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

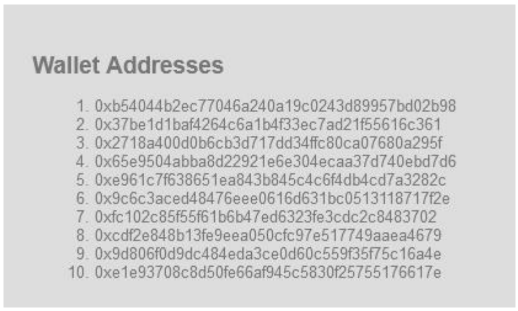
```
var abi =[
    {
        "inputs": [
            {
                "internalType": "address",
                "name": "creditor",
                "type": "address"
            },
            {
                "internalType": "uint32",
                "name": "amount",
                "type": "uint32"
            }
        ],
        "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": [
```

```json
            {
                "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

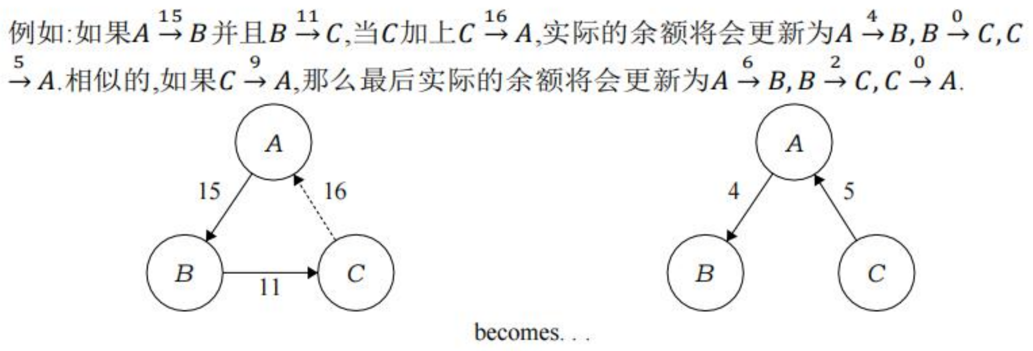


Wallet Addresses

1. 0xb54044b2ec77046a240a19c0243d89957bd02b98
2. 0x37be1d1baf4264c6a1b4f33ec7ad21f55616c361
3. 0x2718a400d0b6cb3d717dd34ffc80ca07680a295f
4. 0x65e9504abba8d22921e6e304ecaa37d740ebd7d6
5. 0xe961c7f638651ea843b845c4c6f4db4cd7a3282c
6. 0x9c6c3aced48476eee0616d631bc0513118717f2e
7. 0xfc102c85f55f61b6b47ed6323fe3cdc2c8483702
8. 0xcdf2e848b13fe9eea050cfc97e517749aaea4679
9. 0x9d806f0d9dc484eda3ce0d60c559f35f75c16a4e
10. 0xe1e93708c8d50fe66af945c5830f25755176617e

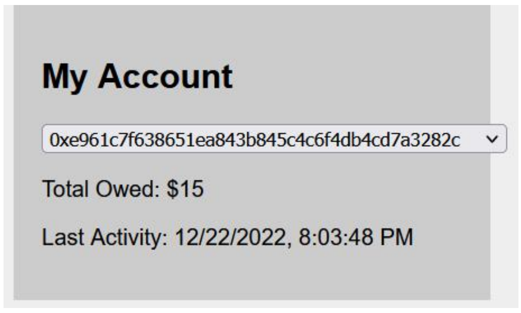
三人的情况

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。



例如:如果$A \xrightarrow{15} B$并且$B \xrightarrow{11} C$,当C加上$C \xrightarrow{16} A$,实际的余额将会更新为$A \xrightarrow{4} B, B \xrightarrow{0} C, C \xrightarrow{5} A$.相似的,如果$C \xrightarrow{9} A$,那么最后实际的余额将会更新为$A \xrightarrow{6} B, B \xrightarrow{2} C, C \xrightarrow{0} A$.

