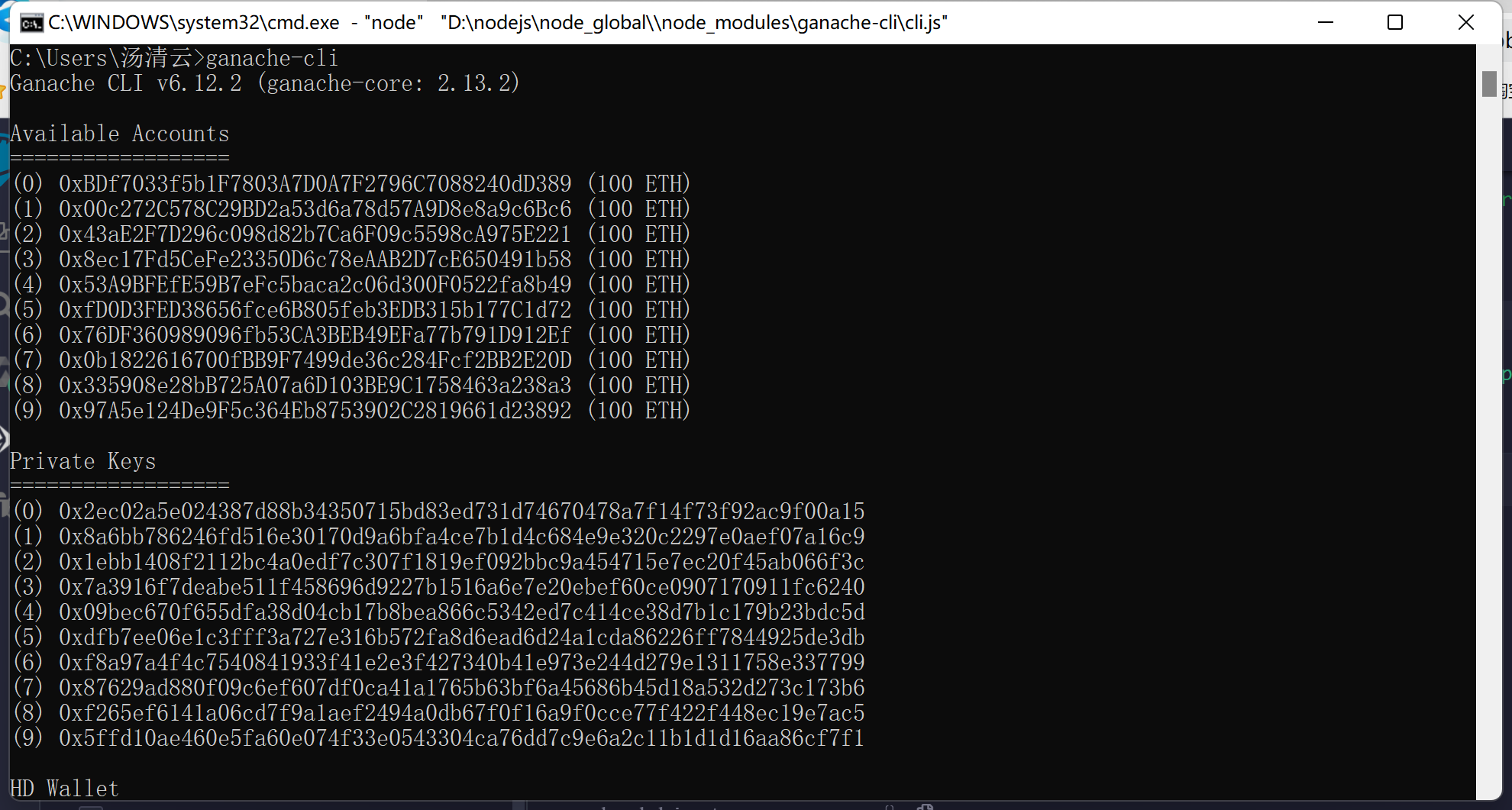
区块链第五次实验报告

2010535赵健坤 2013536汤清云

【注】在实验过程中我们发现原有代码前端框架同步存在严重问题，于是换用了2022秋斯坦福大学CS251的代码框架，详见<https://cs251.stanford.edu>

