实验代码：

```
#include<iostream>
using namespace std;
//定义银行家算法的数据结构
int Available[3] = { 3,3,2 };//系统可用资源
int Max[5][3] = { {7,5,3},{3,2,2},{9,0,2},{2,2,2},{4,3,3} };//进程最大资源需求量
int Allocation[5][3] = { {0,1,0},{2,0,0},{3,0,2},{2,1,1},{0,0,2} };//系统已经给进程分配的资源
int Need[5][3] = { {7,4,3},{1,2,2},{6,0,0},{0,1,1},{4,3,1} };//进程的资源最大需求量
int Work[3];//安全性检查算法中的工作向量
int Finish[5] = { 0,0,0,0,0 };//进程执行完成的标志
int SafeArray[5];//安全序列
int Request[3];//请求资源向量

void ShowSafe(int i);
//打印当前系统资源的分配情况
void Show() {
	cout << "T0 时刻的系统资源分配情况如下：" << endl;
	cout << "进程名\tMax\t\tAllocation\tNeed\t\tAvailable" << endl;
	for (int i = 0; i < 5; i++) {
		cout << "P" << i << "\t";
		for (int j = 0; j < 3; j++) {
			cout << Max[i][j] << " ";
		}
		cout << "\t\t";
		for (int j = 0; j < 3; j++) {
			cout << Allocation[i][j] << " ";
		}
		cout << "\t\t";
		for (int j = 0; j < 3; j++) {
			cout << Need[i][j] << " ";
		}
		cout << "\t\t";
		if (i == 0) {
			for (int j = 0; j < 3; j++) {
				cout << Available[j] << " ";
			}
		}
		cout << endl;
	}
}
//安全性检查中判断需求矩阵与工作向量关系
int NeedLessWork(int i) {
	for (int j = 0; j < 3; j++) {
		if (Need[i][j] > Work[j]) {
				return 0;
			}
		}
		return 1;
}
//打印安全序列
void SafeLine() {
		cout << "当前系统处于安全状态..." << endl;
		cout << "其中一个安全序列为：";
		for (int i = 0; i < 5; i++) {
			if (i == 4)cout << "P" << SafeArray[i];
			else cout << "P" << SafeArray[i] << "-->";
		}
		cout << endl;
		for (int i = 0; i < 5; i++) {
			Finish[i] = 0;
		}
}

//安全性检查算法
void IsSafe(int index) {
		for (int i = 0; i < 5; i++) {
			int temp = NeedLessWork(i);
			if (Finish[i] == 0 && temp) {
				SafeArray[index] = i;
				index++;
				Finish[i] = 1;
				ShowSafe(i);
				for (int j = 0; j < 3; j++) {
					Work[j] = Work[j] + Allocation[i][j];
				}
				break;
			}
		}
		int mult = 1;
		//如果五个标志都为 1 即都已经完成，则打印安全序列，否则继续执行安全性检查算法
		for (int k = 0; k < 5; k++) {
			mult *= Finish[k];
		}
		if (mult == 0) {
			IsSafe(index);
		}
		else {
			SafeLine();
```

