//Banker's Algorithm Slot 2

#include<stdio.h>

#include<stdlib.h>

int ind,A[10][10],M[10][10],N[10][10],Av[10],Safe[10],Finish[10],nor,nop,work[10],req[10][10];

void AcceptData(int X[][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 AcceptAvailability()

{

int i;

for(i=0;i<nor;i++)

{

printf("%c",65+i);

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

work[i]=Av[i];

}

}

void DisplayData()

{

int i,j;

printf("\n\tAllocation\t\tMax\t\tNeed\n");

printf("\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\n");

for(j=0;j<nor;j++)

printf("%4d",work[j]);

}

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 Resource\_Request(int no)

{

int i,f11=0,f12=0;

for(i=0;i<nor;i++)

{

if(req[no][i]<=N[no][i])

f11=1;

else

f11=0;

}

if(f11==0)

{

printf("\n Error!Process has exceeded its maximum claim");

exit(0);

}

if(f11==1)

{

for(i=0;i<nor;i++)

{

if(req[no][i]<=work[i])

f12=1;

else

f12=0;

}

if(f12==0)

{

printf("\n Process has to wait for resources");

exit(0);

}

}

if(f11==1 && f12==1)

{

for(i=0;i<nor;i++)

{

work[i]=work[i]-req[no][i];

A[no][i]=A[no][i]+req[no][i];

N[no][i]=N[no][i]-req[no][i];

}

}

}

int checkNeed(int pno)

{

int i;

for(i=0;i<nor;i++)

if(N[pno][i]>work[i])

return(0);

return(1);

}

void Banker()

{

int i=0,j=0,k=0,flag=0;

while(flag<2)

{

if(!Finish[i])

{

printf("\nNeed%d(",i);

for(j=0;j<nor;j++)

printf("%d",N[i][j]);

if(!checkNeed(i))

{

printf("\b)>Work");

for(j=0;j<nor;j++)

printf("%d",work[j]);

printf("\b)");

printf("\nNeed Cannot be satisfied,consider next process");

}

else

{

printf("b)<=Work(");

for(j=0;j<nor;j++)

printf("%d,",work[j]);

printf("\b)");

printf("\nNeed can be satisfied,so allocate required resources");

printf("\nWork(%d)=",i);

for(j=0;j<nor;j++)

{

work[j]+=A[i][j];

}

for(j=0;j<nor;j++)

printf("%4d",work[j]);

printf("\nAfter P%d terminates it will release all its resources\n",i);

Safe[k++]=i;

Finish[i]=1;

}

}

if((i+1)%nop==0)

flag++;

i=(i+1)%nop;

}

if(k==nop)

{

printf("\nSystem is in safe state...");

printf("\nSafe Sequence:");

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

printf("P%d->",Safe[i]);

printf("\b\b");

}

else

{

printf("\nSystem is in not safe state...");

}

}

int main()

{

int i;

printf("\nEnter no of processes & No of Resources:");

scanf("%d%d",&nop,&nor);

printf("Enter Allocation\n");

AcceptData(A);

printf("Enter Max Requirement\n");

AcceptData(M);

printf("Enter Availability\n");

AcceptAvailability();

CalcNeed();

DisplayData();

Banker();

printf("\n Enter Process member from which request arrives:");

scanf("%d",&ind);

printf("\nEnter request for process%d\n",ind);

for(i=0;i<nor;i++)

{

printf("%c",65+i);

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

}

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

Finish[i]=0;

for(i=0;i<nor;i++)

work[i]=Av[i];

Resource\_Request(ind);

Banker();

return(0);

}

/\*output:

SETB Q1.-------------------

[ty@localhost ~]$ cc Resource1.c;

[ty@localhost ~]$ ./a.out

Enter no of processes & No of Resources:5 3

Enter Allocation

P0:

A:0

B:1

C:0

P1:

A:2

B:0

C:0

P2:

A:3

B:0

C:2

P3:

A:2

B:1

C:1

P4:

A:0

B:0

C:2

Enter Max Requirement

P0:

