Shayan
DT-22037

OS LAB 08

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

Solution:

```
#include <stdio.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 Algorithm ************\n");
    input();
    show();
    cal();
    return 0;
}

void input()
{
    int i, j;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    printf("Enter the number of resource types: ");
    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]);
    }

    // Calculate need matrix
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < r; j++)
        {
            need[i][j] = max[i][j] - alloc[i][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]);
        }
        if (i == 0)
        {
            printf("\t ");
            for (j = 0; j < r; j++)
            {
                printf("%d ", avail[j]);
            }
        }
```

```
        printf("\n");
    }
}

void cal()
{
    int finish[100], dead[100], i, j, k;
    int flag, count = 0;

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

    while (1)
    {
        flag = 0;

        for (i = 0; i < n; i++)
        {
            if (finish[i] == 0)
            {
                int can_allocate = 1;
                for (j = 0; j < r; j++)
                {
                    if (need[i][j] > avail[j])
                    {
                        can_allocate = 0;
                        break;
                    }
                }

                if (can_allocate)
                {
                    for (j = 0; j < r; j++)
                    {
                        avail[j] += alloc[i][j];
                    }
                    finish[i] = 1;
                    flag = 1;
                    break;
                }
            }
        }

        if (flag == 0)
```

```
            break;
    }

    // Check for deadlock
    int deadlock = 0;
    printf("\n\n");
    for (i = 0; i < n; i++)
    {
        if (finish[i] == 0)
        {
            deadlock = 1;
            dead[count++] = i;
        }
    }

    if (deadlock)
    {
        printf("System is in Deadlock.\nDeadlocked processes are: ");
        for (i = 0; i < count; i++)
        {
            printf("P%d ", dead[i]);
        }
        printf("\n");
    }
    else
    {
        printf("No Deadlock Detected. System is in safe state.\n");
    }
}
```

Shayan
DT-22037

```
Enter the number of processes: 5
Enter the number of resource types: 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:
0 0 0

Process  Allocation     Max            Available
P1       0 1 0   7 5 3   0 0 0
P2       2 0 0   3 2 2
P3       3 0 2   9 0 2
P4       2 1 1   2 2 2
P5       0 0 2   4 3 3


System is in Deadlock.
Deadlocked processes are: P0 P1 P2 P3 P4
```