【实验过程】

1. 安装Ganache CLI，在本地命令行窗口输入ganache-cli来运行节点，Ctrl+C来结束节点，其运行结果如下图：



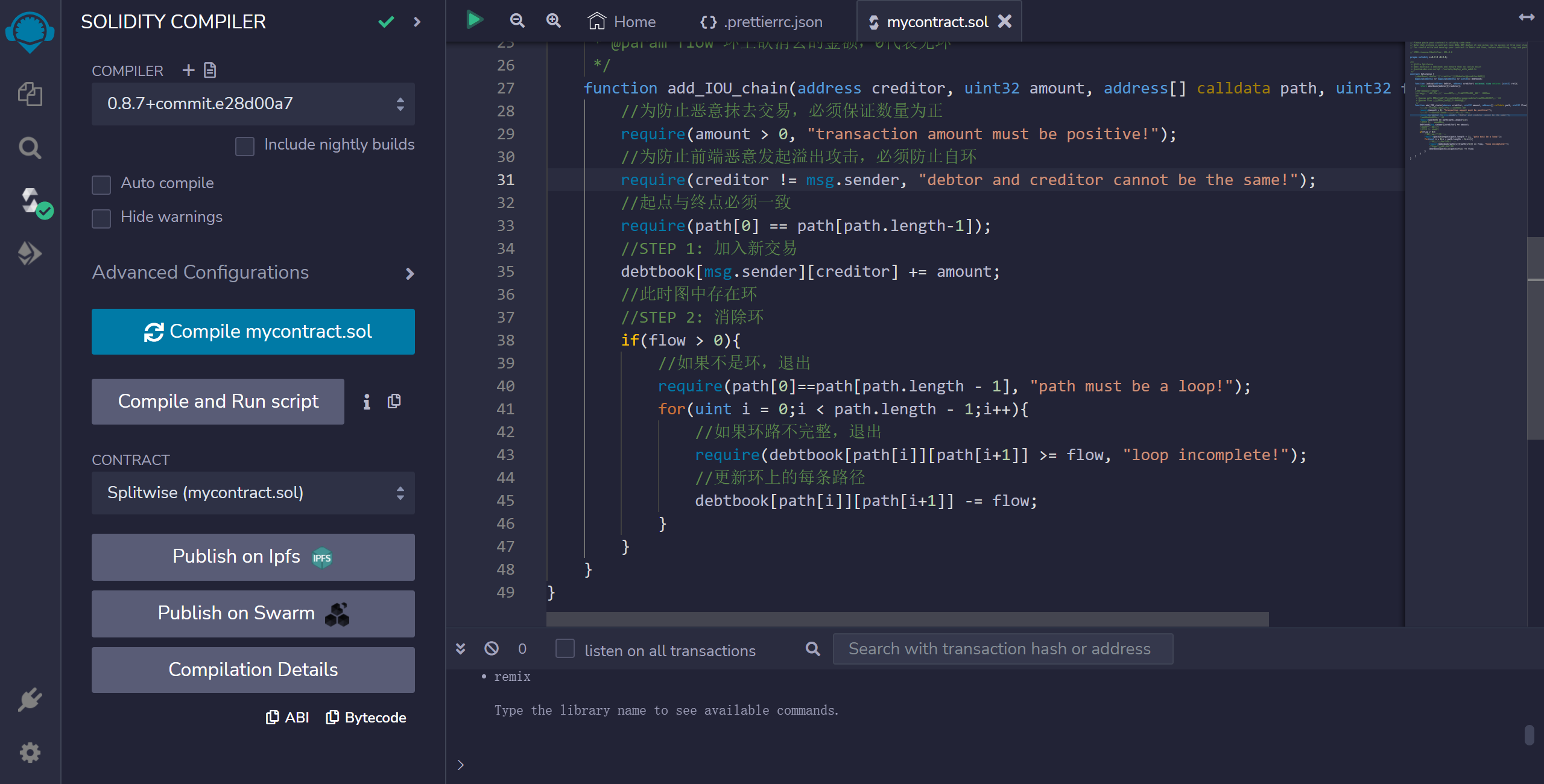
1. 登录remix网站（https://remix.ethereum.org），新建mycontract.sol文件，代码具体如下：

