区块链实验五实验报告

2011428王天行，2012679王娇妹

#### 一、实验内容

在本次作业中，你将使用Solidity和web3.js在以太坊（Ethereum）上实现一个复杂的去中心化应用程序(DApp)。你需要编写一个智能合约和访问它的用户客户端，学习DApp的“全栈”开发。

#### 二、代码

##### 智能合约

// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

contract BlockchainSplitwise {

    struct MapIndex{

        uint32 amount;

        bool exists;

    }

    struct Person {

        address addr;

        mapping(address => MapIndex) owes;

        mapping(address => uint256) time;

        address[] creditors;

        uint32 count\_creditors;

    }

    struct MapIndexUsers {

        Person person;

        bool exists;

    }

    mapping(address => MapIndexUsers) private users;

    mapping(address => uint256) private last\_active;

    address[] private all\_users;

    uint32 private count\_users;

    function add\_IOU(address creditor, uint32 amount) public {

        uint32 owe = users[msg.sender].person.owes[creditor].amount + amount;

        uint32 debt = users[creditor].person.owes[msg.sender].amount;

        last\_active[msg.sender] = block.timestamp;

        if(users[msg.sender].exists == false){

            all\_users.push(msg.sender);

            users[msg.sender].exists = true;

            count\_users += 1;

        }

        Person storage \_sender = users[msg.sender].person;

        Person storage \_creditor = users[creditor].person;

        if (\_sender.owes[creditor].exists == false){

            \_sender.creditors.push(creditor);

            \_sender.count\_creditors += uint32(1);

        }

        if (\_creditor.owes[msg.sender].exists == false){

            \_creditor.creditors.push(msg.sender);

            \_creditor.count\_creditors += uint32(1);

        }

        if (owe > debt){

            \_sender.addr = msg.sender;

            \_sender.owes[creditor] = MapIndex({amount : owe - debt, exists : true});

            \_sender.time[creditor] = block.timestamp;

            \_creditor.addr = creditor;

            \_creditor.owes[msg.sender] = MapIndex({amount : uint32(0), exists : true});

            \_creditor.time[msg.sender] = block.timestamp;

        }else{

            \_sender.addr = msg.sender;

            \_sender.owes[creditor] = MapIndex({amount : uint32(0), exists : true});

            \_sender.time[creditor] = block.timestamp;

            \_creditor.addr = creditor;

            \_creditor.owes[msg.sender] = MapIndex({amount : debt - owe, exists : true});

            \_creditor.time[msg.sender] = block.timestamp;

        }

    }

    function get\_last\_active(address a) public view returns (uint256){

        return last\_active[a];

    }

    function lookup(address debtor, address creditor) public view returns (uint32){

        return users[debtor].person.owes[creditor].amount;

    }

    function getNeighbor(address debtor, uint32 index) public view returns (address){

        return users[debtor].person.creditors[index];

    }

    function getCountNeighbors(address debtor) public view returns (uint32){

        return users[debtor].person.count\_creditors;

    }

    function getUser(uint32 index) public view returns (address){

        return all\_users[index];

    }

    function getCountUser() public view returns (uint32){

        return count\_users;

    }

}

##### 客户端

abi和contractAddress

var abi =[

    {

        "inputs": [

            {

                "internalType": "address",

                "name": "creditor",

                "type": "address"

            },

            {

                "internalType": "uint32",

                "name": "amount",

                "type": "uint32"

            }

        ],

        "name": "add\_IOU",

        "outputs": [],

        "stateMutability": "nonpayable",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "address",

                "name": "a",

                "type": "address"

            }

        ],

        "name": "get\_last\_active",

        "outputs": [

            {

                "internalType": "uint256",

                "name": "",

                "type": "uint256"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "address",

                "name": "debtor",

                "type": "address"

            }

        ],

        "name": "getCountNeighbors",

        "outputs": [

            {

                "internalType": "uint32",

                "name": "",

                "type": "uint32"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [],

        "name": "getCountUser",

        "outputs": [

            {

                "internalType": "uint32",

                "name": "",

                "type": "uint32"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "address",

                "name": "debtor",

                "type": "address"

            },

            {

                "internalType": "uint32",

                "name": "index",

                "type": "uint32"

            }

        ],

        "name": "getNeighbor",

        "outputs": [

            {

                "internalType": "address",

                "name": "",

                "type": "address"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "uint32",

                "name": "index",

                "type": "uint32"

            }

        ],

        "name": "getUser",

        "outputs": [

            {

                "internalType": "address",

                "name": "",

                "type": "address"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    },

    {

        "inputs": [

            {

                "internalType": "address",

                "name": "debtor",

                "type": "address"

            },

            {

                "internalType": "address",

                "name": "creditor",

                "type": "address"

            }

        ],

        "name": "lookup",

        "outputs": [

            {

                "internalType": "uint32",

                "name": "",

                "type": "uint32"

            }

        ],

        "stateMutability": "view",

        "type": "function"

    }

];

var contractAddress = '0x4bFbf2462a8C28f133ca426521dd3d7098694701'

