnter Max Matrix: ");
acceptdata(M);
printf("\nEnter Availability:");
acceptav();
calcneed();
displaydata();
acceptrequest();
resourcerequest();
banker();
}
```

Bankers and Menu driven programs above

File allocation

Write a program to simulate Sequential (Contiguous) file allocation method.Assume disk with n number of blocks. Give value of n as input. Write menu driver program with menu options as mentioned above and implement each option.

```
#include <stdio.h>
#include <stdlib.h>
void contiguous(int files[])
{
int flag = 0, startBlock, len, j, k, ch;
char fname[20];
printf("\nEnter Name of File: ");
scanf("%s",&fname);
printf("Enter the starting block of the files: ");
scanf("%d", &startBlock);
printf("Enter the length of the files: ");
scanf("%d", &len);
for (j = startBlock; j < (startBlock + len); j++)
{
if (files[j] == 0)
flag++;
}
if (len == flag)
{
for (k = startBlock; k < (startBlock + len); k++)
{
if (files[k] == 0)
{
files[k] = 1;
printf("%d\t%d\n", k, files[k]);
}
}
if (k != (startBlock + len - 1))
printf("The file is allocated to the disk\n");
}
else
printf("The file is not allocated to the disk\n");
printf("Do you want to enter more files?\n");
printf("Press 1 for YES, 0 for NO: ");
scanf("%d", &ch);
if (ch == 1)
contiguous(files);
else
exit(0);
return;
}
main()
{
int i,files[50];
for (i = 0; i < 50; i++)
files[i] = 0;
printf("Files Allocated are :\n");
contiguous(files);
return 0;
}
/*
[root@localhost osass2]# cc contiguous.c
[root@localhost osass2]# ./a.out
Files Allocated are :
Enter Name of File: a.txt
Enter the starting block of the files: 3
Enter the length of the files: 5
3 1
4 1
5 1
6 1
7 1
The file is allocated to the disk
Do you want to enter more files?
Press 1 for YES, 0 for NO: 1
Enter Name of File: b.txt
Enter the starting block of the files: 5
Enter the length of the files: 6
The file is not allocated to the disk
Do you want to enter more files?
Press 1 for YES, 0 for NO: 1
Enter Name of File: c.txt
Enter the starting block of the files: 12
Enter the length of the files: 4
12 1
13 1
14 1
15 1
The file is allocated to the disk
Do you want to enter more files?
Press 1 for YES, 0 for NO: 0
*/
```

Slot- II

Write a program to simulate Linked file allocation method. Assume disk with n number of blocks. Give value of n as input. Write menu driver program with menu options as mentioned above and implement each option.

```
#include <stdio.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 starting block: ");
scanf("%d",&st);
printf("Enter length of the file: ");
scanf("%d",&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);
return 0;
}
/*
[root@localhost osass2]# cc linked.c
[root@localhost osass2]# ./a.out
Enter how many blocks already allocated: 10
Enter blocks already allocated: 2
8
12
17
23
26
32
37
43
48
Enter starting block: 4
Enter length of the file: 5
4-------->1
5-------->1
6-------->1
7-------->1
8 Block is already allocated
9-------->1
Do you want to enter more file(Yes - 1/No - 0)1
Enter starting block: 23
Enter length of the file: 5
23 starting block is already allocated
Do you want to enter more file(Yes - 1/No - 0)1
Enter starting block: 34
Enter length of the file: 4
34-------->1
35-------->1
36-------->1
37 Block is already allocated
38-------->1
Do you want to enter more file(Yes - 1/No - 0)0
*/
```

Slot- III

Write a program to simulate Indexed file allocation method. Assume disk with n number of blocks. Give value of n as input. Write menu driver program with menu options as mentioned above and implement each option.