|  |
| --- |
| // Please paste your contract's solidity code here  // Note that writing a contract here WILL NOT deploy it and allow you to access it from your client  // You should write and develop your contract in Remix and then, before submitting, copy and paste it here  // SPDX-License-Identifier: GPL-3.0  pragma solidity >=0.7.0 <0.9.0;  /\*\*   \* @title Splitwise   \* @dev maintain a debtbook and ensure that no cycles exist   \* @custom:dev-run-script ./scripts/deploy\_with\_web3.ts   \*/  contract Splitwise {      //debtbook['debtor']['creditor']保存debtor欠creditor的金额      mapping(address => mapping(address => uint32)) debtbook;      function lookup(address debtor, address creditor) external view returns (uint32 ret){          return debtbook[debtor][creditor];      }      //先插入新边再消除环      //插入边和删除环必须在一个函数内进行，否则会破坏原子性造成错误      /\*\*       \* @param path 欲消去的环，方向由debtor指向creditor，起点与终点必须一致       \* @param flow 环上欲消去的金额，0代表无环       \*/      function add\_IOU\_chain(address creditor, uint32 amount, address[] calldata path, uint32 flow) external{          //为防止恶意抹去交易，必须保证数量为正          require(amount > 0, "transaction amount must be positive!");          //为防止前端恶意发起溢出攻击，必须防止自环          require(creditor != msg.sender, "debtor and creditor cannot be the same!");          //起点与终点必须一致          require(path[0] == path[path.length-1]);          //STEP 1: 加入新交易          debtbook[msg.sender][creditor] += amount;          //此时图中存在环          //STEP 2: 消除环          if(flow > 0){              //如果不是环，退出              require(path[0]==path[path.length - 1], "path must be a loop!");              for(uint i = 0;i < path.length - 1;i++){                  //如果环路不完整，退出                  require(debtbook[path[i]][path[i+1]] >= flow, "loop incomplete!");                  //更新环上的每条路径                  debtbook[path[i]][path[i+1]] -= flow;              }          }      }  } |