Todo函数实现

function **getUsers**() {

    var all\_users = [];

    var count\_users = BlockchainSplitwise.getCountUser.**call**();

    while(count\_users > 0){

     all\_users.**push**(BlockchainSplitwise.getUser.**call**(count\_users - 1));

        count\_users -= 1;

    }

    return all\_users;

}

*// TODO: Get the total amount owed by the user specified by 'user'*

function **getTotalOwed**(user) {

    var count\_neighbors = BlockchainSplitwise.getCountNeighbors.**call**(user);

    var neighbor;

    var total = 0;

    while(count\_neighbors > 0){

        neighbor = BlockchainSplitwise.getNeighbor.**call**(user, count\_neighbors - 1);

        total += BlockchainSplitwise.lookup.**call**(web3.eth.defaultAccount, neighbor) \* 1;

        count\_neighbors -= 1;

    }

    return total;

}

*// TODO: Get the last time this user has sent or received an IOU, in seconds since Jan. 1, 1970*

*// Return null if you can't find any activity for the user.*

*// HINT: Try looking at the way 'getAllFunctionCalls' is written. You can modify it if you'd like.*

function **getLastActive**(user) {

    return BlockchainSplitwise.get\_last\_active.**call**(user);

}

*// TODO: add an IOU ('I owe you') to the system*

*// The person you owe money is passed as 'creditor'*

*// The amount you owe them is passed as 'amount'*

function **add\_IOU**(creditor, amount) {

    var path = **doBFS**(creditor, web3.eth.defaultAccount, getNeighbors);

**log**("path", path);

    if(path === null){

        BlockchainSplitwise.add\_IOU.**sendTransaction**(creditor, amount, {gasPrice : 2000, gas : 500000});

    }else{

        var min\_edge = amount;

        var i = 1;

        while(i < path.length){

      var edge = BlockchainSplitwise.lookup.**call**(path[i - 1], path[i]) \* 1;

            if(min\_edge > edge){

                min\_edge = edge;

            }

            i += 1;

        }

**log**("min\_edge", min\_edge);

        i = path.length;

        while(i > 0){

            BlockchainSplitwise.add\_IOU.**sendTransaction**(path[i - 1], min\_edge, {from : path[i], gasPrice : 2000, gas : 500000});

            i -= 1;

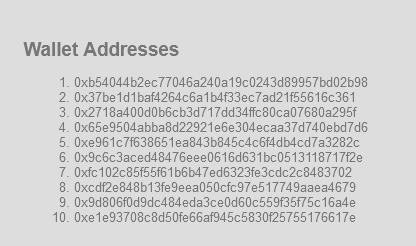
        }

        BlockchainSplitwise.add\_IOU.**sendTransaction**(creditor, amount - min\_edge, {gasPrice : 2000, gas : 500000});

    }

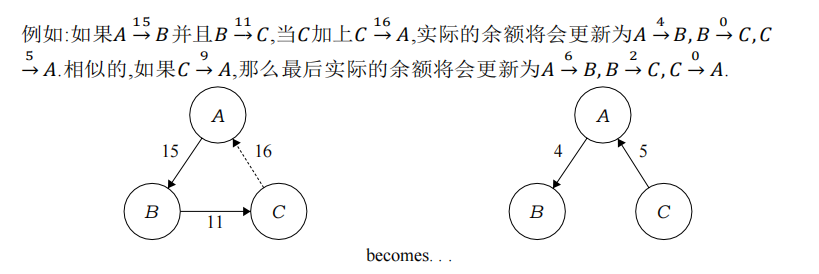
}

#### 三、运行截图

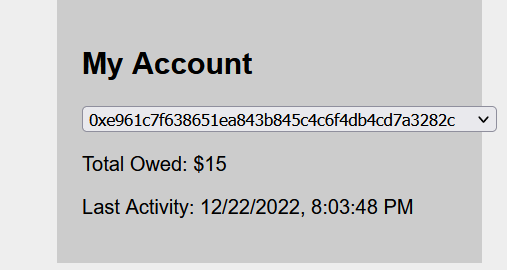
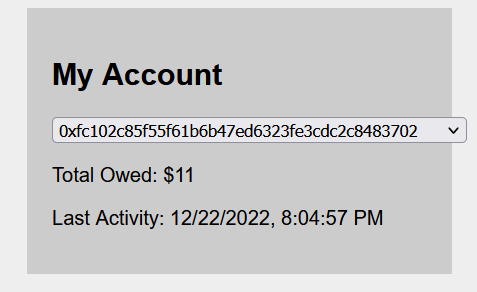


##### 三人的情况

①选取地址5号、7号、8号三个地址进行示范，初始三人total owed皆为0。



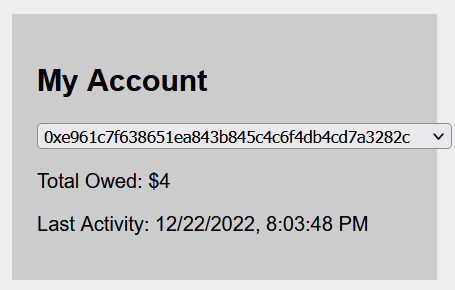
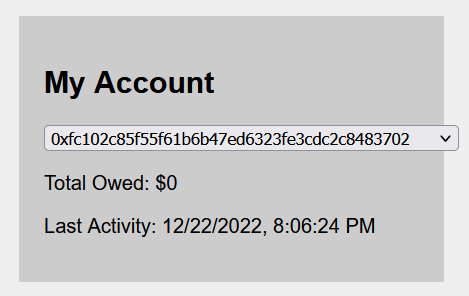
②令5号欠7号15，7号欠8号11，如下图。

