# Experiment No. 6

Name   : Rinoy Kuriyakose
Roll No: 56

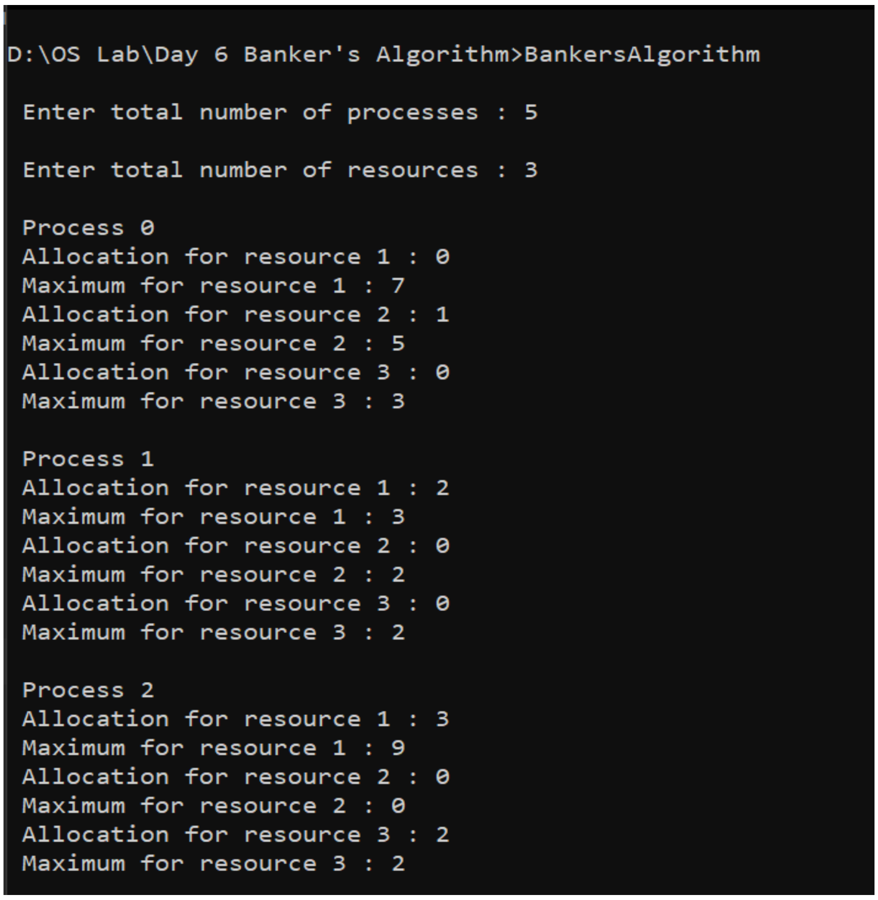
**Aim:**

Implement Banker’s Algorithm

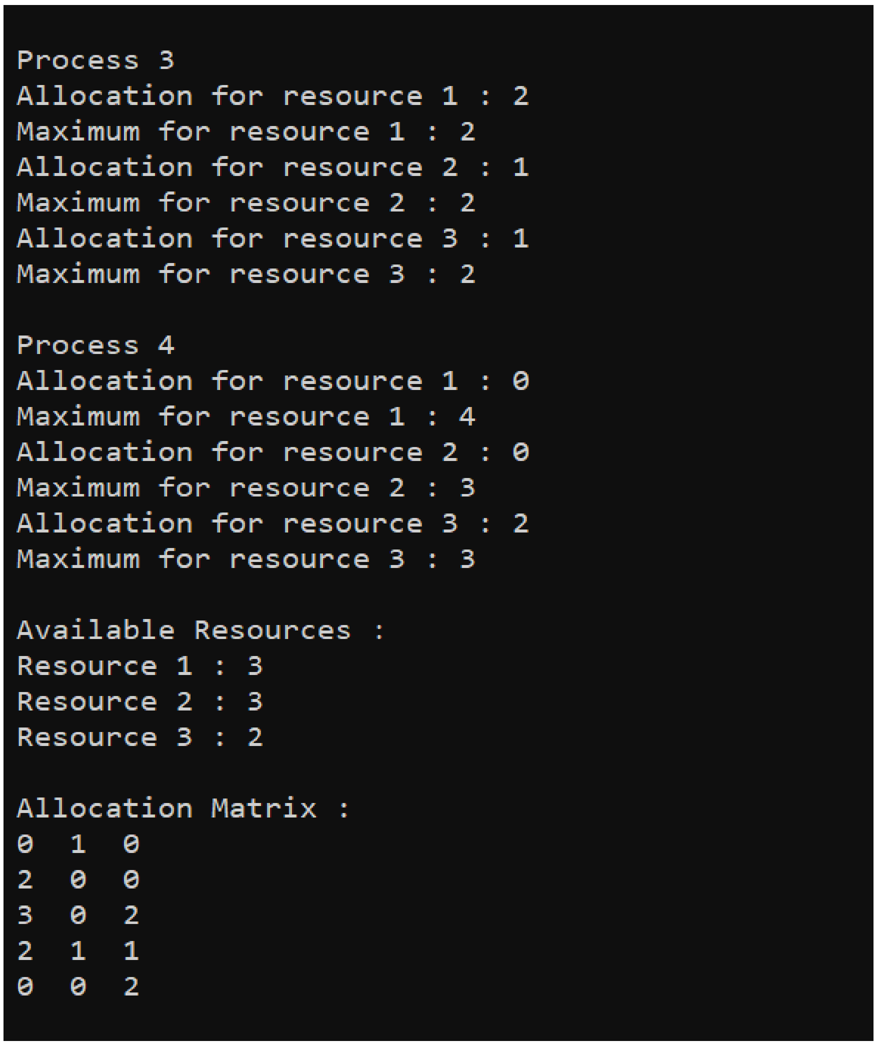
**Program:**

```
#include <stdio.h>
void main(){
    int n,m,i,j,k;
    printf("\n Enter total number of processes : ");
    scanf("%d", &n);
    printf("\n Enter total number of resources : ");
    scanf("%d", &m);
    int Alloc[n][m],Max[n][m],Need[n][m],Avail[m],Finish[n],Work[m],count=0,flag,seq[n],l=0;
    for(i = 0; i < n; i++){
        printf("\n Process %d\n", i);
        for(j = 0; j < m; j++){
            printf(" Allocation for resource %d : ", j+1 );
            scanf("%d", &Alloc[i][j]);
            printf(" Maximum for resource %d : ", j+1 );
            scanf("%d", &Max[i][j]);
        }
    }
    printf("\n Available Resources : \n");
    for (j = 0; j < m; j++){
        printf(" Resource %d : ", j+1);
        scanf("%d", &Avail[j]);
    }
    for(i = 0; i < n; i++){
        for (j = 0; j < m; j++){
            Need[i][j] = Max[i][j] - Alloc[i][j];
        }
    }
    printf("\n Allocation Matrix : \n");
    for(i = 0; i < n; i++){
        for (j = 0; j < m; j++){
            printf(" %d ",Alloc[i][j]);
        }
        printf("\n");
    }
    printf("\n Maximum Matrix : \n");
    for(i = 0; i < n; i++){
