Source Code

```
#include <stdio.h>
#include <conio.h>
void main() {
        int
k=0,a=0,b=0,instance[5],availability[5],allocated[10][5],need[10][5],MAX[
10][5],process,P[10],no_of_resources, cnt=0,i, j;
        printf("\n Enter the number of resources : ");
        scanf("%d", &no_of_resources);
        printf("\n enter the max instances of each resources\n");
        for (i=0;i<no_of_resources;i++) {
                availability[i]=0;
                printf("%c= ",(i+97));
                scanf("%d",&instance[i]);
        }
        printf("\n Enter the number of processes : ");
        scanf("%d", &process);
        printf("\n Enter the allocation matrix \n     ");
        for (i=0;i<no_of_resources;i++)
        printf(" %c",(i+97));
        printf("\n");
        for (i=0;i <process;i++) {
                P[i]=i;
                printf("P[%d]  ",P[i]);
                for (j=0;j<no_of_resources;j++) {
                        scanf("%d",&allocated[i][j]);
                        availability[j]+=allocated[i][j];
                }
        }
        printf("\nEnter the MAX matrix \n     ");
        for (i=0;i<no_of_resources;i++) {
                printf(" %c",(i+97));
                availability[i]=instance[i]-availability[i];
        }
        printf("\n");
        for (i=0;i <process;i++) {
                printf("P[%d]  ",i);
                for (j=0;j<no_of_resources;j++)
                 scanf("%d", &MAX[i][j]);
        }
        printf("\n");
        A: a=-1;
        for (i=0;i <process;i++) {
                cnt=0;
                b=P[i];
                for (j=0;j<no_of_resources;j++) {
                        need[b][j] = MAX[b][j]-allocated[b][j];
                        if(need[b][j]<=availability[j])
                         cnt++;
                }
                if(cnt==no_of_resources) {
                        op[k++]=P[i];
                        for (j=0;j<no_of_resources;j++)
                        availability[j]+=allocated[b][j];
                } else
                 P[++a]=P[i];
        }
```

Output

```
Enter the number of resources : 3

Enter the max instances of each resource
a= 10
b= 5
c= 7

 Enter the number of processes: 5

 Enter the allocation matrix
      a b c
P[0]  0 1 0
P[1]  2 0 0
P[2]  3 0 2
P[3]  2 1 1
P[4]  0 0 2

Enter the MAX matrix
      a b c
P[0]  7 5 3
P[1]  3 2 2
P[2]  9 0 2
P[3]  4 2 2
P[4]  5 3 3
< P[1]  P[3]  P[4]  P[0]  P[2] >
```

Challenging Question

A student majoring in anthropology and minoring in Computer Science has embarked on a research project to see if African Baboons can be taught about deadlocks. He locates a deep canyon and fastens a rope across it, so the baboons can cross hand-over-hand. Several baboons can cross at the same time, provided that they are all going in the same direction. If eastward-moving and westward-moving baboons ever get onto the rope at the same time, a deadlock will result (the baboons will be stuck in the middle) because it is impossible for one baboon to climb over another one while suspended over the canyon. If a baboon wants to cross the canyon, it must check to see that no other baboon is currently crossing in the opposite direction. Write a program using semaphores that avoid deadlock.