PROGRAM:

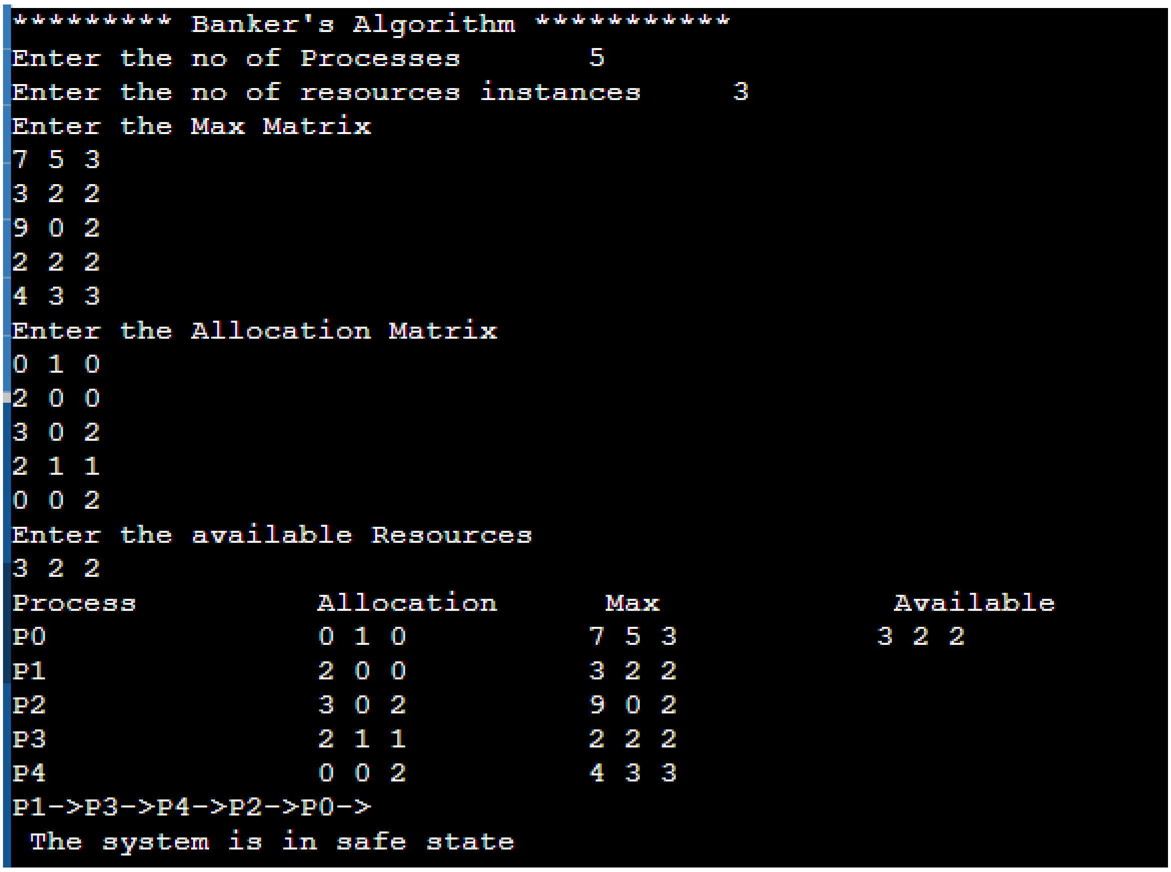
```
#include<stdio.h>
#include<conio.h>
int max[100][100];
int alloc[100][100];
int need[100][100];
int avail[100];
int n,r;
void show();
void cal();
int main()
{
int i,j;
printf("********* Banker's Algorithm ***********\n");
printf("Enter the no of Processes\t");
scanf("%d",&n);
printf("Enter the no of resources instances\t");
scanf("%d",&r);
printf("Enter the Max Matrix\n");
for(i=0;i<n;i++)
{
for(j=0;j<r;j++)
{
scanf("%d",&max[i][j]);
}
}
printf("Enter the Allocation Matrix\n");
for(i=0;i<n;i++)
{
for(j=0;j<r;j++)
{
scanf("%d",&alloc[i][j]);
}
}
printf("Enter the available Resources\n");
for(j=0;j<r;j++)
{
scanf("%d",&avail[j]);
}
show();
cal();
```

```
getch();
return 0;
}

void show()
{
int i,j;
printf("Process\t\t Allocation\t Max\t\t Available\t");
for(i=0;i<n;i++)
{
printf("\nP%d\t\t ",i);
for(j=0;j<r;j++)
{
printf("%d ",alloc[i][j]);
}
printf("\t\t");
for(j=0;j<r;j++)
{
printf("%d ",max[i][j]);
}
printf("\t\t");
if(i==0)
{
for(j=0;j<r;j++)
printf("%d ",avail[j]);
}
}
}
void cal()
{
int finish[100], temp, need[100][100],flag=1,k,c1=0;
int safe[100];
int i,j;
for(i=0;i<n;i++)
{
finish[i]=0;
}
//find need matrix
for(i=0;i<n;i++)
{
for(j=0;j<r;j++)
```

```
{
need[i][j] = max[i][j] -alloc[i][j];
}
}
printf("\n");
while(flag)
{
flag=0;
for(i=0;i<n;i++)
{
int c=0;
for(j=0;j<r;j++)
{
if((finish[i]==0)&&(need[i][j]<=avail[j]))
{
c++;
if(c==r)
{
for(k=0;k<r;k++)
{
avail[k]+=alloc[i][j];
finish[i]=1;
flag=1;
}
printf("P%d->",i);
if(finish[i]==1)
{
i=n;
}
}
}
}
}
for(i=0;i<n;i++)
{
if(finish[i]==1)
{
c1++;
}
else
{
printf("P%d->",i);
}
}
if(c1==n)
{
printf("\n The system is in safe state");
}
else
{
printf("\n Process are in dead lock");
printf("\n System is in unsafe state");
}
}
```

OUTPUT:



```
********* Banker's Algorithm ***********
Enter the no of Processes       5
Enter the no of resources instances     3
Enter the Max Matrix
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter the Allocation Matrix
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter the available Resources
3 2 2
Process          Allocation      Max             Available
P0               0 1 0          7 5 3           3 2 2
P1               2 0 0          3 2 2
P2               3 0 2          9 0 2
P3               2 1 1          2 2 2
P4               0 0 2          4 3 3
P1->P3->P4->P2->P0->
 The system is in safe state
```