        for (j = 0; j < m; j++){
            printf(" %d ",Max[i][j]);
        }
        printf("\n");
    }
    printf("\n Available Matrix : \n");
        for (j = 0; j < m; j++){
            printf(" %d ",Avail[j]);
        }

    printf("\n\n Need Matrix : \n");
    for(i = 0; i < n; i++){
        for (j = 0; j < m; j++){
            printf(" %d ",Need[i][j]);
        }
        printf("\n");
    }
    for(i=0;i<n;i++){
        Finish[i]=0;
    }
    for(j=0;j<m;j++){
        Work[j]=Avail[j];
    }
    for (k = 0; k < n; k++){
        for (i = 0; i < n; i++){
            if (Finish[i] == 0){
                flag = 0;
                for (j = 0; j < m; j++){
                    if (Need[i][j] > Work[j]){
                        flag = 1;
                    }
                }
                if (flag== 0 && Finish[i] == 0){
                    for (j = 0; j < m; j++){
                        Work[j] += Alloc[i][j];
                    }
                    Finish[i] = 1;
                    count++;
                    seq[l]=i;
                    l++;
                }
            }
        }
    }
    if(count==n){
        printf(" Safe Sequence : ");
        for(i=0;i<l;i++){
        printf(" P%d ",seq[i]);
        }
    }else{
        printf(" Deadlock Occurs ");
    }
}
```