②令 5 号欠 7 号 15，7 号欠 8 号 11，如下图。

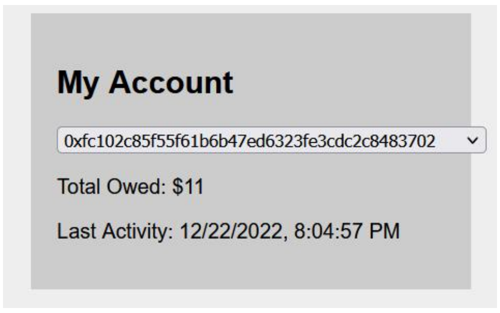


My Account

0xe961c7f638651ea843b845c4c6f4db4cd7a3282c

Total Owed: $15

Last Activity: 12/22/2022, 8:03:48 PM



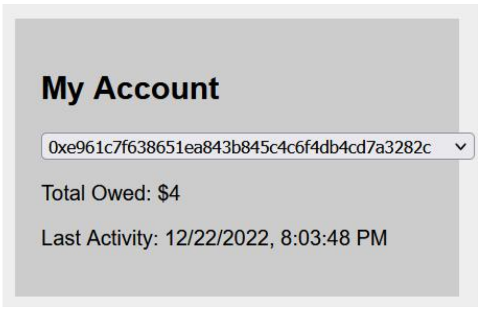
My Account

0xfc102c85f55f61b6b47ed6323fe3cdc2c8483702

Total Owed: $11

Last Activity: 12/22/2022, 8:04:57 PM

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。

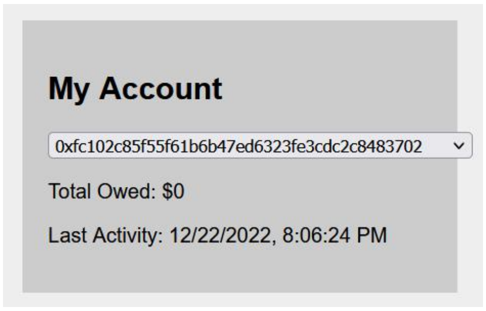


My Account

0xe961c7f638651ea843b845c4c6f4db4cd7a3282c

Total Owed: $4

Last Activity: 12/22/2022, 8:03:48 PM

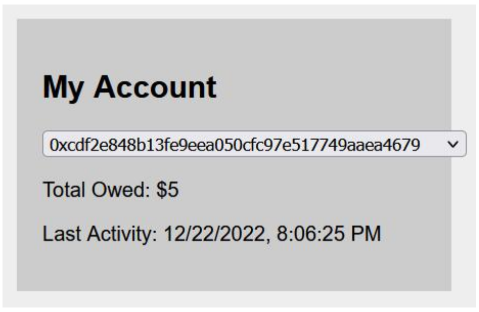


My Account

0xfc102c85f55f61b6b47ed6323fe3cdc2c8483702

Total Owed: $0

Last Activity: 12/22/2022, 8:06:24 PM



My Account

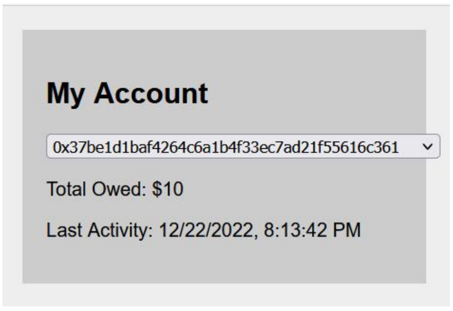
0xcdf2e848b13fe9eea050cfc97e517749aaea4679

Total Owed: $5

Last Activity: 12/22/2022, 8:06:25 PM

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。



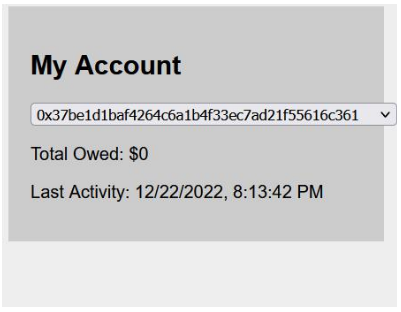
My Account

0x37be1d1baf4264c6a1b4f33ec7ad21f55616c361

Total Owed: $10

Last Activity: 12/22/2022, 8:13:42 PM

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。

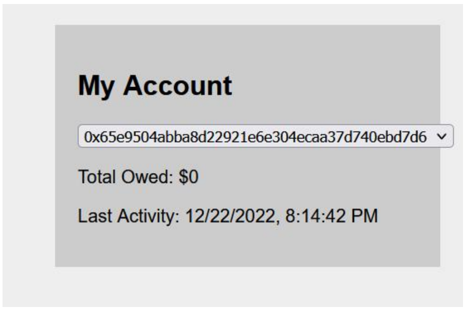


My Account

0x37be1d1baf4264c6a1b4f33ec7ad21f55616c361

Total Owed: $0

Last Activity: 12/22/2022, 8:13:42 PM



My Account

0x65e9504abba8d22921e6e304ecaa37d740ebd7d6

Total Owed: $0

Last Activity: 12/22/2022, 8:14:42 PM