Muhammad Saria **DT - 22036**

LAB # 8

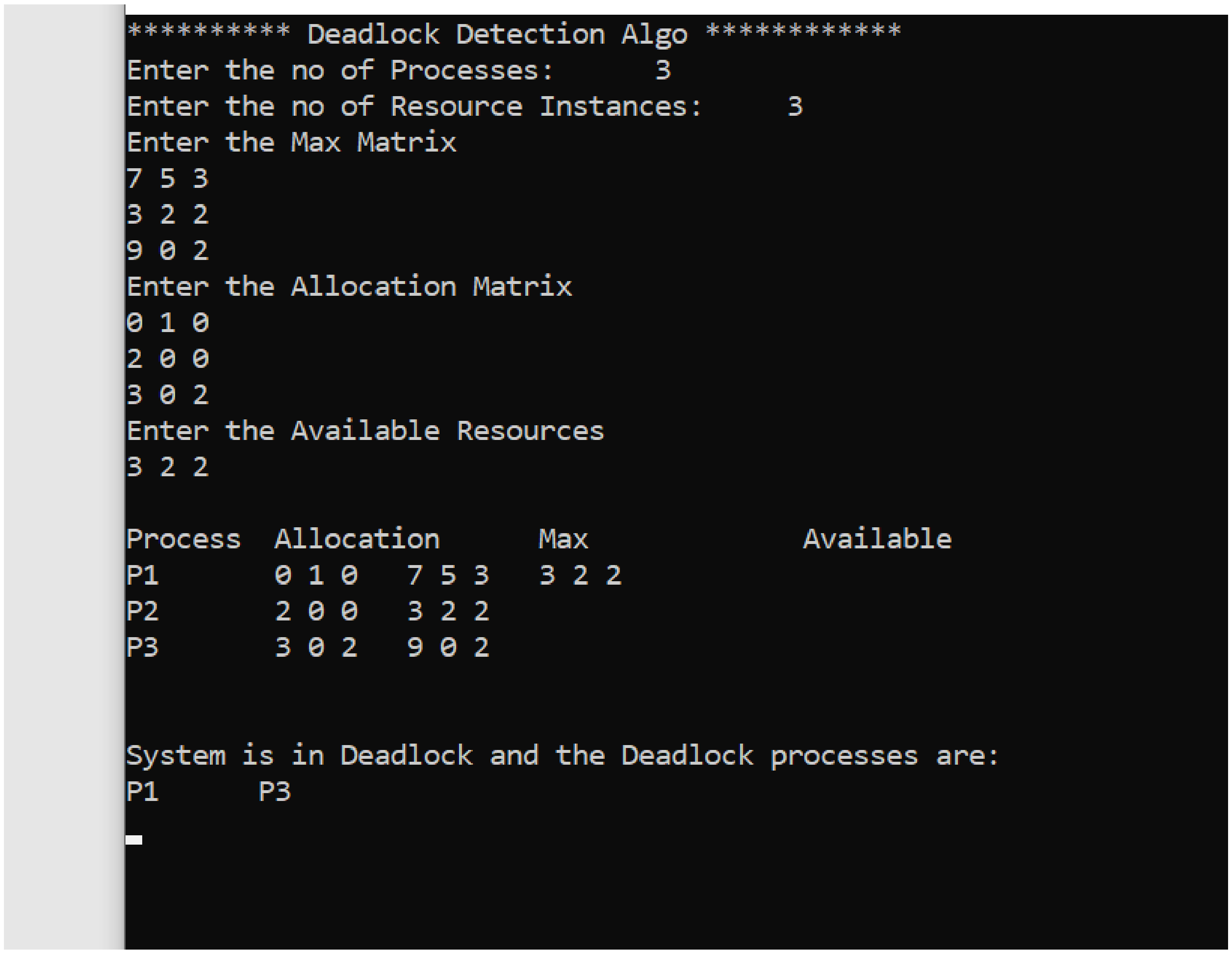
Q) Implement the above code and paste the screen shot of the output.

```
#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()
{
    printf("********** Deadlock Detection Algo ************\n");
    input();
    show();
    cal();
    getch();
    return 0;
}

void input()
{
```

```
int i, j;
printf("Enter the no of Processes:\t");
scanf("%d", &n);
printf("Enter the no 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\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], dead[100], i, j, k, flag = 1;

    // Initialize finish to 0
    for(i = 0; i < n; i++)
        finish[i] = 0;

    // Calculate need matrix
    for(i = 0; i < n; i++)
        for(j = 0; j < r; j++)
            need[i][j] = max[i][j] - alloc[i][j];

    // Begin detection algorithm
    while(flag)
    {
        flag = 0;
        for(i = 0; i < n; i++)
        {
            if(finish[i] == 0)
            {
                int canAllocate = 1;
                for(j = 0; j < r; j++)
                {
                    if(need[i][j] > avail[j])
                    {
                        canAllocate = 0;
                        break;
                    }
                }

                if(canAllocate)
                {
                    for(k = 0; k < r; k++)
                        avail[k] += alloc[i][k];

                    finish[i] = 1;
                    flag = 1;
                }
            }
        }
    }

    int deadCount = 0;
    for(i = 0; i < n; i++)
    {
        if(finish[i] == 0)
        {
            dead[deadCount++] = i;
        }
    }

    if(deadCount > 0)
    {
        printf("\n\nSystem is in Deadlock and the Deadlock processes are:\n");
```

```
        for(i = 0; i < deadCount; i++)
            printf("P%d\t", dead[i] + 1);
        printf("\n");
    }
    else
    {
        printf("\n\nNo Deadlock is detected. System is in a safe state.\n");
    }
}
```



```
********** Deadlock Detection Algo ************
Enter the no of Processes:      3
Enter the no of Resource Instances:     3
Enter the Max Matrix
7 5 3
3 2 2
9 0 2
Enter the Allocation Matrix
0 1 0
2 0 0
3 0 2
Enter the Available Resources
3 2 2

Process  Allocation      Max             Available
P1       0 1 0   7 5 3   3 2 2
P2       2 0 0   3 2 2
P3       3 0 2   9 0 2


System is in Deadlock and the Deadlock processes are:
P1      P3
```