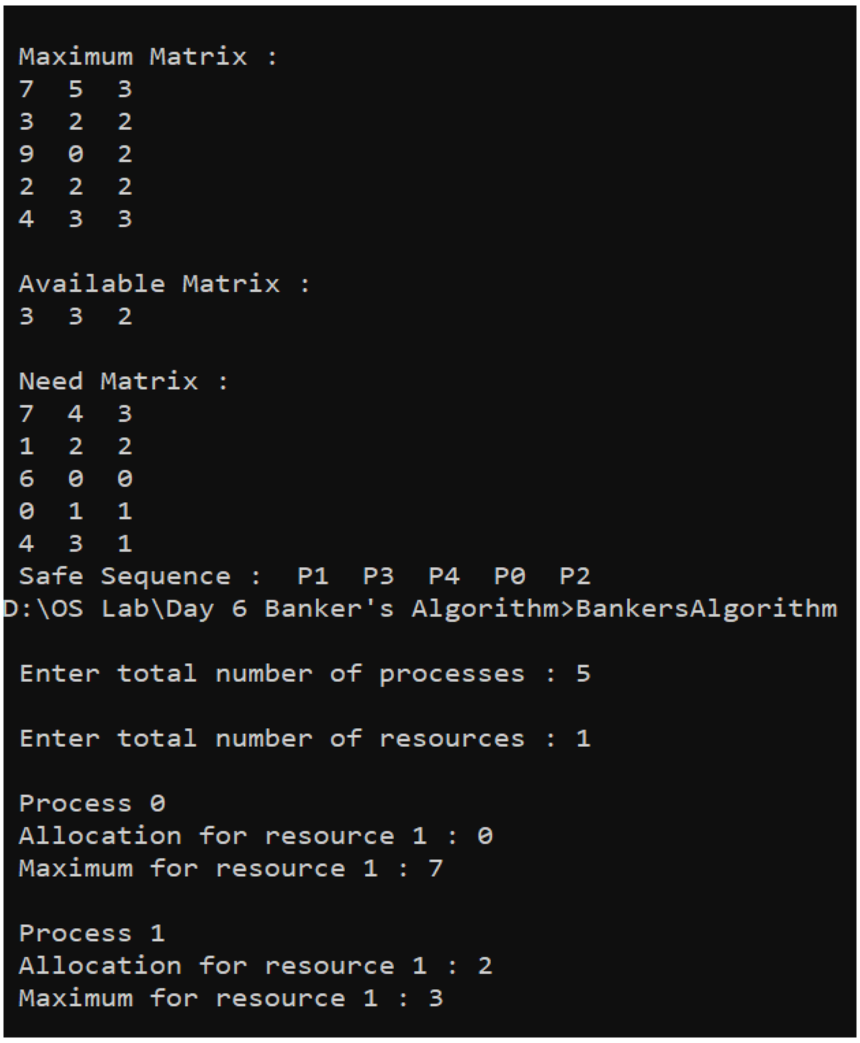
**Output:**



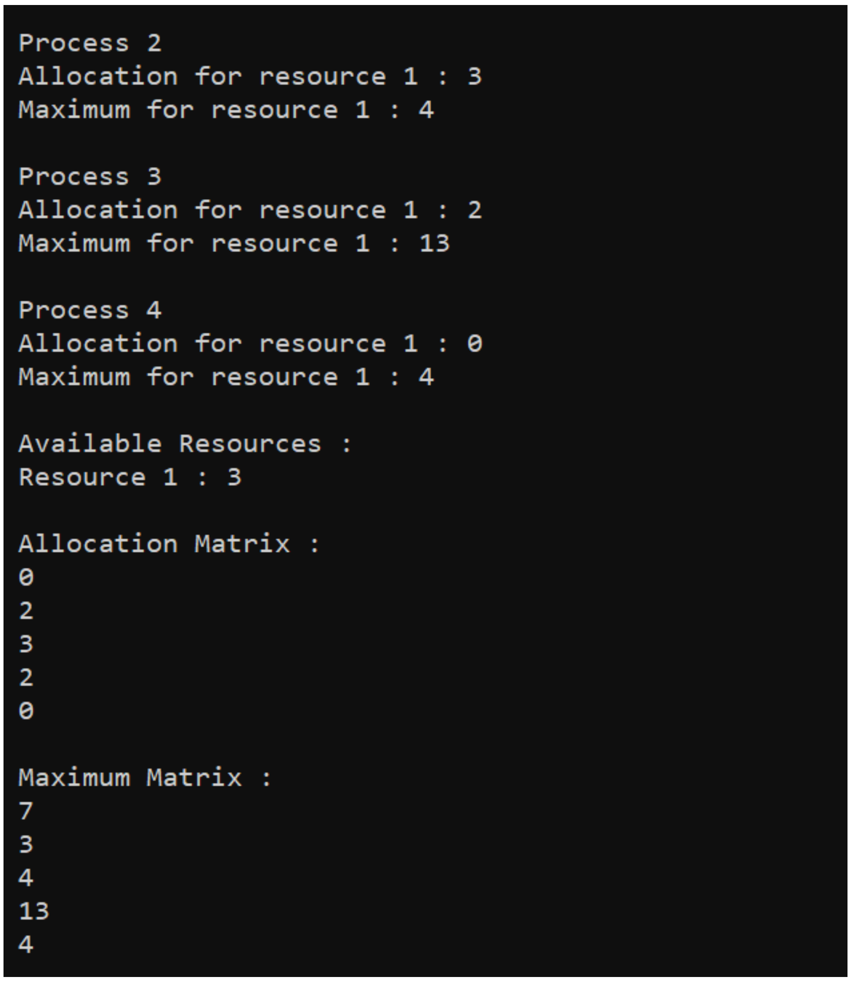
```
D:\OS Lab\Day 6 Banker's Algorithm>BankersAlgorithm

 Enter total number of processes : 5

 Enter total number of resources : 3

 Process 0
 Allocation for resource 1 : 0
 Maximum for resource 1 : 7
 Allocation for resource 2 : 1
 Maximum for resource 2 : 5
 Allocation for resource 3 : 0
 Maximum for resource 3 : 3

 Process 1
 Allocation for resource 1 : 2
 Maximum for resource 1 : 3
 Allocation for resource 2 : 0
 Maximum for resource 2 : 2
 Allocation for resource 3 : 0
 Maximum for resource 3 : 2

 Process 2
 Allocation for resource 1 : 3
 Maximum for resource 1 : 9
 Allocation for resource 2 : 0
 Maximum for resource 2 : 0
 Allocation for resource 3 : 2
 Maximum for resource 3 : 2
```