A:7

B:5

C:3

P1:

A:3

B:2

C:2

P2:

A:9

B:0

C:2

P3:

A:2

B:2

C:2

P4:

A:4

B:3

C:3

Enter Availability

A3

B3

C2

Allocation Max Need

A B C A B C A B C

P0 0 1 0 7 5 3 7 4 3

P1 2 0 0 3 2 2 1 2 2

P2 3 0 2 9 0 2 6 0 0

P3 2 1 1 2 2 2 0 1 1

P4 0 0 2 4 3 3 4 3 1

Available

3 3 2

Need0(74)>Work33)

Need Cannot be satisfied,consider next process

Need1(122b)<=Work(3,3,2)

Need can be satisfied,so allocate required resources

Work(1)= 5 3 2

After P1 terminates it will release all its resources

Need2(60)>Work53)

Need Cannot be satisfied,consider next process

Need3(011b)<=Work(5,3,2)

Need can be satisfied,so allocate required resources

Work(3)= 7 4 3

After P3 terminates it will release all its resources

Need4(431b)<=Work(7,4,3)

Need can be satisfied,so allocate required resources

Work(4)= 7 4 5

After P4 terminates it will release all its resources

Need0(743b)<=Work(7,4,5)

Need can be satisfied,so allocate required resources

Work(0)= 7 5 5

After P0 terminates it will release all its resources

Need2(600b)<=Work(7,5,5)

Need can be satisfied,so allocate required resources

Work(2)= 10 5 7

After P2 terminates it will release all its resources

System is in safe state...

Safe Sequence:P1->P3->P4->P0->P2->

Enter Process member from which request arrives:1

Enter request for process1

A1

B0

C2

Need0(74)>Work23)

Need Cannot be satisfied,consider next process

Need1(020b)<=Work(2,3,0)

Need can be satisfied,so allocate required resources

Work(1)= 5 3 2

After P1 terminates it will release all its resources

Need2(60)>Work53)

Need Cannot be satisfied,consider next process

Need3(011b)<=Work(5,3,2)

Need can be satisfied,so allocate required resources

Work(3)= 7 4 3

After P3 terminates it will release all its resources

Need4(431b)<=Work(7,4,3)

Need can be satisfied,so allocate required resources

Work(4)= 7 4 5

After P4 terminates it will release all its resources

Need0(743b)<=Work(7,4,5)

Need can be satisfied,so allocate required resources

Work(0)= 7 5 5

After P0 terminates it will release all its resources

Need2(600b)<=Work(7,5,5)

Need can be satisfied,so allocate required resources

Work(2)= 10 5 7

After P2 terminates it will release all its resources

System is in safe state...

Safe Sequence:P1->P3->P4->P0->P2[5309@localhost ~]$

\*/

/\*OUTPUT:SETB Q2.--------------

Enter no of processes & No of Resources: 5

4

Enter Allocation

P0:

A:0

B:0

C:1

D:2

P1:

A:1

B:0

C:0

D:0

P2:

A:1

B:3

C:5

D:4

P3:

A:0

B:6

C:3

D:2

P4:

A:0

B:0

C:1

D:4

Enter Max Requirement

P0:

A:0

B:0

C:1

D:2

P1:

A:1

B:7

C:5

D:0

P2:

A:2

B:3

C:5

D:6

P3:

A:0

B:6

C:5

D:2

P4:

A:0

B:6

C:5

D:6

Enter Availability

A1

B5

C2

D0

Allocation Max Need

A B C D A B C D A B C D

P0 0 0 1 2 0 0 1 2 0 0 0 0

P1 1 0 0 0 1 7 5 0 0 7 5 0

P2 1 3 5 4 2 3 5 6 1 0 0 2

P3 0 6 3 2 0 6 5 2 0 0 2 0

P4 0 0 1 4 0 6 5 6 0 6 4 2

Available

1 5 2 0

Need0(0000b)<=Work(1,5,2,0)

Need can be satisfied,so allocate required resources

Work(0)= 1 5 3 2

After P0 terminates it will release all its resources

Need1(075)>Work153)

