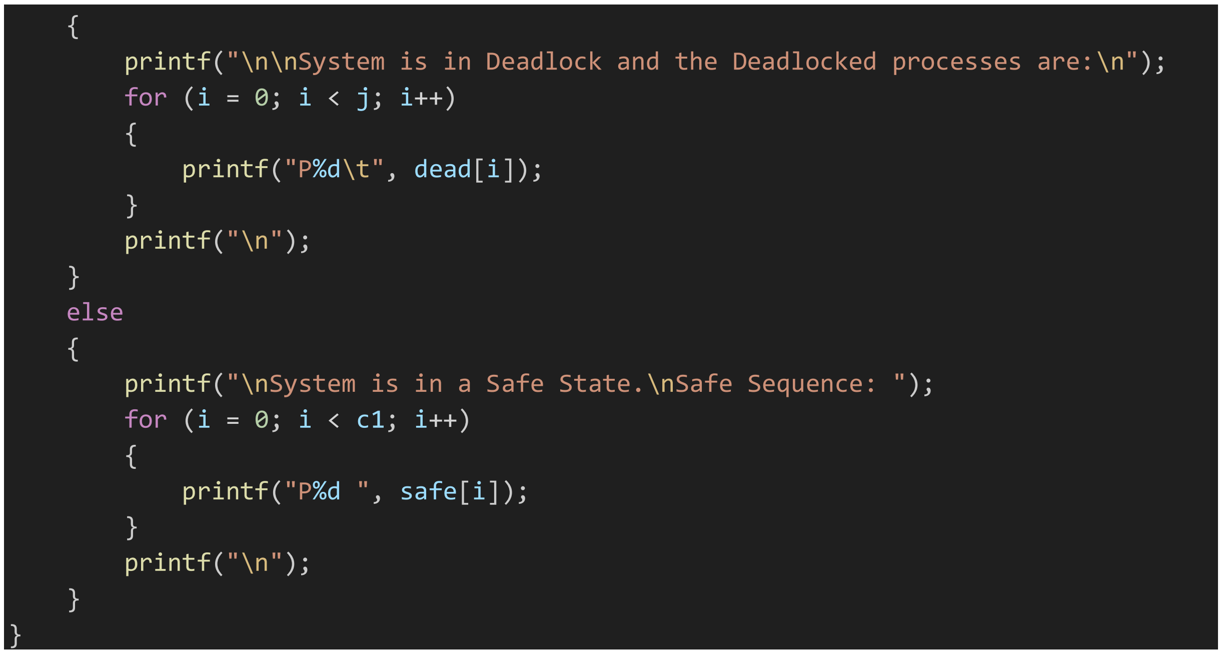
LAB-08

Exercise:

1) Implement the above code and paste the screen shot of the output.