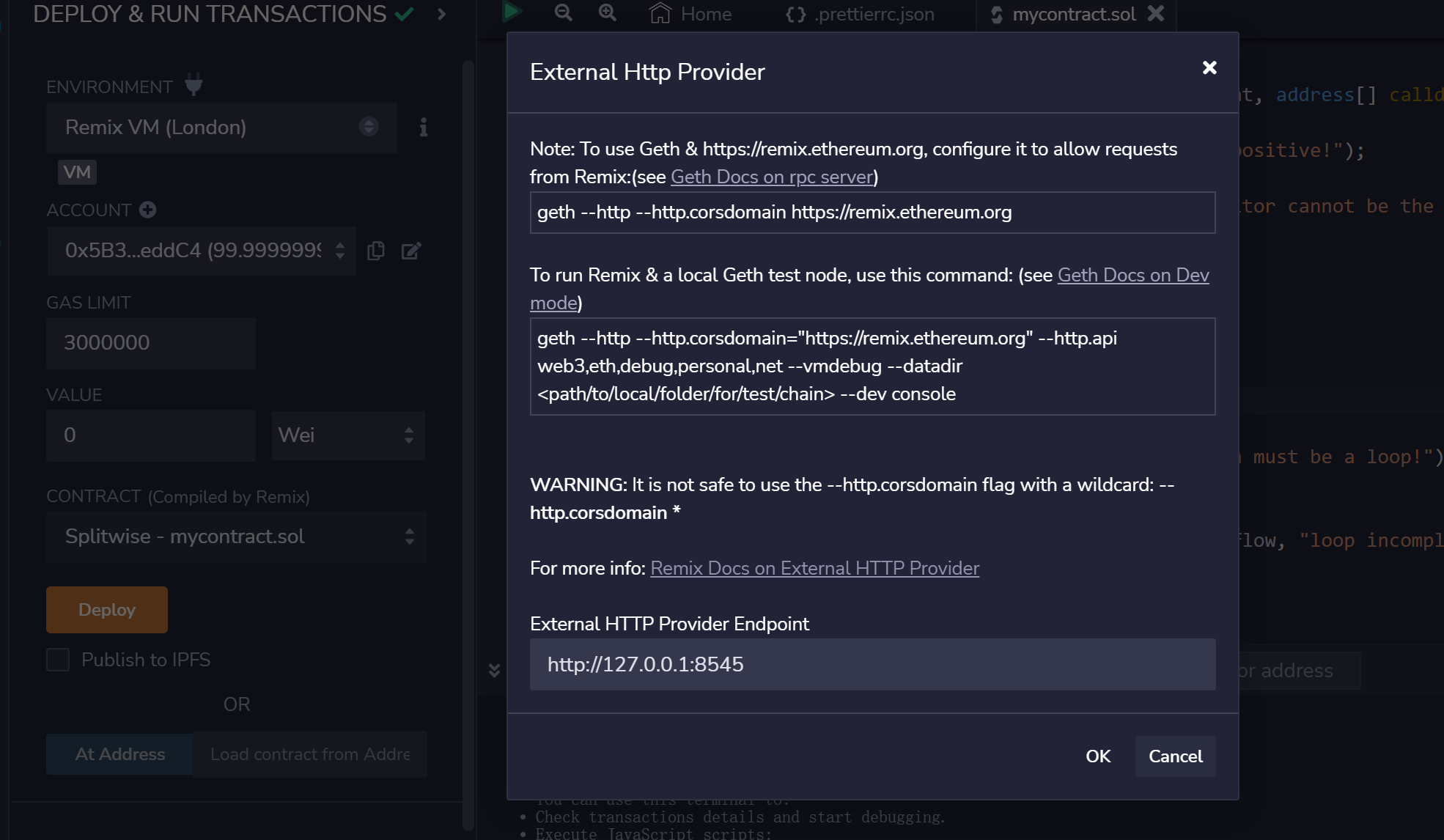
1. 前端代码编写如下：

|  |
| --- |
| 新增helper function |
| // TODO: Add any helper functions here!  var debtBook = {};  async function addLocalIOU(debtor, creditor, amount){      if(!(debtor in debtBook)) debtBook[debtor] = {};      if(!(creditor in debtBook[debtor])) debtBook[debtor][creditor] = 0;      debtBook[debtor][creditor] += amount;  }  function resolve\_cycle(path, flow) {      if(flow == 0) return;      //路径开始与结尾必须相同      if(path[0]!=path[path.length-1]){          throw new Error("The starting node and the end node must be the same!");      }      for(var i = 0; i < path.length - 1; i++) {          if(debtBook[path[i]][path[i+1]]<flow){              throw new Error("Flow is too large!");          }          debtBook[path[i]][path[i+1]] -= flow;      }  }  async function getDebtBook(){      //clear historical data      debtBook = {}      var rawdata = await getAllFunctionCalls(contractAddress, "add\_IOU\_chain");      //sort by timestamp to avoid inconsistency      rawdata.sort((a, b) => (a.t > b.t) ? 1 : -1)      for (let index = 0; index < rawdata.length; index++) {          const call = rawdata[index];          addLocalIOU(call.from.toLowerCase(), call.args[0].toLowerCase(), parseInt(call.args[1]));          resolve\_cycle(call.args[2], call.args[3]);      }  }  async function getNeighbors(node){      if(!(node in debtBook)) return [];      return Object.keys(debtBook[node]);  }  function get\_iou\_value(debtor, creditor) {      debtor = debtor.toLowerCase();      creditor = creditor.toLowerCase();      if(!(debtor in debtBook)) return 0;      if(!(creditor in debtBook[debtor])) return 0;      return debtBook[debtor][creditor];  }  function getMaxFlow(path){      if(path.length<3){//自环无效          return 0;      }      var maxFlow = get\_iou\_value(path[0], path[1]);      for(var i = 1; i < path.length-1; i++) {          let value = get\_iou\_value(path[i], path[i+1]);          if(value < maxFlow) maxFlow = value;      }      return maxFlow;  } |
| getUsers函数实现 |
| async function getUsers() {      //认定everyone who has ever sent or received an IOU为全部users      transactions = await getAllFunctionCalls(contractAddress, 'add\_IOU\_chain');      var users = new Set();      for (let index = 0; index < transactions.length; index++) {          const txn = transactions[index];          users.add(txn.from.toLowerCase());          users.add(txn.args[0]);//args是数组型      }      return Array.from(users);  } |
| Gettotalowed函数实现 |
| async function getTotalOwed(user) {      await getDebtBook();      var sum = 0;      if(!(user.toLowerCase() in debtBook)){          return 0;      }      for(let amount of Object.values(debtBook[user.toLowerCase()])) {          sum += amount;      }      return sum;  } |
| getlastActive函数实现 |
| async function getLastActive(user) {      var rawdata = await getAllFunctionCalls(contractAddress, "add\_IOU\_chain");      var last\_timestamp = null;      for (let index = 0; index < rawdata.length; index++) {          const call = rawdata[index];          if((call.from.toLowerCase() === user.toLowerCase() || call.args[0].toLowerCase() === user.toLowerCase())              && (last\_timestamp === null || call.t > last\_timestamp)) {              last\_timestamp = call.t;          }      }      return last\_timestamp;  } |
| AddIOU实现 |
| async function add\_IOU(creditor, amount) {      await getDebtBook();      var debtor = web3.eth.defaultAccount.toLowerCase();      addLocalIOU(debtor.toLowerCase(), creditor.toLowerCase(), parseInt(amount));      var cycle = await doBFS(debtor.toLowerCase(), debtor.toLowerCase(), getNeighbors);      var flow = await getMaxFlow(cycle);      return BlockchainSplitwise.methods.add\_IOU\_chain(creditor.toLowerCase(), amount, cycle, flow).send({from: debtor, gas:3000000});  } |