Need Cannot be satisfied,consider next process

Need2(1002b)<=Work(1,5,3,2)

Need can be satisfied,so allocate required resources

Work(2)= 2 8 8 6

After P2 terminates it will release all its resources

Need3(0020b)<=Work(2,8,8,6)

Need can be satisfied,so allocate required resources

Work(3)= 2 14 11 8

After P3 terminates it will release all its resources

Need4(0642b)<=Work(2,14,11,8)

Need can be satisfied,so allocate required resources

Work(4)= 2 14 12 12

After P4 terminates it will release all its resources

Need1(0750b)<=Work(2,14,12,12)

Need can be satisfied,so allocate required resources

Work(1)= 3 14 12 12

After P1 terminates it will release all its resources

System is in safe state...

Safe Sequence:P0->P2->P3->P4->P1->

Enter Process member from which request arrives:1

Enter request for process1

A0

B4

C2

D0

Need0(0000b)<=Work(1,1,0,0)

Need can be satisfied,so allocate required resources

Work(0)= 1 1 1 2

After P0 terminates it will release all its resources

Need1(033)>Work111)

Need Cannot be satisfied,consider next process

Need2(1002b)<=Work(1,1,1,2)

Need can be satisfied,so allocate required resources

Work(2)= 2 4 6 6

After P2 terminates it will release all its resources

Need3(0020b)<=Work(2,4,6,6)

Need can be satisfied,so allocate required resources

Work(3)= 2 10 9 8

After P3 terminates it will release all its resources

Need4(0642b)<=Work(2,10,9,8)

Need can be satisfied,so allocate required resources

Work(4)= 2 10 10 12

After P4 terminates it will release all its resources

Need1(0330b)<=Work(2,10,10,12)

Need can be satisfied,so allocate required resources

Work(1)= 3 14 12 12

After P1 terminates it will release all its resources

System is in safe state...

Safe Sequence:P0->P2->P3->P4->P1

\*/

/\*OUTPUT:SETB Q3.------------------

Enter no of processes & No of Resources:5 3

Enter Allocation

P0:

A:0

B:1

C:0

P1:

A:2

B:0

C:0

P2:

A:3

B:0

C:3

P3:

A:2

B:1

C:1

P4:

A:0

B:0

C:2

Enter Max Requirement

P0:

A:0

B:1

C:0

P1:

A:4

B:0

C:2

P2:

A:3

B:0

C:3

P3:

A:3

B:1

C:1

P4:

A:0

B:0

C:1

Enter Availability

A0

B0

C0

Allocation Max Need

A B C A B C A B C

P0 0 1 0 0 1 0 0 0 0

P1 2 0 0 4 0 2 2 0 2

P2 3 0 3 3 0 3 0 0 0

P3 2 1 1 3 1 1 1 0 0

P4 0 0 2 0 0 1 0 0 -1

Available

0 0 0

Need0(000b)<=Work(0,0,0)

Need can be satisfied,so allocate required resources

Work(0)= 0 1 0

After P0 terminates it will release all its resources

Need1(20)>Work01)

Need Cannot be satisfied,consider next process

Need2(000b)<=Work(0,1,0)

Need can be satisfied,so allocate required resources

Work(2)= 3 1 3

After P2 terminates it will release all its resources

Need3(100b)<=Work(3,1,3)

Need can be satisfied,so allocate required resources

Work(3)= 5 2 4

After P3 terminates it will release all its resources

Need4(00-1b)<=Work(5,2,4)

Need can be satisfied,so allocate required resources

Work(4)= 5 2 6

After P4 terminates it will release all its resources

Need1(202b)<=Work(5,2,6)

Need can be satisfied,so allocate required resources

Work(1)= 7 2 6

After P1 terminates it will release all its resources

System is in safe state...

Safe Sequence:P0->P2->P3->P4->P1->

Enter Process member from which request arrives:4

Enter request for process4

A0

B0

C1

Error!Process has exceeded its maximum claim.

\*/