#include<stdio.h>

#include<stdlib.h>

typedef struct

{

int max[10];

int need[10];

int allocation[10];

}proces;

int available[10],temp[10];

proces \*p[20];

int np,nr;

int seq[10];

void computeNeed();

void findSafeSeq();

int notinseq(int i,int count);

void findSafeSeq();

void requestalgo();

void main()

{

int i,j;

printf("Enter the no. of proceses:");

scanf("%d",&np);

printf("Enter the no. of resource types:");

scanf("%d",&nr);

for(i=1;i<=np;i++)

{

p[i]=(proces\*)malloc(sizeof(proces));

}

for(i=1;i<=np;i++)

{

printf("\nEnter the details of proces %d:\n",i);

printf("Max:\n");

for(j=1;j<=nr;j++)

{

printf("%d:",j);

scanf("%d",&p[i]->max[j]);

}

printf("\nAllocation:\n");;

for(j=1;j<=nr;j++)

{

printf("%d:",j);

scanf("%d",&p[i]->allocation[j]);

}

}

computeNeed();

findSafeSeq();

requestalgo();

}

void computeNeed()

{

int i,j;

for(i=1;i<=np;i++)

{

for(j=1;j<=nr;j++)

{

p[i]->need[j]=p[i]->max[j]-p[i]->allocation[j];

}

}

}

int notinseq(int i,int count)

{

int j;

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

{

if(i==seq[j])

return 0;

}

return 1;

}

void findSafeSeq()

{

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

printf("Enter available resources:\n");

for(i=1;i<=nr;i++)

{

printf("%d:",i);

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

Available[i]=available[i];

}

i=1;

while(count<np)

{

for(j=1;j<=nr;j++)

{

if(p[i]->need[j]>available[j])

{

flag=1;

break;

}

}

if(flag==0 && notinseq(i,count)==1)

{

for(j=1;j<=nr;j++)

{

available[j]+=p[i]->allocation[j];

}

seq[count++]=i;

}

i++;

if(i>np)

{

i=1;

}

flag=0;

}

printf("The safe sequence is:\n");

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

{

printf("p%d ",seq[i]);

}

}

void requestalgo()

{

int pid,i,j,request[10],count=0,flag=0;

int noOfItr=1;

for(i=0;i<10;i++)

{

seq[i]=0;

}

printf("\nEnter the proces no. which would like to request resources:");

scanf("%d",&pid);

printf("Enter the instances:\n");

for(j=1;j<=nr;j++)

{

printf("%d:",j);

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

}

for(j=1;j<=nr;j++)

{

if(request[j]>p[pid]->need[j])

{

printf("\nThe request cannot be granted immediately.\n");

exit(0);

}

if(request[j]>Available[j])

{

printf("\nThe request cannot be granted immediately.\n");

exit(0);

}

}

for(j=1;j<=nr;j++)

{

Available[j]-=request[j];

p[pid]->allocation[j]+=request[j];

p[pid]->need[j]-=request[j];

}

i=1;

while(count<np && noOfItr<=np)

{

for(j=1;j<=nr;j++)

{

if(p[i]->need[j]>Available[j])

{

flag=1;

break;

}

}

if(flag==0 && notinseq(i,count)==1)

{

for(j=1;j<=nr;j++)

{

Available[j]+=p[i]->allocation[j];

}

seq[count++]=i;

}

i++;

if(i>np)

{

i=1;

noOfItr++;

}

flag=0;

}

if(count==np)

{

printf("The safe sequence is:\n");

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

{

printf("p%d ",seq[i]);

}

printf("\nproces p%d's request can be granted immediately.\n",pid);

}

else

{

printf("\nThe request cannot be granted immediately.\n");

}

}