#include<stdio.h>

typedef struct

{

int allocated[5];

int need[5];

int max[5];

}proces;

proces process[10];

int available[5];

int no\_p,no\_r;

int safeseq[10];

int rsc[5];

int temp[5];

void cmptAvailable();

void cmptNeed();

int runSafeSeq();

int inSeq(int i,int n);

void readData();

void display();

void requestRsc();

void main()

{

int t;

readData();

display();

t=runSafeSeq();

requestRsc();

}

void readData()

{

printf("Enter no. of process : ");

scanf("%d",&no\_p);

printf("Enter no. of resources : ");

scanf("%d",&no\_r);

int i,j;

printf("Enter number of instances of each resource\n");

for(i=0;i<no\_r;i++)

{

printf("R%d :",i+1);

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

}

for(i=0;i<no\_p;i++)

{

printf("Enter maximum requirement for process P%d :",i);

for(j=0;j<no\_r;j++)

{

scanf("%d",&process[i].max[j]);

}

}

for(i=0;i<no\_p;i++)

{

printf("Enter Allocated Resources for process P%d :",i);

for(j=0;j<no\_r;j++)

{

scanf("%d",&process[i].allocated[j]);

}

}

cmptNeed();

cmptAvailable();

}

void display()

{

int i,j,k=0;

printf(" \t ALLOC\t\t MAX \t\t NEED \t\t AVAIL\n");

char str[12];

for(j=0;j<no\_r;j++)

{

str[k++]='R';

str[k++]=j+1+'0';

str[k++]=' ';

}

for(j=k;j<12;j++)

{

str[j]=' ';

}

printf(" \t%s\t%s\t%s\t%s\n",str,str,str,str);

for(i=0;i<no\_p;i++)

{

printf("P%d\t",i);

for(j=0;j<no\_r;j++)

{

printf("%d ",process[i].allocated[j]);

}

printf(" \t");

for(j=0;j<no\_r;j++)

{

printf("%d ",process[i].max[j]);

}

printf(" \t");

for(j=0;j<no\_r;j++)

{

printf("%d ",process[i].need[j]);

}

printf(" \t");

if(i==0)

{

for(j=0;j<no\_r;j++)

{

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

}

}

printf("\n");

}

}

void cmptNeed()

{

int i,j;

for(i=0;i<no\_p;i++)

{

for(j=0;j<no\_r;j++)

{

process[i].need[j]=process[i].max[j]-process[i].allocated[j];

}

}

}

void cmptAvailable()

{

int i,j,count=0;

for(j=0;j<no\_r;j++)

{

for(i=0;i<no\_p;i++)

{

count+=process[i].allocated[j];

}

available[j]=rsc[j]-count;

count=0;

}

}

int runSafeSeq()

{

int i=0,j,k=0,l=0,flag=1;

for(i=0;i<no\_r;i++)

{

temp[i]=available[i];

}

for(i=0;i<no\_p;i++)

{

safeseq[i]=0;

}

i=0;

while(1)

{

flag=1;

if(inSeq(i,k))

{

i++;

if(i==no\_p)

{

i=0;

}

continue;

}

for(j=0;j<no\_r;j++)

{

if(process[i].need[j]>available[j])

{

flag=0;

}

}

if(flag==1)

{

l=0;

safeseq[k++]=i;

for(j=0;j<no\_r;j++)

{

available[j]+=process[i].allocated[j];

}

}

if(flag==0)

{

l++;

}

if(k==no\_p)

{

printf("system is in safe state\nThe safe sequence is ");

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

{

printf("P%d ",safeseq[j]);

}

printf("\n");

for(j=0;j<no\_r;j++)

{

available[j]=temp[j];

}

return 1;

}

if(l==no\_p-k)

{

printf("system is not in safe state\n");

for(j=0;j<no\_r;j++)

{

available[j]=temp[j];

}

return 0;

}

i++;

if(i==no\_p)

{

i=0;

}

}

}

int inSeq(int i,int n)

{

int j;

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

{

if(safeseq[j]==i)

return 1;

}

return 0;

}

void requestRsc()

{

int a,b[no\_r],i,j,count=0;

int flag=0,temp1[no\_r],temp2[no\_r];

printf("Enter Process that makes request : ");

scanf("%d",&a);

printf("Enter the request :");

for(j=0;j<no\_r;j++)

{

scanf("%d",&b[j]);

temp1[j]=process[a].allocated[j];

temp2[j]=available[j];

}

for(j=0;j<no\_r;j++)

{

if(b[j]>available[j])

flag=1;

if(process[a].allocated[j]+b[j]>process[a].max[j])

flag=1;

}

if(flag==1)

{

printf("The request made by process P%d cannot be granted immediately\n",a);

return;

}

for(j=0;j<no\_r;j++)

{

process[a].need[j]-=b[j];

available[j]=available[j]-b[j];

process[a].allocated[j]=process[a].allocated[j]+b[j];

}

display();

if(runSafeSeq()==1)

{

printf("The request made by process P%d can be granted immediately\n",a);

}

else

{

printf("The request made by process P%d cannott be granted immediately\n",a);

for(j=0;j<no\_r;j++)

{

process[a].allocated[j]=temp1[j];

available[j]=temp2[j];

}

}

}

/\*

PS F:\SEM4\OS> gcc -o k Assign7.c

PS F:\SEM4\OS> ./k

Enter no. of process : 3

Enter no. of resources : 3

Enter number of instances of each resource

R1 :8

R2 :6

R3 :7

Enter maximum requirement for process P0 :5 4 3

Enter maximum requirement for process P1 :1 2 2

Enter maximum requirement for process P2 :3 2 1

Enter Allocated Resources for process P0 :3 2 1

Enter Allocated Resources for process P1 :1 0 0

Enter Allocated Resources for process P2 :2 1 1

ALLOC MAX NEED AVAIL

R1 R2 R3 R1 R2 R3 R1 R2 R3 R1 R2 R3

P0 3 2 1 5 4 3 2 2 2 2 3 5

P1 1 0 0 1 2 2 0 2 2

P2 2 1 1 3 2 1 1 1 0

system is in safe state

The safe sequence is P0 P1 P2

Enter Process that makes request : 1

Enter the request :0 2 2

ALLOC MAX NEED AVAIL

R1 R2 R3 R1 R2 R3 R1 R2 R3 R1 R2 R3

P0 3 2 1 5 4 3 2 2 2 2 1 3

P1 1 2 2 1 2 2 0 0 0

P2 2 1 1 3 2 1 1 1 0

system is in safe state

The safe sequence is P1 P2 P0

The request made by process P1 can be granted immediately

\*/