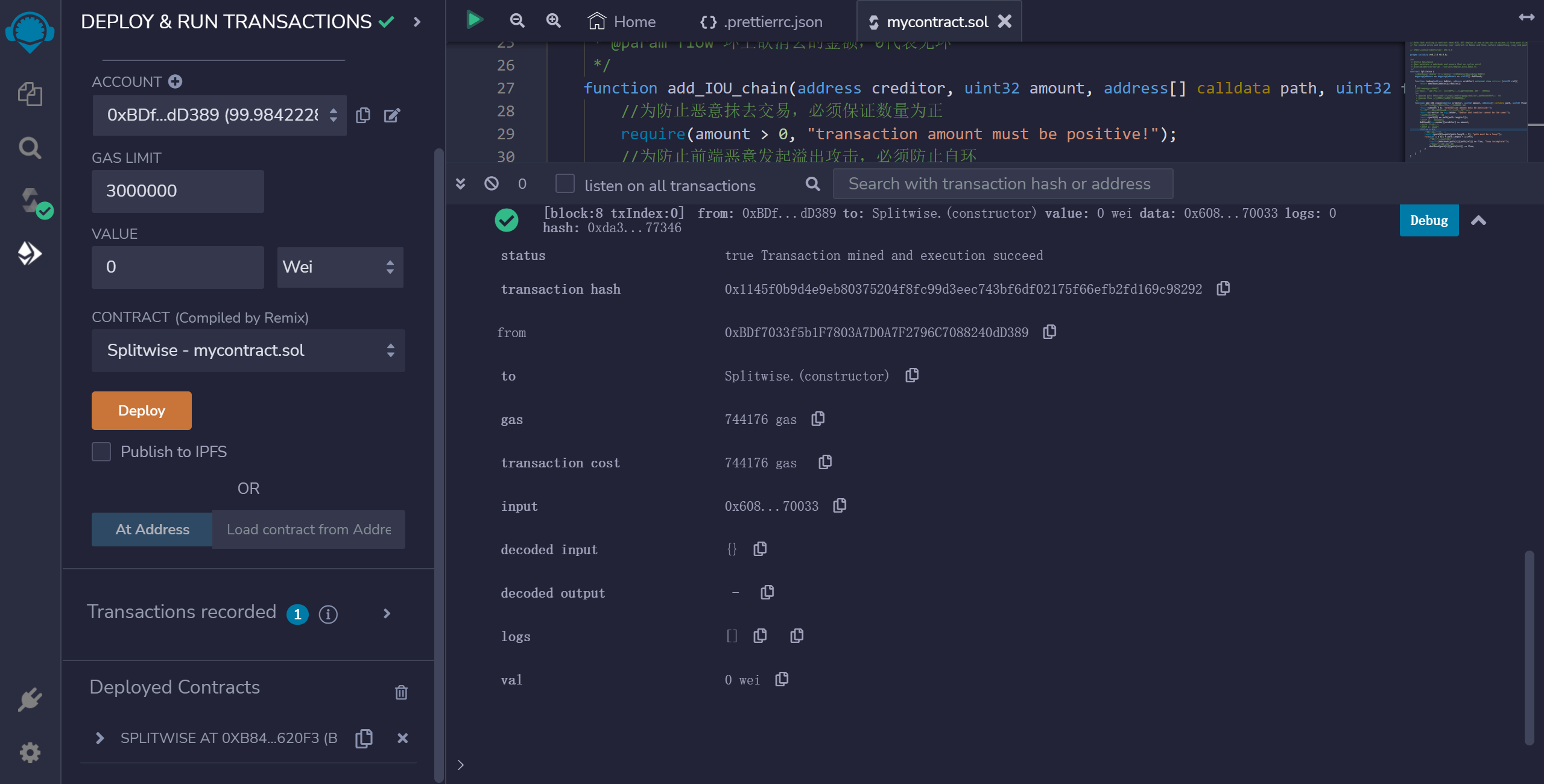
1. 在SOLIDITY选项卡中编译所写的文件，出现绿色的勾，说明编译成功。



1. 在Deploy & Run选项卡中设置环境为External