```
/* Program to simulate indexed file allocation strategy */

#include<stdio.h>
#include<stdlib.h>
main()
{
int f[50], index[50],i, n, st, len, j, c, k, ind,count=0;
for(i=0;i<50;i++)
f[i]=0;
x:printf("Enter the index block: ");
scanf("%d",&ind);
if(f[ind]!=1)
{
printf("Enter no of blocks needed for the index %d on the disk : \n", ind);
scanf("%d",&n);
}
else
{
printf("%d index is already allocated \n",ind);
goto x;
}
y: count=0;
printf("Enter blocks no for index %d on the disk : \n", ind);
for(i=0;i<n;i++)
{
scanf("%d", &index[i]);
if(f[index[i]]==0)
count++;
}
if(count==n)
{
for(j=0;j<n;j++)
f[index[j]]=1;
printf("Allocated\n");
printf("File Indexed\n");
for(k=0;k<n;k++)
printf("%d-------->%d : %d\n",ind,index[k],f[index[k]]);
}
else
{
printf("File in the index is already allocated \n");
printf("Enter another file indexed");
goto y;
}
printf("Do you want to enter more file(Yes - 1/No - 0)");
scanf("%d", &c);
if(c==1)
goto x;
else
exit(0);
getch();
}
/[root@localhost osass2]# cc indexed.c
[root@localhost osass2]# ./a.out
Enter the index block: 4
Enter no of blocks needed for the index 4 on the disk :
5
Enter blocks no for index 4 on the disk :
1
13
45
23
14
Allocated
File Indexed
```

```
4-------->1 : 1
4-------->13 : 1
4-------->45 : 1
4-------->23 : 1
4-------->14 : 1
Do you want to enter more file(Yes - 1/No - 0)1
Enter the index block: 13
13 index is already allocated
Enter the index block: 5
Enter no of blocks needed for the index 5 on the disk :
2
Enter blocks no for index 5 on the disk :
8
10
Allocated
File Indexed
5-------->8 : 1
5-------->10 : 1
Do you want to enter more file(Yes - 1/No - 0)0
*/
```

Disk Scheduling

1) Write a simulation program for disk scheduling using FCFS algorithm. Accept total number of disk blocks, disk request string, and current head position from the user. Display the list of request in the order in which it is served. Also display the total number of head moments.

```
#include<stdio.h>
int ReqString[20],nor,nob,start,thm;
main()
{
int i,j,k;
printf("\nEnter No.of Requests: ");
scanf("%d",&nor);
printf("\nEnter Requests:\n");
for(i=0;i<nor;i++)
{
printf("[%d]=",i);
scanf("%d",&ReqString[i]);
}
printf("\nEnter No.of Cylinders: ");
scanf("%d",&nob);
printf("\nEnter Start Block: ");
scanf("%d",&start);
for(i=0;i<nor;i++)
{
printf("\n%d-%d",start,ReqString[i]);
if(ReqString[i]>=start)
{
thm+=ReqString[i]-start;
start=ReqString[i];
}
else
{
thm+=start-ReqString[i];
start=ReqString[i];
}
}
printf("\nTotal Head Movement: %d",thm);
}
/*
Enter No.of Requests: 8
Enter Requests:
[0]=98
[1]=183
[2]=37
[3]=122
[4]=14
[5]=124
[6]=65
[7]=67
Enter No.of Cylinders: 200
Enter Start Block: 53
53-98
98-183
183-37
37-122
122-14
14-124
124-65
65-67
Total Head Movement: 640
*/
```

2) Write a simulation program for disk scheduling using SSTF algorithm. Accept total number of disk blocks, disk request string, and current head position from the user. Display the list of request in the order in which it is served. Also display the total number of head moments.

```
#include<stdio.h>
int ReqString[20],nor,nob,start,thm;
void sort(int RS[])
{
int i,j,temp;
for(i=1;i<nor;i++)
{
for(j=0;j<nor-i;j++)
{
if(RS[j]>RS[j+1])
{
temp=RS[j];
RS[j]=RS[j+1];
RS[j+1]=temp;
}
}
}
}
void display(int RS[])
{
int i;
for(i=0;i<nor;i++)
printf(" %d",RS[i]);
}
int searchstartblock(int RS[])
{
int i;
for(i=0;i<nor;i++)
{
if(RS[i]==start)
return i;
}
return 1;
}
main()
{
int
i,j,k,ans,finish=0,leftpos,startpos,rightpos,leftdis,rightdis,initpos
,flag=0;
printf("\nEnter No.of Requests: ");
scanf("%d",&nor);
printf("\nEnter Requests:\n");
for(i=0;i<nor;i++)
{
printf("[%d]=",i);
scanf("%d",&ReqString[i]);
}
```