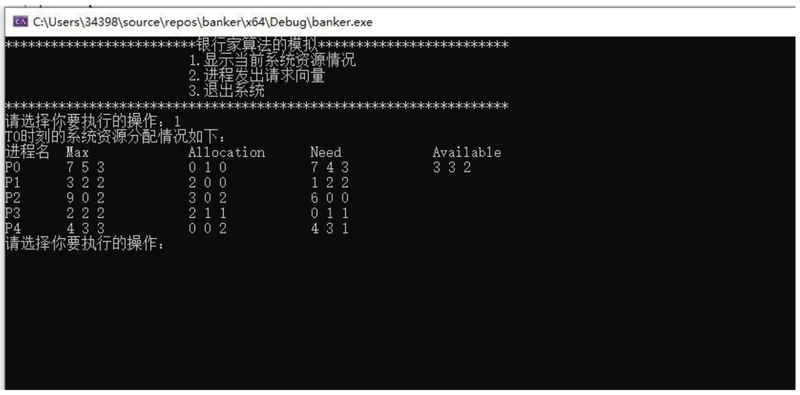
```
        }
}
void ShowSafe(int i) {
        cout << "P" << i << "\t";
        for (int j = 0; j < 3; j++) {
                cout << Work[j] << " ";
        }
        cout << "\t\t";
        for (int j = 0; j < 3; j++) {
                cout << Need[i][j] << " ";
        }
        cout << "\t\t";
        for (int j = 0; j < 3; j++) {
                cout << Allocation[i][j] << " ";
        }
        cout << "\t\t";
        for (int j = 0; j < 3; j++) {
                cout << Work[j] + Allocation[i][j] << " ";
        }
        cout << "\t\t";
        cout << Finish[i];
        cout << endl;

}
void SafeCheck() {
        cout << "试探着将资源分配给它后，系统安全情况分析如下：" << endl;
        cout << "进程\tWork\t\tNeed\t\tAllocation\tWork+Allocation\tFinish" << endl;
        for (int i = 0; i < 3; i++) {
                Work[i] = Available[i];
        }
        IsSafe(0);
}

int RequestLessNeed(int i) {
        for (int j = 0; j < 3; j++) {
                if (Request[j] > Need[i][j]) {
                        return 0;
                }
        }
        return 1;
}
int RequestLessAvailable(int i) {
        for (int j = 0; j < 3; j++) {
                if (Request[j] > Available[j]) {
```

```
                return 0;
            }
        }
        return 1;
}

//处理进程发出的资源请求
void RequestResourse() {
        cout << "请输入发出资源请求的进程：";
        int n;
        cin >> n;
        cout << "请依次输入所请求的资源数量：";
        for (int i = 0; i < 3; i++) {
                cin >> Request[i];
        }
        if (RequestLessNeed(n)) {
                if (RequestLessAvailable(n)) {
                        for (int j = 0; j < 3; j++) {
                                Available[j] = Available[j] - Request[j];
                                Allocation[n][j] = Allocation[n][j] + Request[j];
                                Need[n][j] = Need[n][j] - Request[j];
                        }
                        SafeCheck();//试探着将资源分配给请求进程，并进行安全性检查
                }
                else {
                        cout << "P" << n << "请求的资源向量已超过系统可用资源向量，请求失败！让
其继续等待..." << endl;
                        cout << "-----------------------------------------------------------------" << endl;
                        return;
                }
        }
        else {
                cout << "P" << n << "请求的资源向量已超过其最大需求向量，请求失败！让其继续
等待..." << endl;
                cout << "-----------------------------------------------------------------" << endl;
                return;
        }
}


int main() {
        cout << "************************银行家算法的模拟************************"
<< endl;
        cout << "        \t\t\t1.显示当前系统资源情况" << endl;
```

```
	cout << "        \t\t\t2.进程发出请求向量" << endl;
	cout << "        \t\t\t3.退出系统" << endl;
	cout << "*************************************************************************" 
<< endl;
	while (true) {
		int choose;
		cout << "请选择你要执行的操作：";
		cin >> choose;
		switch (choose) {
		case 1:
			Show();//展示当前系统资源分配情况
			break;
		case 2:
			RequestResourse();//处理进程发出的资源请求
			break;
		case 3:
			cout << "已退出系统！" << endl;
			return 0;
		default:
			cout << "输入错误，请重新输入！" << endl;
			continue;
		}
	}
}
```

