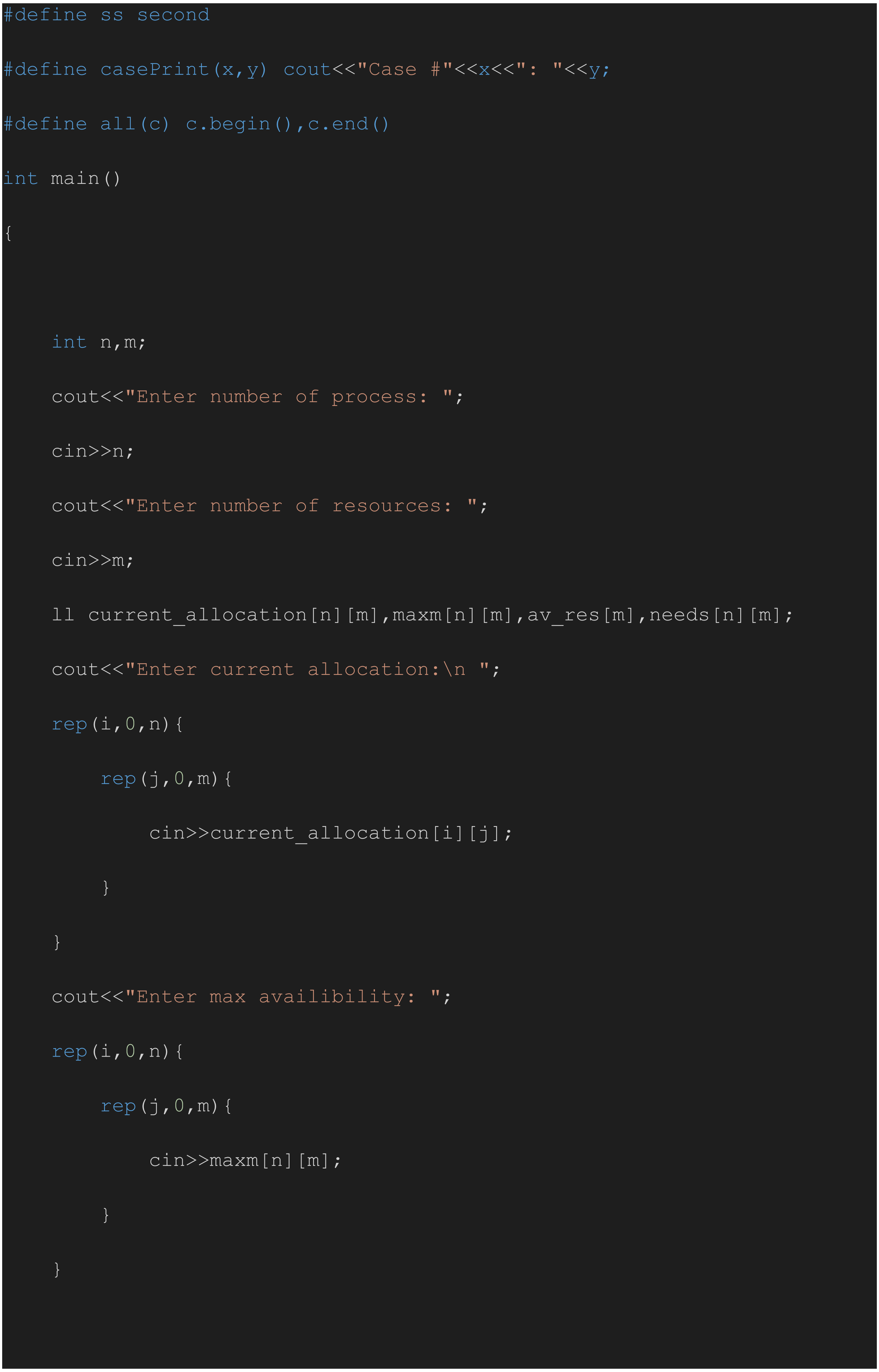
CSE2005- Operating Systems

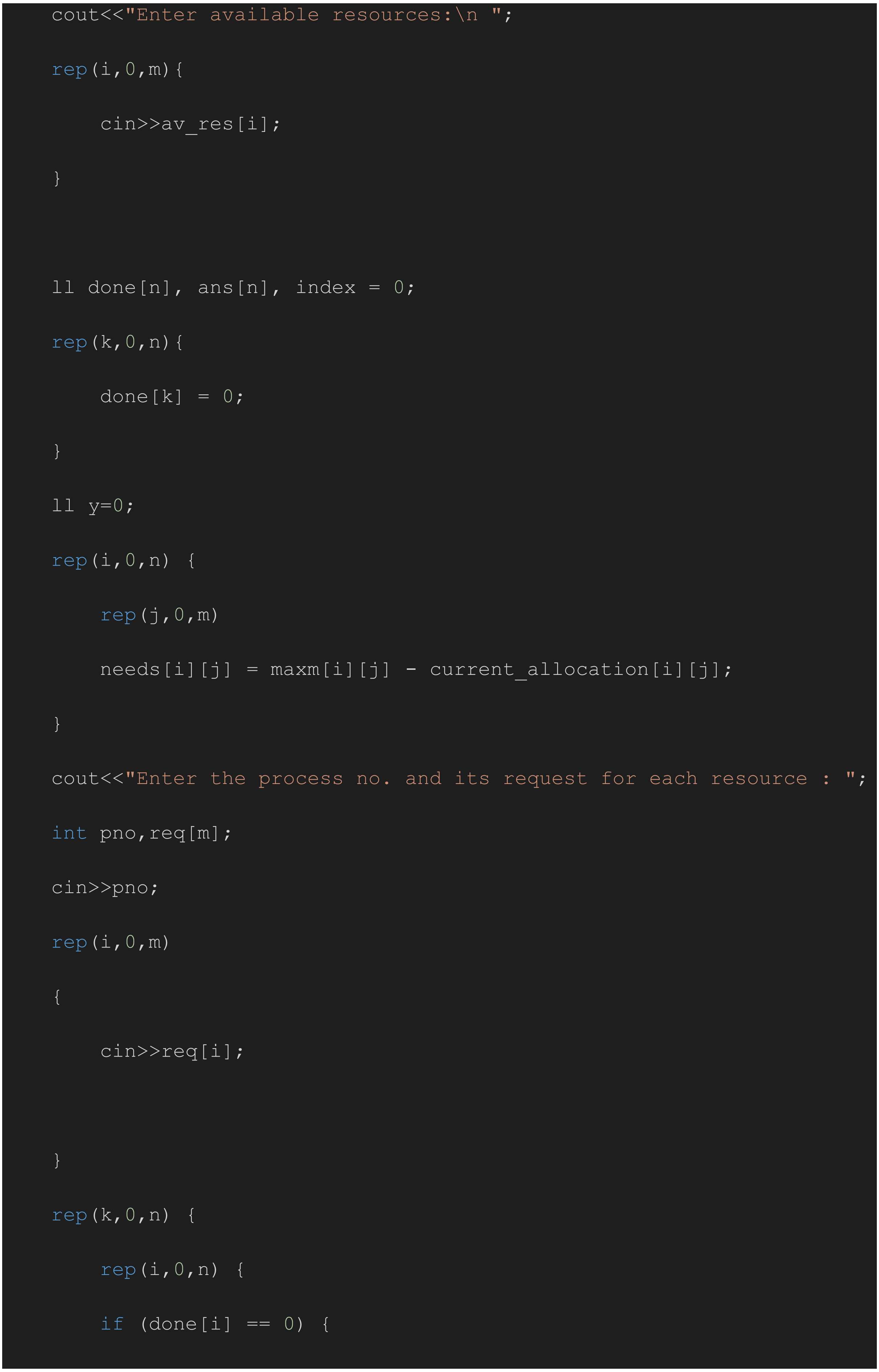
Lab Ex. 7 Banker’s algorithm

Code:

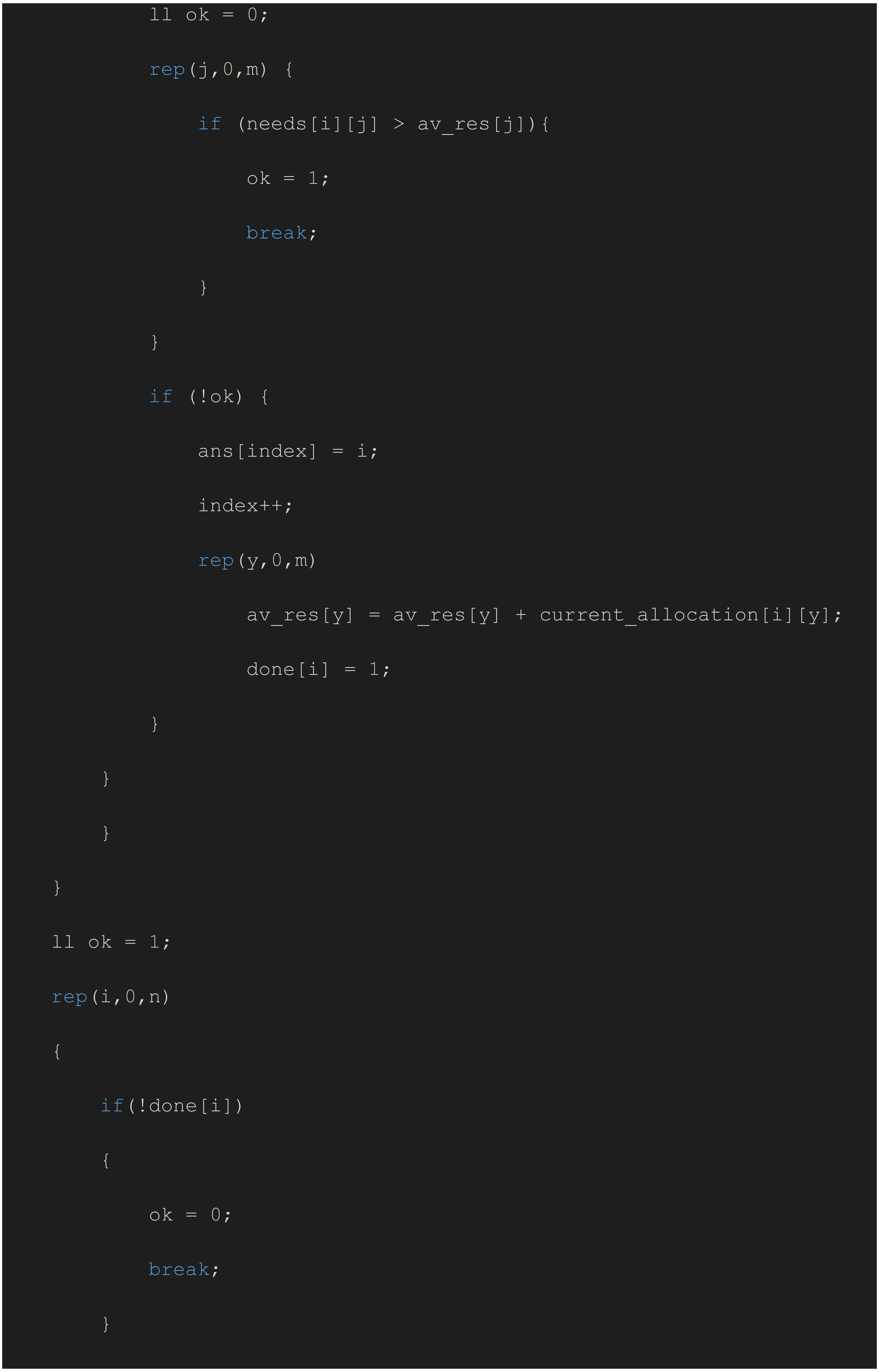
```
#include<bits/stdc++.h>
using namespace std;
using ll=long long;
void ynans(bool x){if(x) cout<<"YES";else cout<<"NO";}
#define vi vector<int>
#define rep(i,k,n) for(ll i=k;i<n;i++)
#define rof(i,k,n) for(ll i=k;i>n;i--)
#define pb(x) push_back(x)
#define sp(x,y) fixed<<setprecision(y)<<x
int sum() { return 0; }
template<typename T, typename... Args>