```
printf("\nEnter No.of Cylinders: ");
scanf("%d",&nob);
printf("\nEnter Start Block: ");
scanf("%d",&start);
ans=searchstartblock(ReqString);
if(ans==1)
ReqString[nor++]=start;
sort(ReqString);
printf("\nRefString After Sorting: ");
display(ReqString);
startpos=searchstartblock(ReqString);
initpos=startpos;
leftpos=initpos-1;
rightpos=initpos+1;
while(1)
{
leftdis=ReqString[startpos]-ReqString[leftpos];
rightdis=ReqString[rightpos]-ReqString[startpos];
if(leftdis<rightdis)
{
printf("\n%d-%d",ReqString[startpos],ReqString[leftpos]);
thm+=ReqString[startpos]-ReqString[leftpos];
startpos=leftpos;
leftpos--;
if(leftpos==-1)
{
break;
}
}
else if(leftdis>rightdis)
{
printf("\n%d-%d",ReqString[rightpos],ReqString[startpos]);
thm+=ReqString[rightpos]-ReqString[startpos];
startpos=rightpos;
rightpos++;
if(rightpos==nor)
{
break;
}
}
}
printf("\nEnd of while loop");
while(leftpos>=0)
{
printf("\n%d-%d",ReqString[startpos],ReqString[leftpos]);
thm+=ReqString[startpos]-ReqString[leftpos];
startpos=leftpos;
leftpos--;
}
while(rightpos<nor)
{
printf("\n%d-%d",ReqString[rightpos],ReqString[startpos]);
thm+=ReqString[rightpos]-ReqString[startpos];
startpos=rightpos;
rightpos++;
}
printf("\nTotal Head Movement: %d",thm);
}
/*
[root@localhost disc sheduling]# cc sstf.c
[root@localhost disc sheduling]# ./a.out
Enter No.of Requests: 7
Enter Requests:
[0]=70
[1]=91
[2]=95
[3]=102
[4]=130
[5]=147
[6]=170
Enter No.of Cylinders: 200
Enter Start Block: 125
RefString After Sorting: 70 91 95 102 125 130 147 170
130-125
147-130
170-147
End of while loop
170-102
```

```
102-95
95-91
91-70
Total Head Movement: 145[root@localhost disc sheduling]# cc sstf.c
[root@localhost disc sheduling]# ./a.out
Enter No.of Requests: 8
Enter Requests:
[0]=11
[1]=34
[2]=62
[3]=64
[4]=95
[5]=119
[6]=123
[7]=180
Enter No.of Cylinders: 200
Enter Start Block: 50\
RefString After Sorting: 11 34 50 62 64 95 119 123 180
62-50
64-62
64-34
34-11
End of while loop
95-11
119-95
123-119
180-123
Total Head Movement: 236*/
```

3) Write a simulation program for disk scheduling using SCAN algorithm. Accept total number of disk blocks, disk request string, and current head position from the user. Display the list of request in the order in which it is served. Also display the total number of head moments.

```
//scan

#include<stdio.h>
int ReqString[20],nor,nob,start,thm,min[10],max[10];
char direction;
```

