Experiment 3. Banker's Algorithm

- Purpose:

Understand the resource allocation of multiple processes executing concurrently in a multiprogramming system
Master the causes of deadlocks, the necessary conditions for deadlocks and the basic methods of dealing with deadlocks
Master the method of deadlock prevention and the basic concept of system security state.
Master the banker's algorithm and understand the resource allocation strategy of resources in the concurrent execution of processes

- Experiment content:

Design a program in which n concurrent processes share m system resources to implement the banker's algorithm. Requirements include:
Simple selection interface.
It can display the occupancy and remaining conditions of the current system resources.
The process makes a resource request, if it can be satisfied, it will be allocated, if it cannot be satisfied, it will be blocked.

- hint:

Each process must specify in advance the maximum amount of resources it requires
Each process makes a partial resource request and gets allocated or blocked each time;
If the maximum amount of resources of the process is satisfied, all should be returned to the system within a limited time

Code:

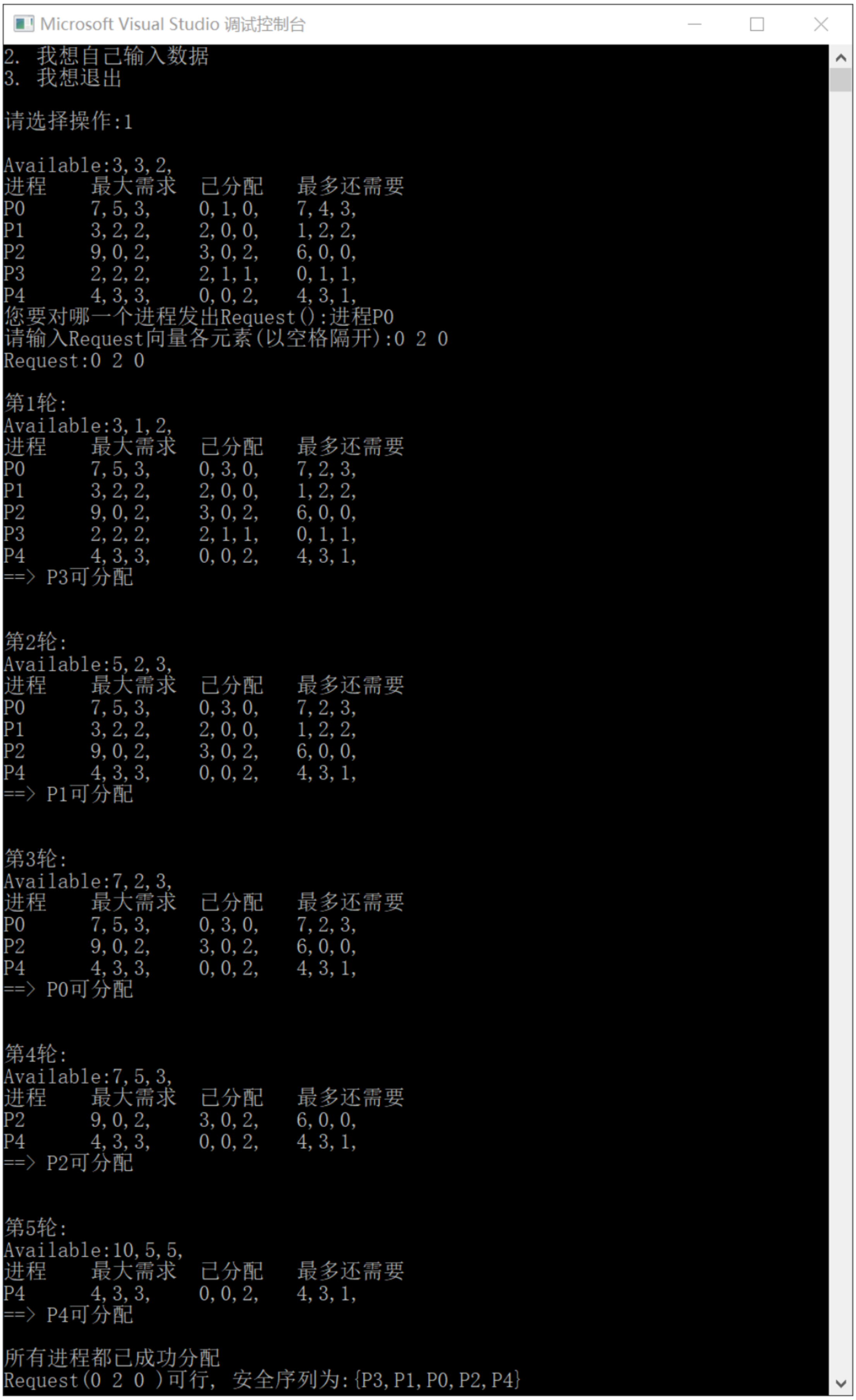
```
#include <iostream>
using namespace std;

#define m 3
#define n 5

int Available[ m ]
= { 3,3,2 }; //Available resource vector
int Max[ n ][ m ]
= { {7,5,3}, {3,2,2}, {9,0,2} ,{2,2,2}, {4,3,3} }; //maximum demand matrix
int Allocation[ n ][ m ]
= { {0,1,0},{2,0,0},{3,0,2},{2,1,1},{0,0,2}}; //allocation matrix
int Need[ n ][ m ]; //need matrix
int Request[ m ]
= { 0,0,0 };
int drop[ n ] = { 0,0,0,0,0 };
int Safe[n] = { 0,0,0,0,0 };

int Init(int flag) {
    if (flag == 1) {  //use default data
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                Need[i][j] = Max[i][j] - Allocation[i][j];
    }
    else if (flag == 0) {
        for (int i = 0; i < m; i++) {
            Available[i] = 0;
        }
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                Max[i][j] = 0;
                Need[i][j] = 0;
            }
        }
        cout << "Available:";
        for (int i = 0; i < m; i++) {
            cin >> Available[i];
        }

        for (int i = 0; i < n; i++) {
            cout << "P" << i << ":" << endl;
            cout << "Max:";
            for (int j = 0; j < m; j++) {
                cin >> Max[i][j];
            }
            cout << "Allocation:";
            for (int j = 0; j < m; j++) {
                cin >> Allocation[i][j];
            }
        }

        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                Need[i][j] = Max[i][j] - Allocation[i][j];
    }
    return 0;
}
```

```
void showSituation() {
    cout << "Available:" ;
    for ( int i = 0; i < m ; i++) {
        cout << Available[i] << "," ;
    }
    cout << endl << "Process\tMaximum Needs Allocated At Most Remaining Needs" << endl;
    for (int i = 0; i < n; i++) {
        if (drop[i] == 1)
            continue;
        cout << "P" << i << '\t';
        for (int j = 0; j < m; j++) {
            cout << Max[i][j] << ',';
        }
        cout << '\t' << "  ";
        for (int j = 0; j < m; j++) {
            cout << Allocation[i][j] << ',';
        }
        cout << "   ";
        for (int j = 0; j < m; j++) {
            cout << Need[i][j] << ',';

        }
        cout << endl;
    }
}