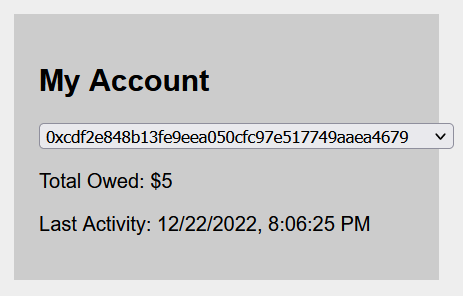
 

③令8号欠5号16。

更新余额后的结果应为：5号欠7号4，7号欠8号0，8号欠5号5。

运行结果如下图，符合推算的结果。

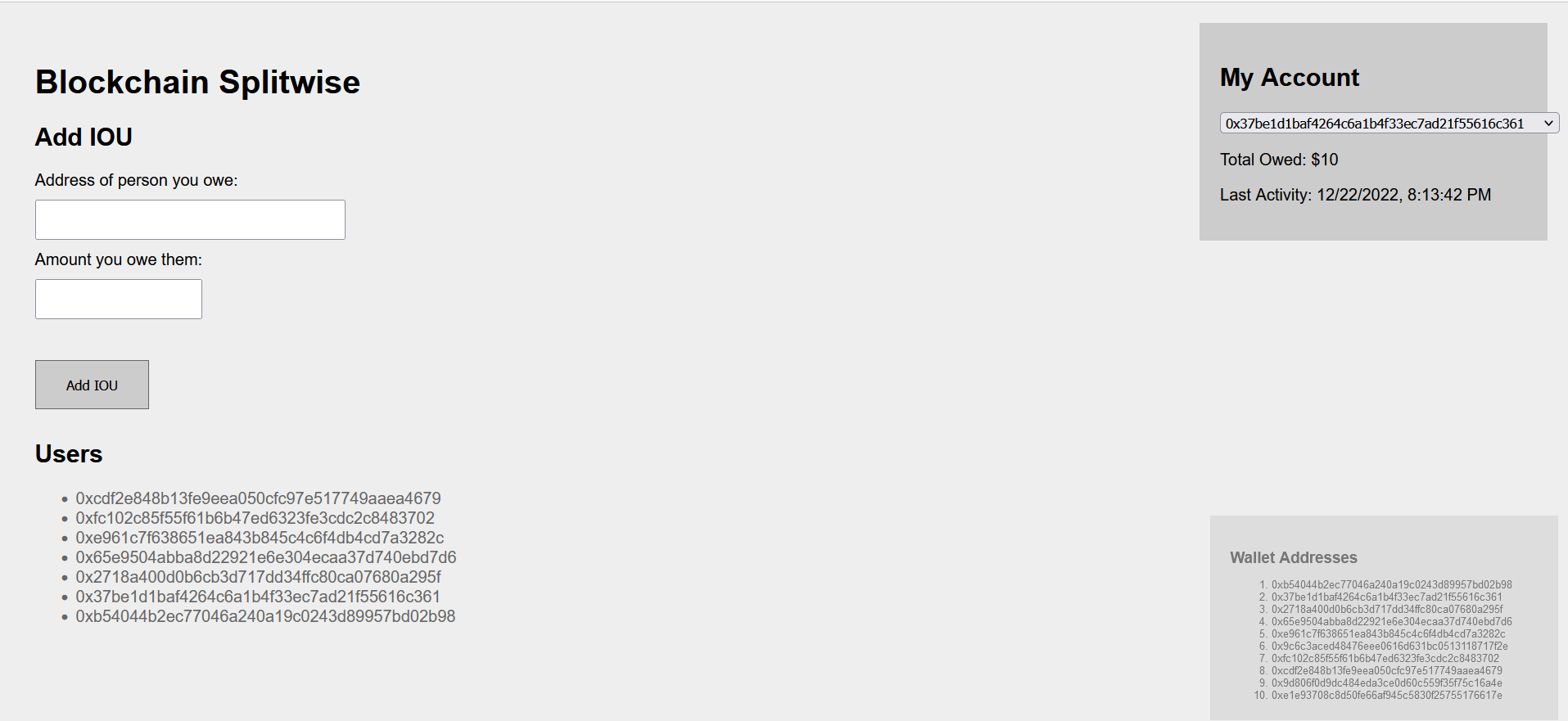
 



##### 两人的情况

以2号4号地址之间进行演示。

①设置2号欠4号10，如下图。



②设置4号欠2号10。那么此时4号与2号应该是互不相欠的状态。

运行代码，查询结果如下图，符合。