```
int getmin()
{
int i,j=0,min=999;
for(i=0;i<nor;i++)
if(ReqString[i]<=min)
min=ReqString[i];
return min;
}
int getmax()
{
int i,j=0,max=0;
for(i=0;i<nor;i++)
if(ReqString[i]>max)
max=ReqString[i];
return max;
}
main()
{
int i,j,k,max,min;
printf("\nEnter No.of Requests: ");
scanf("%d",&nor);
printf("\nEnter Requests:\n");
for(i=0;i<nor;i++)
{
printf("[%d]=",i);
scanf("%d",&ReqString[i]);
}
printf("\nEnter No.of Cylinders: ");
scanf("%d",&nob);
printf("\nEnter Start Block: ");
scanf("%d",&start);
printf("\nEnter Direction: ");
scanf(" %c",&direction);
min=getmin();
max=getmax();
if(direction=='L')
{
printf("\n%d-0",start);
thm+=start-0;
start=0;
printf("\n%d-%d",max,start);
thm+=max-start;
}
else if(direction=='R')
{
printf("\n%d-%d",start,nob-1);
thm+=(nob-1)-start;
start=nob-1;
printf("\n%d-%d",start,min);
thm+=start-min;
}
printf("\nTotal Head Movement: %d",thm);
}
/*
[root@localhost disc sheduling]# cc scan.c
[root@localhost disc sheduling]# ./a.out
Enter No.of Requests: 8
Enter Requests:
[0]=95
[1]=180
[2]=34
[3]=119
[4]=11
[5]=123
[6]=62
[7]=64
Enter No.of Cylinders: 200
Enter Start Block: 50
Enter Direction: L
50-0
180-0
Total Head Movement: 230
*/
```

4) Write a simulation program for disk scheduling using LOOK algorithm. Accept total number of disk blocks, disk request string, and current head position from the user. Display the list of request in the order in which it is served. Also display the total number of head moments.

```
//LOOK
#include<stdio.h>
int ReqString[20],nor,nob,start,thm,min[10],max[10];
char direction;
int getmin()
{
int i,j=0,min=999;
for(i=0;i<nor;i++)
if(ReqString[i]<=min)
min=ReqString[i];
return min;
}
int getmax()
{
int i,j=0,max=0;
for(i=0;i<nor;i++)
if(ReqString[i]>max)
max=ReqString[i];
return max;
}
main()
{
int i,j,k,max,min;
printf("\nEnter No.of Requests: ");
scanf("%d",&nor);
printf("\nEnter Requests:\n");
for(i=0;i<nor;i++)
{
printf("[%d]=",i);
scanf("%d",&ReqString[i]);
}
printf("\nEnter No.of Cylinders: ");
scanf("%d",&nob);
printf("\nEnter Start Block: ");
scanf("%d",&start);
printf("\nEnter Direction: ");
scanf(" %c",&direction);
min=getmin();
max=getmax();
if(direction=='L')
{
printf("\n%d-%d",start,min);
thm+=start-min;
start=min;
printf("\n%d-%d",max,start);
thm+=max-start;
}
else if(direction=='R')
{
printf("\n%d-%d",start,max);
thm+=max-start;
start=max;
printf("\n%d-%d",start,min);
thm+=start-min;
}
printf("\nTotal Head Movement: %d",thm);
}
/*
[root@localhost disc sheduling]# cc look.c
[root@localhost disc sheduling]# ./a.out
Enter No.of Requests: 8
Enter Requests:
[0]=95
[1]=180
[2]=34
[3]=119
[4]=11
[5]=123
[6]=62
[7]=64
Enter No.of Cylinders: 200
Enter Start Block: 50
Enter Direction: L
50-11
180-11
Total Head Movement: 208
*/
```

5) Write a simulation program for disk scheduling using C-SCAN algorithm. Accept total number of disk blocks, disk request string, and current head position from the user. Display the list of request in the order in which it is served. Also display the total number of head moments.