int show(int k, int ii) {
    cout << "The" << k << "round:" << endl;

    showSituation();

    cout << "==> P" << ii << "assignable" << endl;
    return 0;
}
int Bank() {
    int flag = 0;
    int i = 0, j = 0 ,time=0;
    for (int k = 0; k < n; k++) {
        for (i = 0; i < n; i++) {
            if (drop[i] == 1)
                continue;
            else {
                for (j = 0; j < m; j++) {
                    if (Need[i][j] > Available[j]) {
                        flag = 1;
                        break;
                    }
                }
                if (flag == 0) {
                    flag = 2;
                    break;
                }
                else if (flag == 1)
                    flag = 0;
            }
        }
        if (flag == 0 || i == 5) {
            cout << "At this time, do not continue!!!" << endl << "To sum up, when
Request=<" ;
            for ( int ii = 0; ii < m ; ii++) {
                cout << Request[ii] << ' ' ;
            }
            cout << ">The system will be in an unsafe state!!!" << endl << "Request is
unsatisfiable and will block!!!" << endl << endl;
            return -1;
        }
        else if (flag == 2) {
            flag = 0;
            show(k + 1, i);
            for ( int p = 0; p < m ; p++) {
                Available[p] = Allocation[i][p] + Available[p];
                //cout << Available[p] << " ";
            }
            cout << endl;

            Safe[time] = i;
            time++;

            drop[i] = 1;
            if (k == 4) {
                cout << "All processes were allocated successfully" << endl <<
"Request(" ;
                for ( int r = 0; r < 3; r++)
                    cout << Request[r] << ' ' ;
                cout << ") feasible, " ;
                cout << "The safe sequence is: {P" << Safe[0] << ",P" << Safe[1] << ",P"
<< Safe[2] << ",P" << Safe[3] << ",P" << Safe[4] << "}" << endl;
            }
        }
        cout << endl;
    }
    return 0;
}
int Check(int ff) {
    for (int i = 0; i < m; i++) {
        Available[i] = Available[i] - Request[i];
        Allocation[ff][i] = Allocation[ff][i] + Request[i];
        Need[ff][i] = Need[ff][i] - Request[i];
    }
    //if (Bank() == -1)
    //  cout << "ERROR !!!";
    return 0;
}

int Menu() {
    cout << "-----------------------Banker Algorithm Simulation System
-------------------- --------" << endl;
    cout << "1. I want to use the default data and automatically load the default test data"
<< endl << "2. I want to enter the data myself" << endl << "3. I want to quit" << endl <<
endl << "Please select an action:" ;
    int f = 0, ff = 0;
    cin >> f;
    if (f == 1 || f == 2) {
        if (f == 1) {
            Init(1);
            cout << endl;
            showSituation();
        }
        if (f == 2) {
            Init(0);
            cout << endl;
            showSituation();
        }
        cout << "Which process do you want to issue Request(): process P" ;
        cin >> ff;
        cout << "Please enter the elements of the Request vector (separated by spaces):" ;
        for (int i = 0; i < m; i++)
            cin >> Request[i];
        cout << "Request:";
        for (int i = 0; i < m; i++)
            cout << Request[i] << ' ';
        cout << endl << endl;
        Check(ff);
        Bank();
        return 1;
    }
    else
        return 0;
}

int main() {
    Menu();
}
```