T sum(T a, Args... args) { return a + sum(args...); }
#define vi vector<int>
#define vc vector<char>
#define vs vector<string>
#define vll vector<ll>
#define vvi vector < vi >
#define pll pair<ll, ll>
#define ff first
```



```
#define ss second
#define casePrint(x,y) cout<<"Case #"<<x<<": "<<y;
#define all(c) c.begin(),c.end()
int main()
{

    int n,m;
    cout<<"Enter number of process: ";
    cin>>n;
    cout<<"Enter number of resources: ";
    cin>>m;
    ll current_allocation[n][m],maxm[n][m],av_res[m],needs[n][m];
    cout<<"Enter current allocation:\n ";
    rep(i,0,n){
        rep(j,0,m){
            cin>>current_allocation[i][j];
        }
    }
    cout<<"Enter max availibility: ";
    rep(i,0,n){
        rep(j,0,m){
            cin>>maxm[n][m];
        }
    }
```



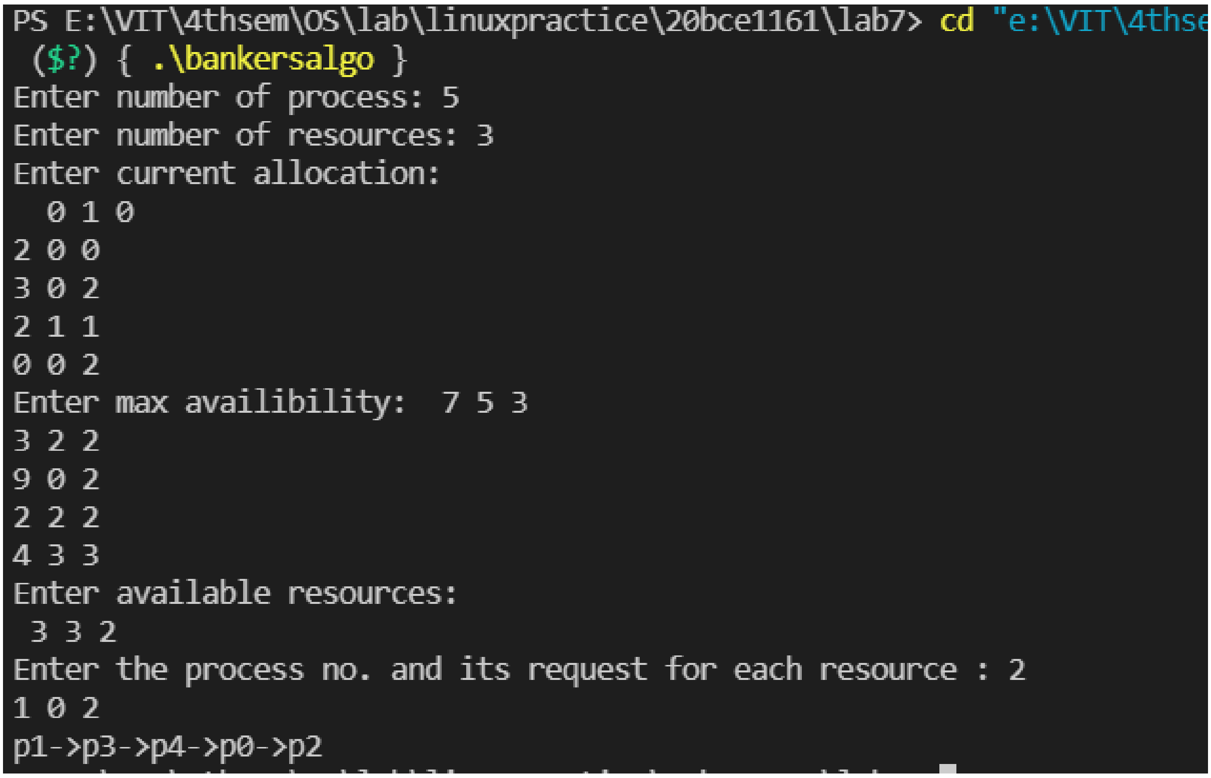
```
    cout<<"Enter available resources:\n ";
    rep(i,0,m){
        cin>>av_res[i];
    }

    ll done[n], ans[n], index = 0;
    rep(k,0,n){
        done[k] = 0;
    }
    ll y=0;
    rep(i,0,n) {
        rep(j,0,m)