```
//cscan
#include<stdio.h>
int ReqString[20],nor,nob,start,thm,min[10],max[10];
char direction;
void sort(int RS[])
{
int i,j,temp;
for(i=1;i<nor;i++)
{
for(j=0;j<nor-i;j++)
{
if(RS[j]>RS[j+1])
{
temp=RS[j];
RS[j]=RS[j+1];
RS[j+1]=temp;
}
}
}
}
main()
{
int i,j,k,spos;
printf("\nEnter No.of Requests: ");
scanf("%d",&nor);
printf("\nEnter Requests:\n");
for(i=0;i<nor;i++)
{
printf("[%d]=",i);
scanf("%d",&ReqString[i]);
}
printf("\nEnter No.of Cylinders: ");
scanf("%d",&nob);
printf("\nEnter Start Block: ");
scanf("%d",&start);
ReqString[nor++]=start;
sort(ReqString);
for(i=0;i<nor;i++)
printf(" %d",ReqString[i]);
for(spos=0;spos<=nor && ReqString[spos]!=start; spos++);
printf("\nEnter Direction: ");
scanf(" %c",&direction);
if(direction=='L')
{
printf("\n%d-0",start);
thm+=start-0;
printf("\n%d-%d",nob-1,ReqString[spos+1]);
thm+=(nob-1)-ReqString[spos+1];
}
else if(direction=='R')
{
printf("\n%d-%d",nob-1,start);
thm+=(nob-1)-start;
printf("\n%d-0",ReqString[spos-1]);
thm+=ReqString[spos-1]-0;
}
printf("\nTotal Head Movement: %d",thm);
}
/*
[root@localhost disc sheduling]# cc cscan.c
[root@localhost disc sheduling]# ./a.out
Enter No.of Requests: 8
Enter Requests:
[0]=95
[1]=180
[2]=34
[3]=119
[4]=11
[5]=123
[6]=62
[7]=64
```

```
Enter No.of Cylinders: 200
Enter Start Block: 50
11 34 50 62 64 95 119 123 180
Enter Direction: L
50-0
199-62
Total Head Movement: 187
*/
```

6) Write a simulation program for disk scheduling using C-LOOK algorithm. Accept total number of disk blocks, disk request string, and current head position from the user. Display the list of request in the order in which it is served. Also display the total number of head moments.

```
#include<stdio.h>
int ReqString[20],nor,nob,start,thm,min[10],max[10];
char direction;
int getmin()
{
int i,j=0,min=999;
for(i=0;i<nor;i++)
if(ReqString[i]<=min)
min=ReqString[i];
return min;
}
int getmax()
{
int i,j=0,max=0;
for(i=0;i<nor;i++)
if(ReqString[i]>max)
max=ReqString[i];
return max;
}
void sort(int RS[])
{
int i,j,temp;
for(i=1;i<nor;i++)
{
for(j=0;j<nor-i;j++)
{
```

```
if(RS[j]>RS[j+1])
{
temp=RS[j];
RS[j]=RS[j+1];
RS[j+1]=temp;
}
}
}
}
main()
{
int i,j,k,max,min,spos;
printf("\nEnter No.of Requests: ");
scanf("%d",&nor);
printf("\nEnter Requests:\n");
for(i=0;i<nor;i++)
{
printf("[%d]=",i);
scanf("%d",&ReqString[i]);
}
printf("\nEnter No.of Cylinders: ");
scanf("%d",&nob);
printf("\nEnter Start Block: ");
scanf("%d",&start);
ReqString[nor++]=start;
sort(ReqString);
for(i=0;i<nor;i++)
printf(" %d",ReqString[i]);
for(spos=0;spos<=nor && ReqString[spos]!=start; spos++);
printf("\nEnter Direction: ");
scanf(" %c",&direction);
min=getmin();
max=getmax();
if(direction=='L')
{
printf("\n%d-%d",start,ReqString[0]);
thm+=start-ReqString[0];
printf("\n%d-%d",max,ReqString[spos+1]);
thm+=max-ReqString[spos+1];
```

```
}
else if(direction=='R')
{
printf("\n%d-%d",max,start);
thm+=max-start;
printf("\n%d-%d",ReqString[spos-1],min);
thm+=ReqString[spos-1]-min;
}
printf("\nTotal Head Movement: %d",thm);
}
/*
[root@localhost disc sheduling]# cc clook.c
[root@localhost disc sheduling]# ./a.out
Enter No.of Requests: 8
Enter Requests:
[0]=95
[1]=180
[2]=34
[3]=119
[4]=11
[5]=123
[6]=62
[7]=64
Enter No.of Cylinders: 200
Enter Start Block: 50
11 34 50 62 64 95 119 123 180
Enter Direction: R
180-50
34-11
Total Head Movement: 153
*/
```