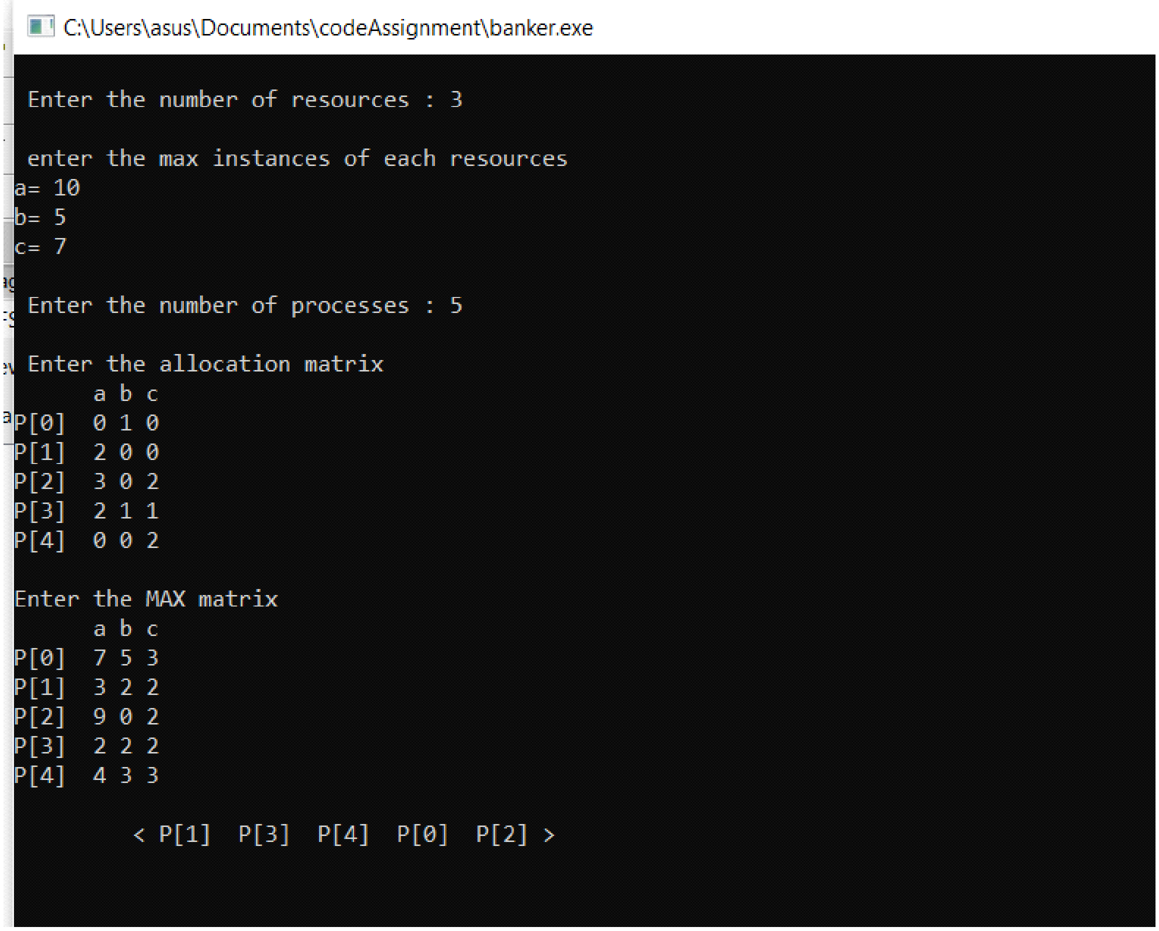
SOMESHWAR JHA (IT-C)

BANKER'S ALGORITHM

```
#include<stdio.h>
#include<conio.h>
void main() {
	int
k=0,output[10],d=0,t=0,ins[5],i,avail[5],allocated[10][5],need[10][5],MAX[10][5],pno,P[10],j,rz, count=0;
	printf("\n Enter the number of resources : ");
	scanf("%d", &rz);
	printf("\n enter the max instances of each resources\n");
	for (i=0;i<rz;i++) {
		avail[i]=0;
		printf("%c= ",(i+97));
		scanf("%d",&ins[i]);
	}
	printf("\n Enter the number of processes : ");
	scanf("%d", &pno);
	printf("\n Enter the allocation matrix \n     ");
	for (i=0;i<rz;i++)
	printf(" %c",(i+97));
	printf("\n");
	for (i=0;i <pno;i++) {
		P[i]=i;
		printf("P[%d]  ",P[i]);
		for (j=0;j<rz;j++) {
			scanf("%d",&allocated[i][j]);
			avail[j]+=allocated[i][j];
		}
	}
	printf("\nEnter the MAX matrix \n     ");
	for (i=0;i<rz;i++) {
		printf(" %c",(i+97));
		avail[i]=ins[i]-avail[i];
	}
	printf("\n");
	for (i=0;i <pno;i++) {
		printf("P[%d]  ",i);
		for (j=0;j<rz;j++)
		 scanf("%d", &MAX[i][j]);
	}
	printf("\n");
	A: d=-1;
```

```
        for (i=0;i <pno;i++) {
                count=0;
                t=P[i];
                for (j=0;j<rz;j++) {
                        need[t][j] = MAX[t][j]-allocated[t][j];
                        if(need[t][j]<=avail[j])
                         count++;
                }
                if(count==rz) {
                        output[k++]=P[i];
                        for (j=0;j<rz;j++)
                        avail[j]+=allocated[t][j];
                } else
                 P[++d]=P[i];
        }
        if(d!=-1) {
                pno=d+1;
                goto A;
        }
        printf("\t <");
        for (i=0;i<k;i++)
        printf(" P[%d] ",output[i]);
        printf(">");
        getch();
}
```

OUTPUT-



```
 Enter the number of resources : 3

 enter the max instances of each resources
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]  2 2 2
P[4]  4 3 3

         < P[1]  P[3]  P[4]  P[0]  P[2] >
```