        needs[i][j] = maxm[i][j] - current_allocation[i][j];
    }
    cout<<"Enter the process no. and its request for each resource : ";
    int pno,req[m];
    cin>>pno;
    rep(i,0,m)
    {
        cin>>req[i];

    }
    rep(k,0,n) {
        rep(i,0,n) {
        if (done[i] == 0) {
```



```
        ll ok = 0;

        rep(j,0,m) {

            if (needs[i][j] > av_res[j]){

                ok = 1;

                break;

            }

        }

        if (!ok) {

            ans[index] = i;

            index++;

            rep(y,0,m)

                av_res[y] = av_res[y] + current_allocation[i][y];

                done[i] = 1;

        }

    }

    }

}

ll ok = 1;

rep(i,0,n)

{

    if(!done[i])

    {

        ok = 0;

        break;

    }
    }

    if(ok)
    {
        cout << "Safe Sequence\n";
        rep(i,0,n-1){
            cout << " p" << ans[i] << " ->";
        }
        cout << " p" << ans[n - 1];
    }
    else{
         cout << "Need is greater than availability\n";

    }


    return 0;
}
```

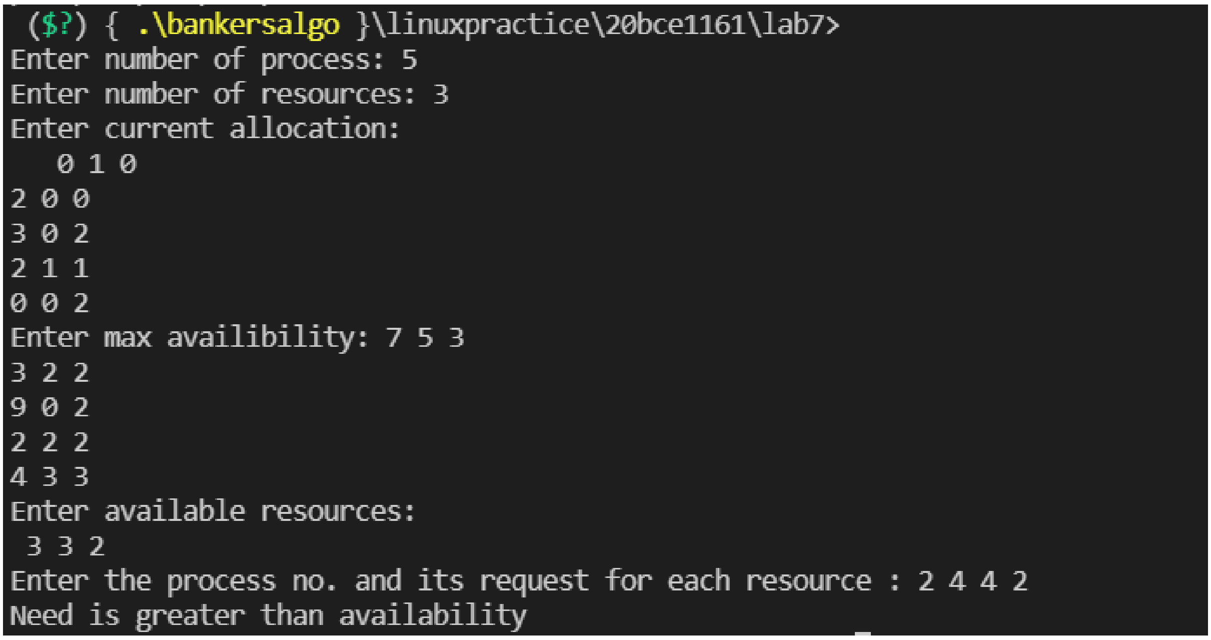
Output:

1.



