LAB-12

1. **Banker's Algorithm.**

⇨ **Code:**

```
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
int allDone(int n,int finish[]);
int processCanTake(int n,int need[],int avail[]);
void main()
{
    int i,j,n=5,m=3,index=0;
    printf("Enter number of resources\n");
    scanf("%d",&m);
    printf("Enter number of processes\n");
    scanf("%d",&n);
    int ans[n];
    int finish[n];
    int available[m];
    int max[n][m];
    int allocated[n][m];
    int need[n][m];
    printf("Enter available vectore\n");
    for(j=0;j<m;j++)
    {
            scanf("%d",&available[j]);
    }
    printf("Enter max matrix\n");
    for(i=0;i<n;i++)
    {
            finish[i]=0;
            for(j=0;j<m;j++)
            {
                    scanf("%d",&max[i][j]);
            }
    }
```

```
    printf("Enter allocation matrix\n");
    for(i=0;i<n;i++)
    {
            for(j=0;j<m;j++)
            {
                    scanf("%d",&allocated[i][j]);
            }
    }
    for(i=0;i<m;i++)
    {
            int total=0;
            for(j=0;j<n;j++)
            {
                    total+=allocated[j][i];
            }
            available[i]-=total;
    }
    printf("Need\n");
    for(i=0;i<n;i++)
    {
            for(j=0;j<m;j++)
            {
                    need[i][j]=max[i][j]-allocated[i][j];
            }
    }
    while(!allDone(n,finish))
    {
            int k=0;
            while((finish[k]==1) || !processCanTake(m,need[k],available) &&
k<n)
                    k++;
            if(k==n)
            {
                    printf("Unsafe stat\n");
                    return;
```

```
            }
            for(i=0;i<m;i++)
            {
                available[i]+=allocated[k][i];
                need[k][i]=0;
                allocated[k][i]=0;
            }
            finish[k]=1;
            ans[index++]=k;
    }
    printf("\nSequence of processes\n");
    for(i=0;i<index;i++)
            printf("P%d ",ans[i]);
    printf("\n");
}
int allDone(int n,int finish[])
{
    int i,flag=1;
    for(i=0;i<n;i++)
    {
            if(finish[i]==0)
            {
                flag=0;
                break;
            }
    }
    return flag;
}
int processCanTake(int n,int need[],int avail[])
{
    int i=0,flag=1;
    for(i=0;i<n;i++)
    {
            if(need[i]>avail[i])
            {
```

```
                    flag=0;
                    break;
                }
            }
        return flag;
    }
```

⇨ **Output:**