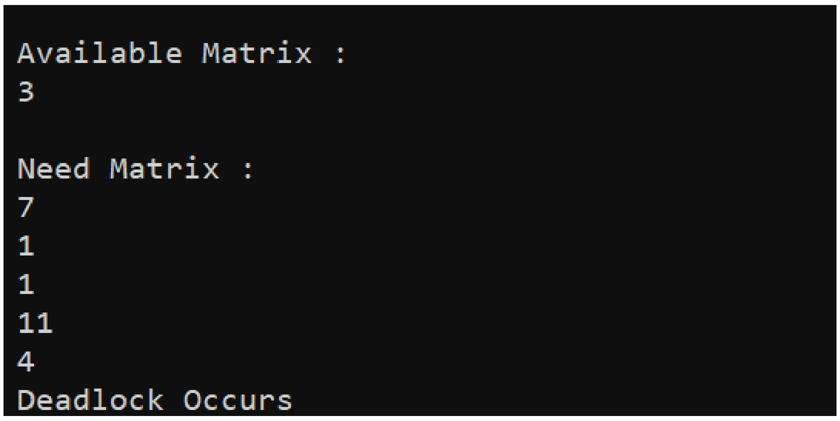
```
 Process 3
 Allocation for resource 1 : 2
 Maximum for resource 1 : 2
 Allocation for resource 2 : 1
 Maximum for resource 2 : 2
 Allocation for resource 3 : 1
 Maximum for resource 3 : 2

 Process 4
 Allocation for resource 1 : 0
 Maximum for resource 1 : 4
 Allocation for resource 2 : 0
 Maximum for resource 2 : 3
 Allocation for resource 3 : 2
 Maximum for resource 3 : 3

 Available Resources :
 Resource 1 : 3
 Resource 2 : 3
 Resource 3 : 2

 Allocation Matrix :
 0  1  0
 2  0  0
 3  0  2
 2  1  1
 0  0  2
```



```
Maximum Matrix :
7  5  3
3  2  2
9  0  2
2  2  2
4  3  3

Available Matrix :
3  3  2

Need Matrix :
7  4  3
1  2  2
6  0  0
0  1  1
4  3  1
Safe Sequence :  P1  P3  P4  P0  P2
D:\OS Lab\Day 6 Banker's Algorithm>BankersAlgorithm

Enter total number of processes : 5

Enter total number of resources : 1

Process 0
Allocation for resource 1 : 0
Maximum for resource 1 : 7

Process 1
Allocation for resource 1 : 2
Maximum for resource 1 : 3
```



```
Process 2
Allocation for resource 1 : 3
Maximum for resource 1 : 4

Process 3
Allocation for resource 1 : 2
Maximum for resource 1 : 13

Process 4
Allocation for resource 1 : 0
Maximum for resource 1 : 4

Available Resources :
Resource 1 : 3

Allocation Matrix :
0
2
3
2
0

Maximum Matrix :
7
3
4
13
4
```



```
Available Matrix :
3

Need Matrix :
7
1
1
11
4
Deadlock Occurs
```