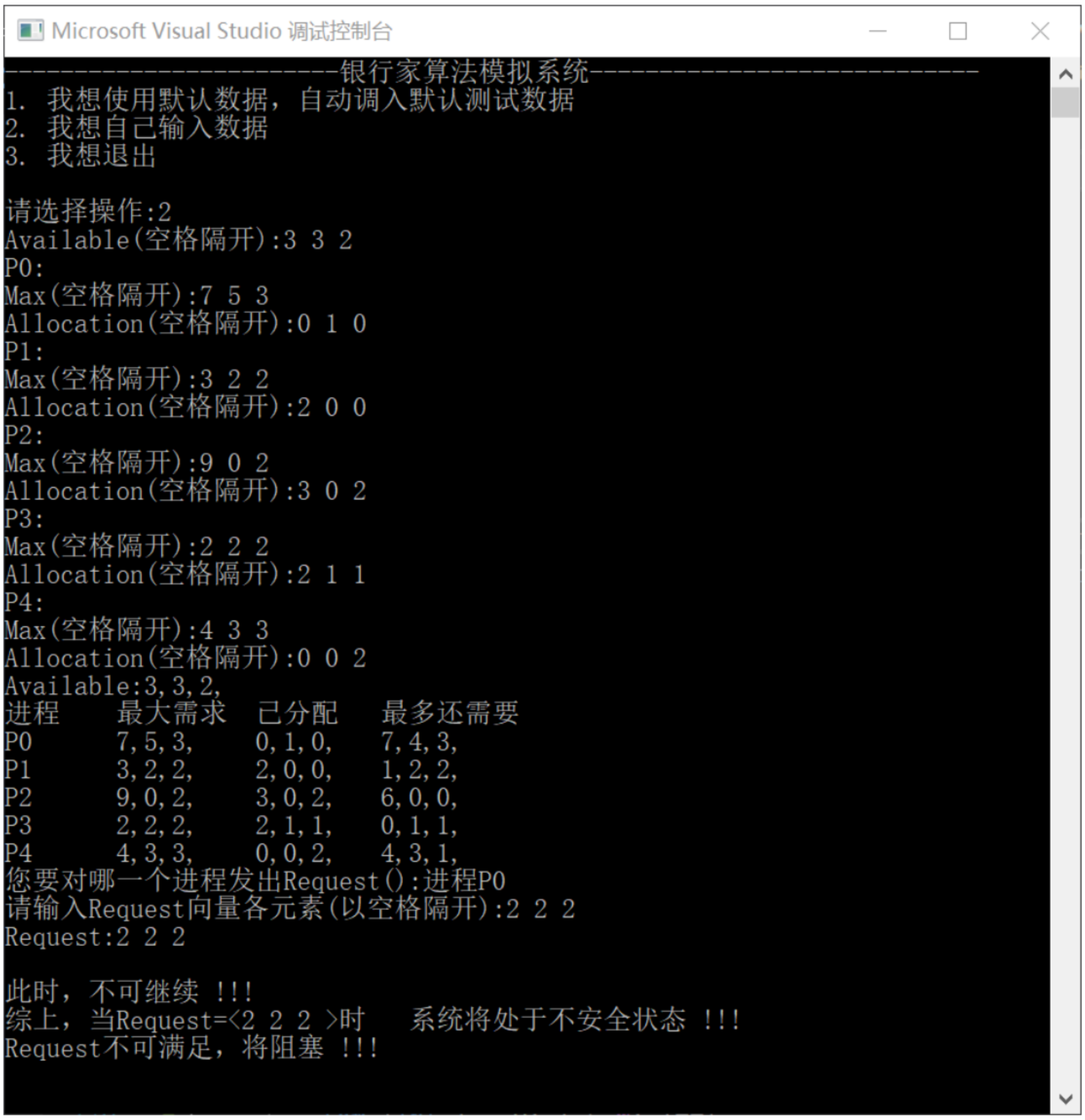
Screenshot of the experiment result:

Screenshot 1: Use the default data, Request0(0,2,0), the allocation is successful

Screenshot 2: Input data by yourself, Request0(2,2,2), the system will be in an unsafe state


Microsoft Visual Studio 调试控制台

```
2. 我想自己输入数据
3. 我想退出

请选择操作:1

Available:3,3,2,
进程    最大需求  已分配   最多还需要
P0      7,5,3,    0,1,0,   7,4,3,
P1      3,2,2,    2,0,0,   1,2,2,
P2      9,0,2,    3,0,2,   6,0,0,
P3      2,2,2,    2,1,1,   0,1,1,
P4      4,3,3,    0,0,2,   4,3,1,
您要对哪一个进程发出Request():进程P0
请输入Request向量各元素(以空格隔开):0 2 0
Request:0 2 0

第1轮:
Available:3,1,2,
进程    最大需求  已分配   最多还需要
P0      7,5,3,    0,3,0,   7,2,3,
P1      3,2,2,    2,0,0,   1,2,2,
P2      9,0,2,    3,0,2,   6,0,0,
P3      2,2,2,    2,1,1,   0,1,1,
P4      4,3,3,    0,0,2,   4,3,1,
==> P3可分配


第2轮:
Available:5,2,3,
进程    最大需求  已分配   最多还需要
P0      7,5,3,    0,3,0,   7,2,3,
P1      3,2,2,    2,0,0,   1,2,2,
P2      9,0,2,    3,0,2,   6,0,0,
P4      4,3,3,    0,0,2,   4,3,1,
==> P1可分配


第3轮:
Available:7,2,3,
进程    最大需求  已分配   最多还需要
P0      7,5,3,    0,3,0,   7,2,3,
P2      9,0,2,    3,0,2,   6,0,0,
P4      4,3,3,    0,0,2,   4,3,1,
==> P0可分配


第4轮:
Available:7,5,3,
进程    最大需求  已分配   最多还需要
P2      9,0,2,    3,0,2,   6,0,0,
P4      4,3,3,    0,0,2,   4,3,1,
==> P2可分配


第5轮:
Available:10,5,5,
进程    最大需求  已分配   最多还需要
P4      4,3,3,    0,0,2,   4,3,1,
==> P4可分配

所有进程都已成功分配
Request(0 2 0 )可行, 安全序列为:{P3,P1,P0,P2,P4}
```



```
------------------------银行家算法模拟系统----------------------------
1. 我想使用默认数据, 自动调入默认测试数据
2. 我想自己输入数据
3. 我想退出

请选择操作:2
Available(空格隔开):3 3 2
P0:
Max(空格隔开):7 5 3
Allocation(空格隔开):0 1 0
P1:
Max(空格隔开):3 2 2
Allocation(空格隔开):2 0 0
P2:
Max(空格隔开):9 0 2
Allocation(空格隔开):3 0 2
P3:
Max(空格隔开):2 2 2
Allocation(空格隔开):2 1 1
P4:
Max(空格隔开):4 3 3
Allocation(空格隔开):0 0 2
Available:3,3,2,
进程    最大需求  已分配   最多还需要
P0      7,5,3,    0,1,0,   7,4,3,
P1      3,2,2,    2,0,0,   1,2,2,
P2      9,0,2,    3,0,2,   6,0,0,
P3      2,2,2,    2,1,1,   0,1,1,
P4      4,3,3,    0,0,2,   4,3,1,
您要对哪一个进程发出Request():进程P0
请输入Request向量各元素(以空格隔开):2 2 2
Request:2 2 2

此时, 不可继续 !!!
综上, 当Request=<2 2 2 >时   系统将处于不安全状态 !!!
Request不可满足, 将阻塞 !!!
```