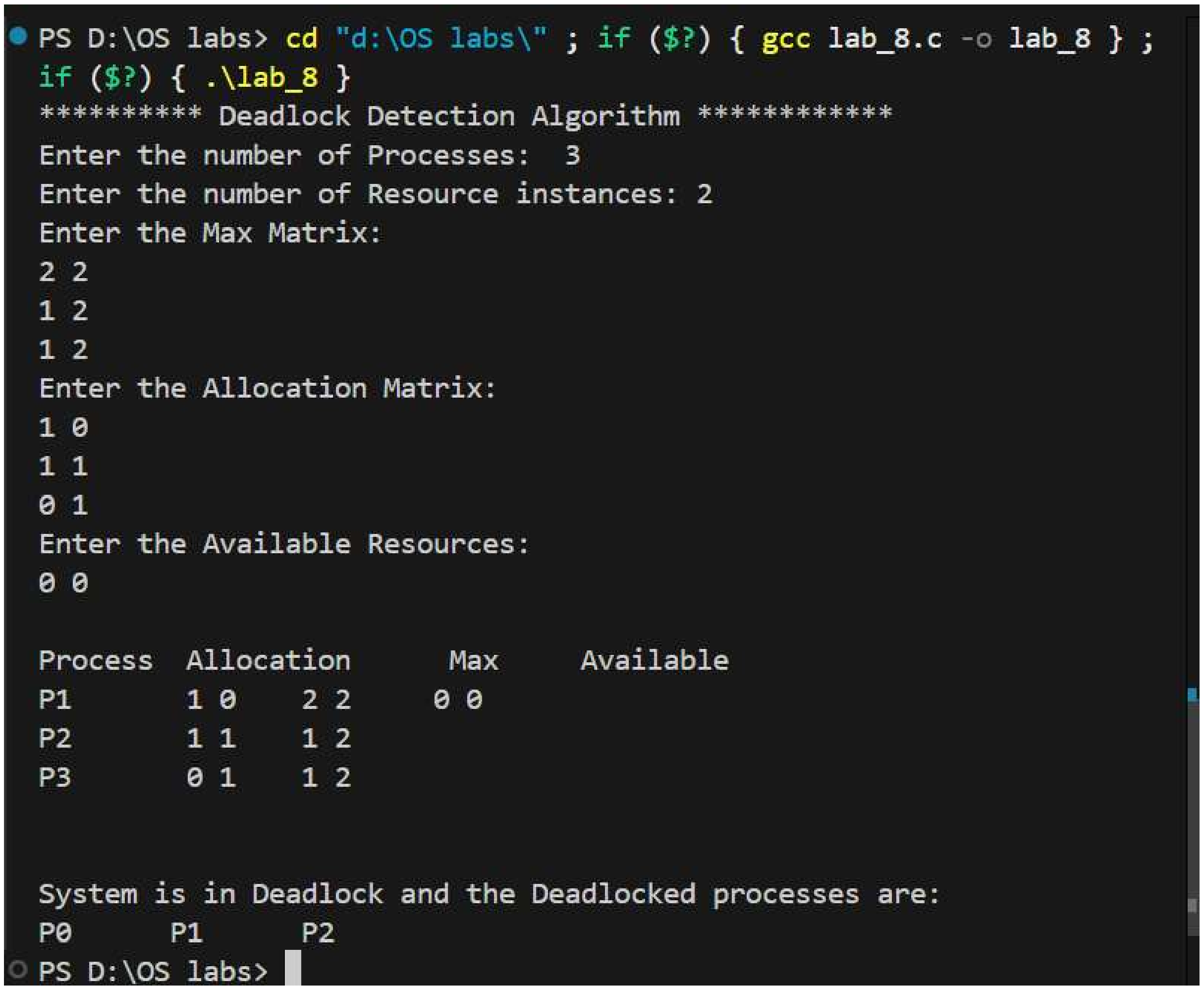
PROGRAM:

```c
#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 input();
void show();
void cal();
int main()
{
    int i, j;
    printf("********** Deadlock Detection Algorithm ************\n");
    input();
    show();
    cal();
    getch();
    return 0;
}
void input()
{
    int i, j;
    printf("Enter the number of Processes:\t");
    scanf("%d", &n);

    printf("Enter the number of Resource 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]);
    }
}
void show()
{
    int i, j;
    printf("\nProcess\t Allocation\t Max\t Available\n");
    for (i = 0; i < n; i++)
    {
        printf("P%d\t ", i + 1);

        for (j = 0; j < r; j++)
        {
            printf("%d ", alloc[i][j]);
        }

        printf("\t");

        for (j = 0; j < r; j++)
        {
            printf("%d ", max[i][j]);
        }

        printf("\t");

        if (i == 0)
        {
            for (j = 0; j < r; j++)
            {
                printf("%d ", avail[j]);
            }
        }

        printf("\n");
    }
}
void cal()
{
    int finish[100], temp, flag = 1, k, c1 = 0;
    int dead[100], safe[100];
    int i, j;
    for (i = 0; i < n; i++)
    {
        finish[i] = 0;
    }
    // Calculate the need matrix
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < r; j++)
        {
            need[i][j] = max[i][j] - alloc[i][j];
        }
    }
    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 && finish[i] == 0)
            {
                for (k = 0; k < r; k++)
                {
                    avail[k] += alloc[i][k];
                }

                finish[i] = 1;
                flag = 1;
                safe[c1++] = i;
            }
        }
    }
    j = 0;
    flag = 0;
    for (i = 0; i < n; i++)
    {
        if (finish[i] == 0)
        {
            dead[j++] = i;
            flag = 1;
        }
    }
    if (flag == 1)
```



```
    {
        printf("\n\nSystem is in Deadlock and the Deadlocked processes are:\n");
        for (i = 0; i < j; i++)
        {
            printf("P%d\t", dead[i]);
        }
        printf("\n");
    }
    else
    {
        printf("\nSystem is in a Safe State.\nSafe Sequence: ");
        for (i = 0; i < c1; i++)
        {
            printf("P%d ", safe[i]);
        }
        printf("\n");
    }
}
```

OUTPUT:



```
PS D:\OS labs> cd "d:\OS labs\" ; if ($?) { gcc lab_8.c -o lab_8 } ;
if ($?) { .\lab_8 }
********** Deadlock Detection Algorithm ************
Enter the number of Processes:  3
Enter the number of Resource instances: 2
Enter the Max Matrix:
2 2
1 2
1 2
Enter the Allocation Matrix:
1 0
1 1
0 1
Enter the Available Resources:
0 0

Process  Allocation      Max     Available
P1       1 0    2 2     0 0
P2       1 1    1 2
P3       0 1    1 2


System is in Deadlock and the Deadlocked processes are:
P0      P1      P2
PS D:\OS labs>
```