ASSIGNMENT - VII

TITLE :
Implement the Banker's algorithm to avoid deadlock by analysing resource allocation and safely granting resource requests.

NAME : Shinde Shubham Dnyandev, **DIV :** SY-B, **ROLL NO. :** 23107121.

PROGRAM :

```
#include<stdio.h>
int allocation[10][10], max_need[10][10], remaining[10][10];
int available[10];
int NP,NR;

void readmatrix(int matrix[10][10])
{
   int i,j;
   for (i=0; i<NP; i++)
   for (j=0; j<NR; j++)
   scanf("%d", &matrix[i][j]);
}

void display_matrix(int matrix[10][10])
{
   int i,j;
   for (i=0; i<NP; i++)
   {
      printf("\n P%d : ",i);
      for(j=0; j<NR; j++)
      {
         printf("%d", matrix[i][j]);
      }
   }
}

void calculate_remaining()
{
   int i,j;
   for (i=0; i<NP; i++)
   for (j=0; j<NR; j++)
   remaining[i][j] = max_need[i][j] - allocation[i][j];
}
```

```
void banker()
{
   int i,j,k = 0, flag;
   int finish[10], safe_seq[10];
   for (i=0; i<NP; i++)
   {
      finish[i] = 0;
   }
   for (i=0; i<NP; i++)
   {
      flag = 0;
      if(finish[i] == 0)
      {
         for(j=0; j<NR; j++)
         {
            if(remaining[i][j] > available[j])
            {
               flag = 1;
               break;
            }
         }
         if(flag == 0)
         {
            finish[i] = 1;
            safe_seq[k] = i;
            k++;
            for(j=0; j<NR; j++)
            available[j] += allocation[i][j];
            i = -1;
         }
      }
   }
   flag = 0;
   for(i=0; i<NP; i++)
   {
      if(finish[i] == 0)
      {
         printf("\nThe system is in Deadlock");
         flag = 1;
         break;
      }
   }
   if(flag == 0)
   {
      printf("\nThe system is in Safe State\n Safe Sequence is ==> ");
```

```
        for(i=0; i<NP; i++)
        printf("P%d->", safe_seq[i]);
    }
}

int main()
{
    int j;

    {
        printf("\nEnter no of Processes : ");
        scanf("%d",&NP);

        printf("\nEnter no of Resources : ");
        scanf("%d",&NR);

        printf("\nEnter Initial Allocation matrix : \n");
        readmatrix(allocation);

        printf("\nEnter Maximum Need matrix : \n");
        readmatrix(max_need);
    }

    {
        printf("\nEntered Data is : ");
        printf("\nInitial Allocation : ");
        display_matrix(allocation);

        printf("\nMaximum need : ");
        display_matrix(max_need);

        printf("\n\nEnter Available Resources : \n");
        for(j=0; j<NR; j++)
        scanf("%d", &available[j]);
    }

    {
        calculate_remaining();
        printf("\n Remaining Nodes are : ");
        display_matrix(remaining);
    }
    banker();
    printf("\n");
    return 0;
}
```

OUTPUT :

```
shubham@ShubhsPC:~$ gcc banker.c
shubham@ShubhsPC:~$ ./a.out

 Enter no of Processes : 4

 Enter no of Resources : 3

 Enter Initial Allocation matrix :
 0 1 0
 2 0 0
 3 0 2
 2 1 0

 Enter Maximum Need matrix :
 7 5 3
 3 2 2
 7 0 2
 2 2 2

 Entered Data is :
 Initial Allocation :
  P0 : 010
  P1 : 200
  P2 : 302
  P3 : 210
 Maximum need :
  P0 : 753
  P1 : 322
  P2 : 702
  P3 : 222

 Enter Available Resources :
 3
 3
 2

  Remaining Nodes are :
  P0 : 743
  P1 : 122
  P2 : 400
  P3 : 012
 The system is in Safe State
  Safe Sequence is ==> P1->P2->P3->P0->
```