Lab-8

```
#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 instances: ");
   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\tAllocation\tMax\t\tAvailable\n");
    for (i = 0; i < n; i++)
    {
        printf("P%d\t", i);
        for (j = 0; j < r; j++)
        {
            printf("%d ", alloc[i][j]);
        }
        printf("\t\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];
    int i, j, k, c1 = 0, flag = 1, count = 0;

    // Initialize finish array and calculate need
    for (i = 0; i < n; i++)
    {
        finish[i] = 0;
        for (j = 0; j < r; j++)
        {
            need[i][j] = max[i][j] - alloc[i][j];
        }
```

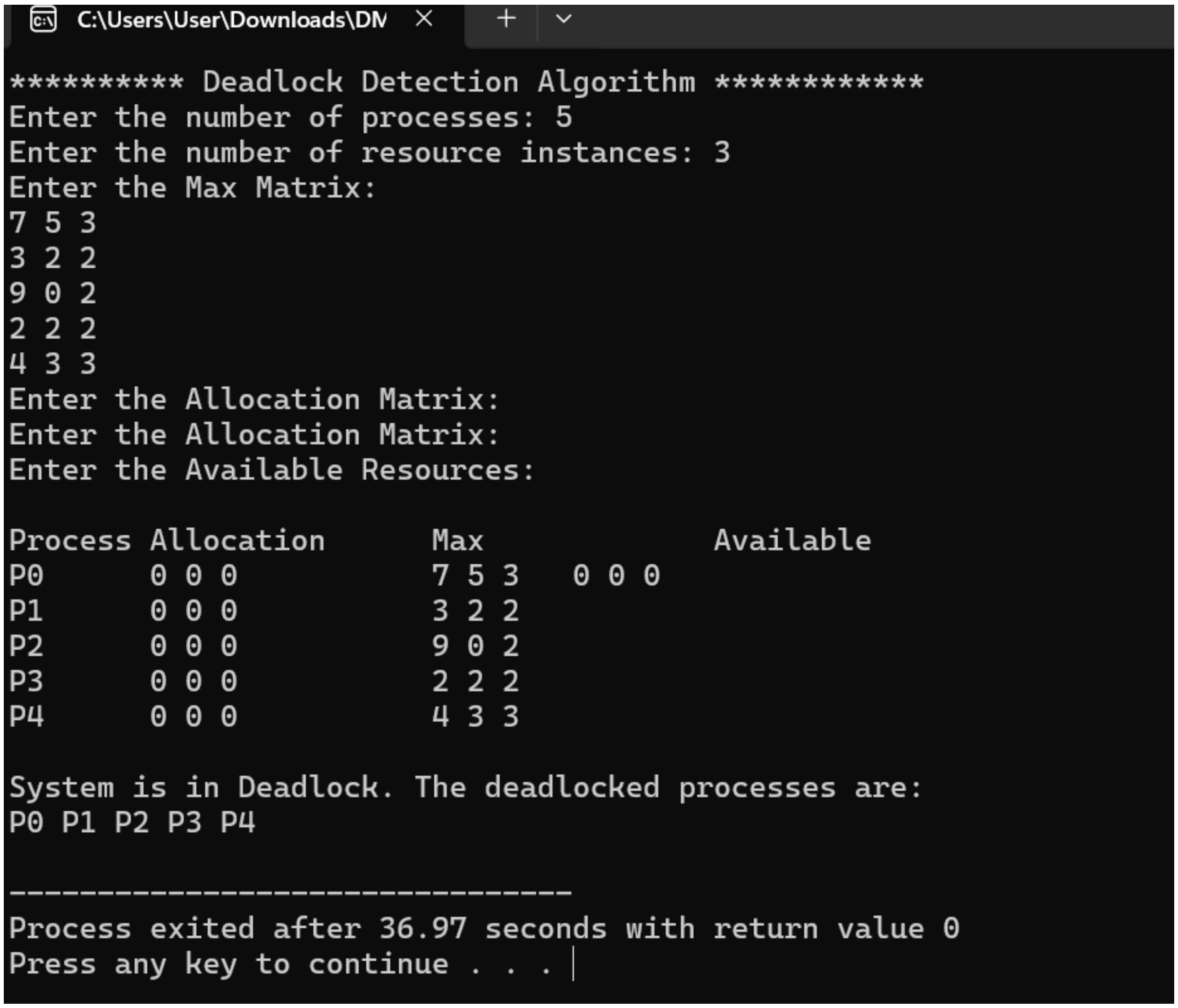
```
}

while (flag)
{
   flag = 0;
   for (i = 0; i < n; i++)
   {
      if (finish[i] == 0)
      {
         int exec = 1;
         for (j = 0; j < r; j++)
         {
            if (need[i][j] > avail[j])
            {
               exec = 0;
               break;
            }
         }
         if (exec)
         {
            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("\nSystem is in Deadlock. The deadlocked processes are:\n");
   for (i = 0; i < deadCount; i++)
   {
      printf("P%d ", dead[i]);
   }
   printf("\n");
```

```
    }
    else
    {
       printf("\nNo Deadlock Detected. System is in a safe state.\n");
    }
}
```



```
C:\Users\User\Downloads\DM

********** Deadlock Detection Algorithm ************
Enter the number of processes: 5
Enter the number of resource instances: 3
Enter the Max Matrix:
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter the Allocation Matrix:
Enter the Allocation Matrix:
Enter the Available Resources:

Process Allocation      Max             Available
P0      0 0 0           7 5 3   0 0 0
P1      0 0 0           3 2 2
P2      0 0 0           9 0 2
P3      0 0 0           2 2 2
P4      0 0 0           4 3 3

System is in Deadlock. The deadlocked processes are:
P0 P1 P2 P3 P4

--------------------------------
Process exited after 36.97 seconds with return value 0
Press any key to continue . . .
```