**Group : A**

**Assignment No. : 9**

**Title : Write a program to implement Bankers Algorithm**

**Roll No. : 2365**

**-------------------------------------------------------------------------------------------**

import java.io.\*;

import java.util.\*;

class Bankers

{

Scanner sc=new Scanner(System.in);

int noofprocesses=0,noofresources=0;

int rinstances[];

int maxreq[][];

int allocated[][];

int need[][];

int free[]=new int[10];

int totalallocated[]=new int[10];

int total=0;

void accept()

{

System.out.print("\nEnter no of processes::");

noofprocesses=sc.nextInt();

System.out.print("Enter no of resources::");

noofresources=sc.nextInt();

rinstances=new int[noofresources];

maxreq=new int[noofprocesses+1][noofresources+1];

allocated=new int[noofprocesses+1][noofresources+1];

System.out.print("\nEnter no. of Resources");

for(int i=0;i<noofresources;i++)

{

System.out.print("\nResource "+(i)+" ::");

rinstances[i]=sc.nextInt();

}

System.out.print("\nEnter Maximum Requirement\n");

for(int i=0;i<noofprocesses;i++)

{

System.out.print("\nProcess "+(i)+" ::\n");

for(int j=0;j<noofresources;j++)

{

System.out.print("Resource "+(j)+" ::");

maxreq[i][j]=sc.nextInt();

}

}

System.out.print("\nEnter no. of Allocated Resources\n");

for(int i=0;i<noofprocesses;i++)

{

System.out.print("\nProcess "+(i)+" ::\n");

for(int j=0;j<noofresources;j++)

{

System.out.print("Resource "+(j)+" ::");

allocated[i][j]=sc.nextInt();

}

}

}

void Bankers()

{

need=new int[noofprocesses][noofresources];

for(int i=0;i<noofprocesses;i++)

{

for(int j=0;j<noofresources;j++)

{

need[i][j]=maxreq[i][j]-allocated[i][j];

}

}

for(int i=0;i<noofresources;i++)

{

total=0;

for(int j=0;j<noofprocesses;j++)

{

total=total+allocated[j][i];

}

totalallocated[i]=total;

free[i]=rinstances[i]-total;

}

}

void display()

{

int safeseq[]=new int[noofprocesses];

System.out.println("\tMax\tAllocated\tNeed(Max-Allocated)");

for(int j=0;j<noofprocesses;j++)

{

System.out.print("\nP"+j);

System.out.print("\t");

for(int k=0;k<noofresources;k++)

{

System.out.print(maxreq[j][k]);

}

System.out.print("\t");

for(int k=0;k<noofresources;k++)

{

System.out.print(allocated[j][k]);

}

System.out.print("\t\t");

for(int k=0;k<noofresources;k++)

{

System.out.print(need[j][k]);

}

System.out.print("\t");

}

System.out.println("\nTotal allocated resources: ");

for (int i = 0; i <noofresources; i++) {

System.out.print(totalallocated[i]+"\t");

}

System.out.println("\nTotal free resources: ");

for (int i = 0; i <noofresources; i++) {

System.out.print(free[i]+"\t");

}

int cnt=0,selected=-1;

int flag[]=new int[10];

System.out.print("\nSafe sequence::");

while(cnt!=noofprocesses)

{

for(int i=0;i<noofprocesses;i++)

{

for(int j=0;j<noofresources;j++)

{

if(free[j]>=need[i][j] && flag[i]==0)

{

selected=i;

}

else

{

selected=-1;

break;

}

}

if(selected==-1)

{

if(flag[i]!=1)

System.out.println("\nP"+i+"Rejected");

}

else

{

safeseq[cnt]=selected;

System.out.println("\nP"+i+"Selected\n");

flag[selected]=1;

cnt++;

for(int k=0;k<noofresources;k++)

{

free[k]=free[k]-need[selected][k];

//System.out.print("Free R "+k+" :"+free[k]+"\t");

}

//System.out.println("\nAfter execution of P"+selected+"\n");

for(int l=0;l<noofresources;l++)

{

free[l]=free[l]+maxreq[selected][l];

//System.out.print("Free R "+l+" :"+free[l]+"\t");

}

}

}

}

System.out.print("\n::Safe Sequence::\n");

for (int i = 0; i < safeseq.length; i++) {

System.out.print("P"+safeseq[i]+"->");

}

/\*

//Another safe seq

Bankers();

for (int i = 0; i < flag.length; i++) {

flag[i]=0;

}

System.out.print("\nSecond safe sequence::");

cnt=0;

while(cnt!=noofprocesses)

{

for(int i=0;i<noofprocesses;i++)

{

for(int j=0;j<noofresources;j++)

{

if(free[j]>=need[i][j] && flag[i]==0)

{

selected=i;

}

else

{

selected=-1;

break;

}

}

if(selected==-1)

{

if(flag[i]!=1)

System.out.println("\nP"+i+"Rejected");

}

else

{

safeseq[cnt]=selected;

System.out.println("\nP"+i+"Selected\n");

flag[selected]=1;

cnt++;

for(int k=0;k<noofresources;k++)

{

free[k]=free[k]-need[selected][k];

System.out.print("Free R "+k+" :"+free[k]+"\t");

}

System.out.println("\nAfter execution of P"+selected+"\n");

for(int l=0;l<noofresources;l++)

{

free[l]=free[l]+maxreq[selected][l];

System.out.print("Free R "+l+" :"+free[l]+"\t");

}

break;

}

}

}

System.out.print("\n::Safe Sequence::\n");

for (int i = 0; i < safeseq.length; i++) {

System.out.print("P"+safeseq[i]+"->");

}

\*/

}

}

public class BankersA

{

public static void main(String[] args) {

// TODO Auto-generated method stub

Bankers bankalgo=new Bankers();

bankalgo.accept();

bankalgo.Bankers();

bankalgo.display();

}

}

/\*

\*

Enter no of processes::4

Enter no of resources::3

Enter no. of Resources

Resource 0 ::9

Resource 1 ::3

Resource 2 ::6

Enter Maximum Requirement

Process 0 ::

Resource 0 ::3

Resource 1 ::2

Resource 2 ::2

Process 1 ::

Resource 0 ::6

Resource 1 ::1

Resource 2 ::3

Process 2 ::

Resource 0 ::3

Resource 1 ::1

Resource 2 ::4

Process 3 ::

Resource 0 ::4

Resource 1 ::2

Resource 2 ::2

Enter no. of Allocated Resources

Process 0 ::

Resource 0 ::1

Resource 1 ::0

Resource 2 ::0

Process 1 ::

Resource 0 ::6

Resource 1 ::1

Resource 2 ::2

Process 2 ::

Resource 0 ::2

Resource 1 ::1

Resource 2 ::1

Process 3 ::

Resource 0 ::0

Resource 1 ::0

Resource 2 ::2

Max Allocated Need(Max-Allocated)

P0 322 100 222

P1 613 612 001

P2 314 211 103

P3 422 002 420

Total allocated resources:

9 2 5

Total free resources:

0 1 1

Safe sequence::

P0Rejected

P1Selected

P2Selected

P3Selected

P0Selected

::Safe Sequence::

P1->P2->P3->P0->\*/