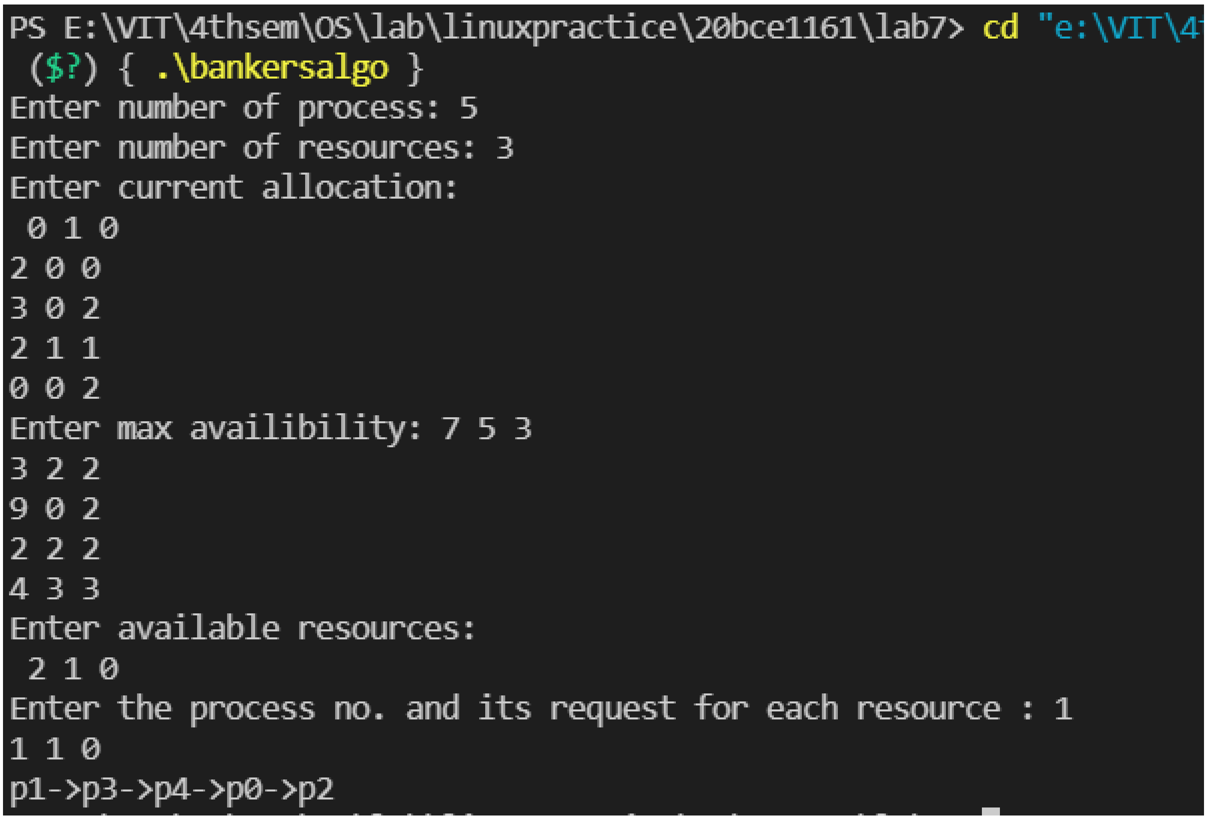
```
PS E:\VIT\4thsem\OS\lab\linuxpractice\20bce1161\lab7> cd "e:\VIT\4thse
 ($?) { .\bankersalgo }
Enter number of process: 5
Enter number of resources: 3
Enter current allocation:
  0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter max availibility:  7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter available resources:
 3 3 2
Enter the process no. and its request for each resource : 2
1 0 2
p1->p3->p4->p0->p2
```

2.



```
 ($?) { .\bankersalgo }\linuxpractice\20bce1161\lab7>
Enter number of process: 5
Enter number of resources: 3
Enter current allocation:
   0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter max availibility: 7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter available resources:
 3 3 2
Enter the process no. and its request for each resource : 2 4 4 2
Need is greater than availability
```

3.



```
PS E:\VIT\4thsem\OS\lab\linuxpractice\20bce1161\lab7> cd "e:\VIT\4
 ($?) { .\bankersalgo }
Enter number of process: 5
Enter number of resources: 3
Enter current allocation:
 0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter max availibility: 7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter available resources:
 2 1 0
Enter the process no. and its request for each resource : 1
1 1 0
p1->p3->p4->p0->p2
```