实验结果：

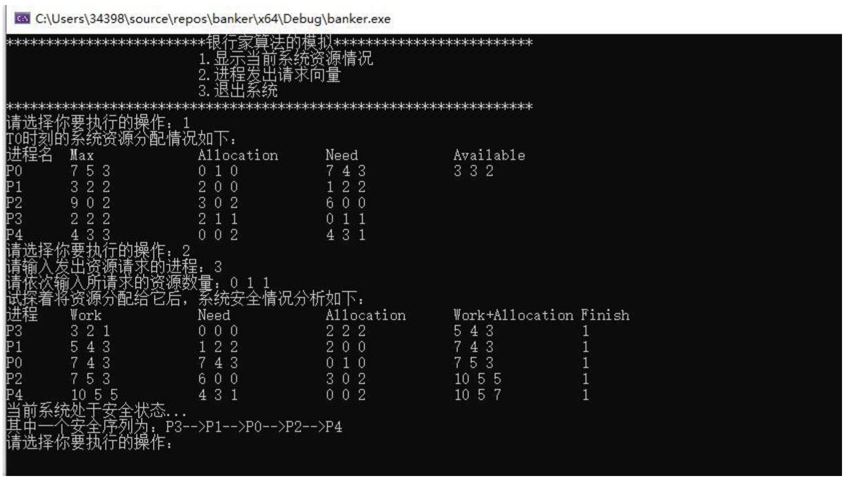
（1）当前分配：



C:\Users\34398\source\repos\banker\x64\Debug\banker.exe

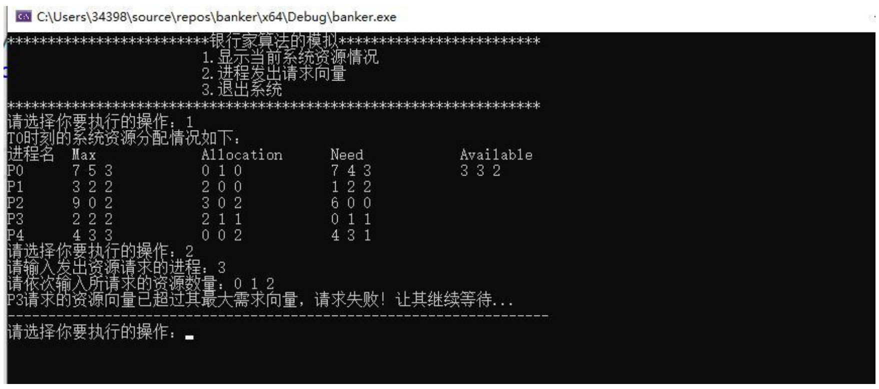
```
************************银行家算法的模拟************************
                        1.显示当前系统资源情况
                        2.进程发出请求向量
                        3.退出系统
****************************************************************
请选择你要执行的操作：1
T0时刻的系统资源分配情况如下：
进程名  Max             Allocation      Need            Available
P0      7 5 3           0 1 0           7 4 3           3 3 2
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
请选择你要执行的操作：
```

（2）安全序列



```
C:\Users\34398\source\repos\banker\x64\Debug\banker.exe
*************************银行家算法的模拟*************************
                        1.显示当前系统资源情况
                        2.进程发出请求向量
                        3.退出系统
******************************************************************
请选择你要执行的操作：1
T0时刻的系统资源分配情况如下：
进程名  Max             Allocation      Need            Available
P0      7 5 3           0 1 0           7 4 3           3 3 2
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
请选择你要执行的操作：2
请输入发出资源请求的进程：3
请依次输入所请求的资源数量：0 1 1
试探着将资源分配给它后，系统安全情况分析如下：
进程    Work            Need            Allocation      Work+Allocation Finish
P3      3 2 1           0 0 0           2 2 2           5 4 3           1
P1      5 4 3           1 2 2           2 0 0           7 4 3           1
P0      7 4 3           7 4 3           0 1 0           7 5 3           1
P2      7 5 3           6 0 0           3 0 2           10 5 5          1
P4      10 5 5          4 3 1           0 0 2           10 5 7          1
当前系统处于安全状态...
其中一个安全序列为：P3-->P1-->P0-->P2-->P4
请选择你要执行的操作：
```

（3）不安全序列



```
C:\Users\34398\source\repos\banker\x64\Debug\banker.exe
*************************银行家算法的模拟*************************
                        1.显示当前系统资源情况
                        2.进程发出请求向量
                        3.退出系统
******************************************************************
请选择你要执行的操作：1
T0时刻的系统资源分配情况如下：
进程名  Max             Allocation      Need            Available
P0      7 5 3           0 1 0           7 4 3           3 3 2
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
请选择你要执行的操作：2
请输入发出资源请求的进程：3
请依次输入所请求的资源数量：0 1 2
P3请求的资源向量已超过其最大需求向量，请求失败！让其继续等待...
-------------------------------------------------------------------
请选择你要执行的操作：
```