Http Provider，端点设置为http://localhost:8545

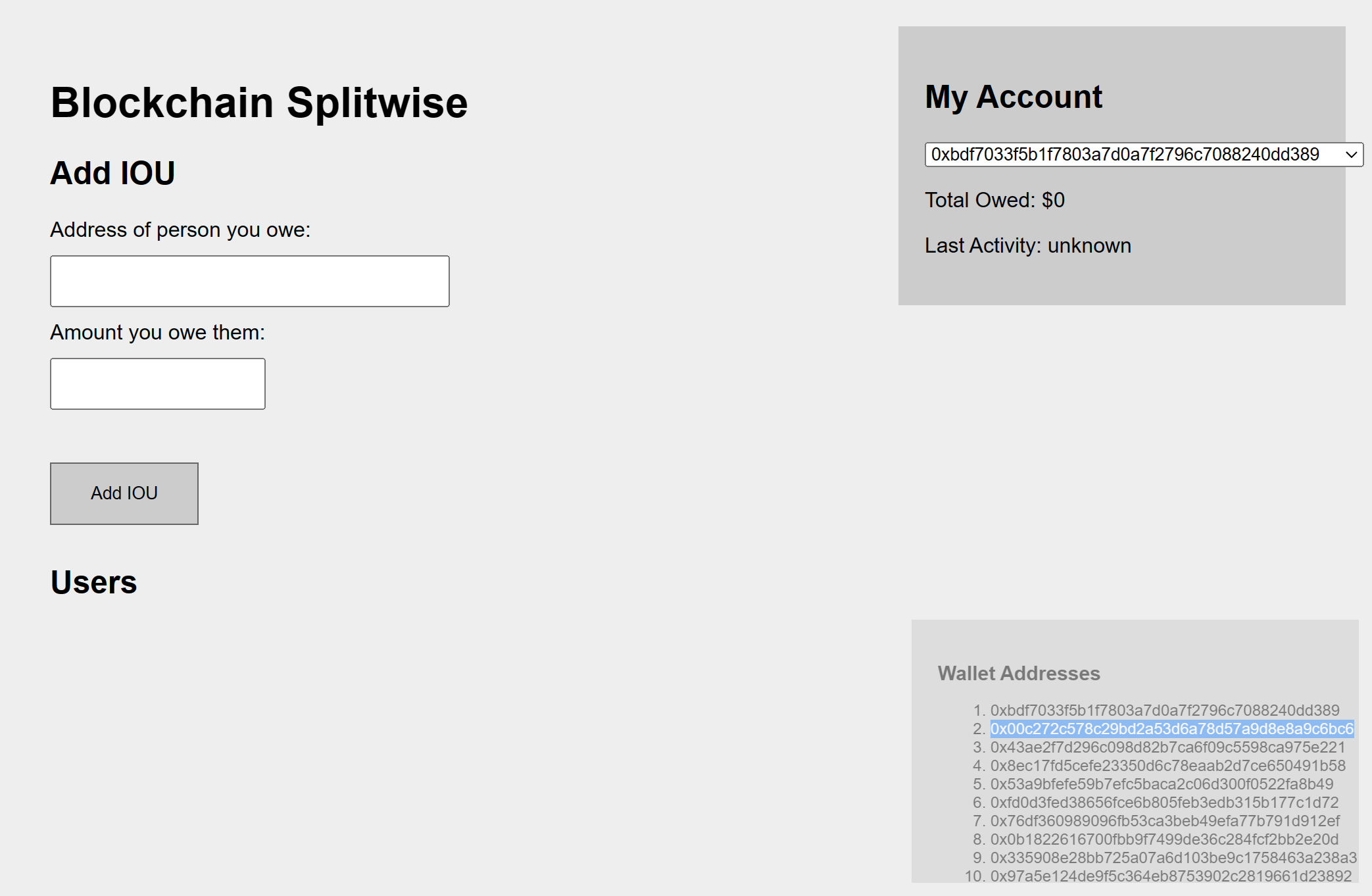


1. 运行节点并进行测试：

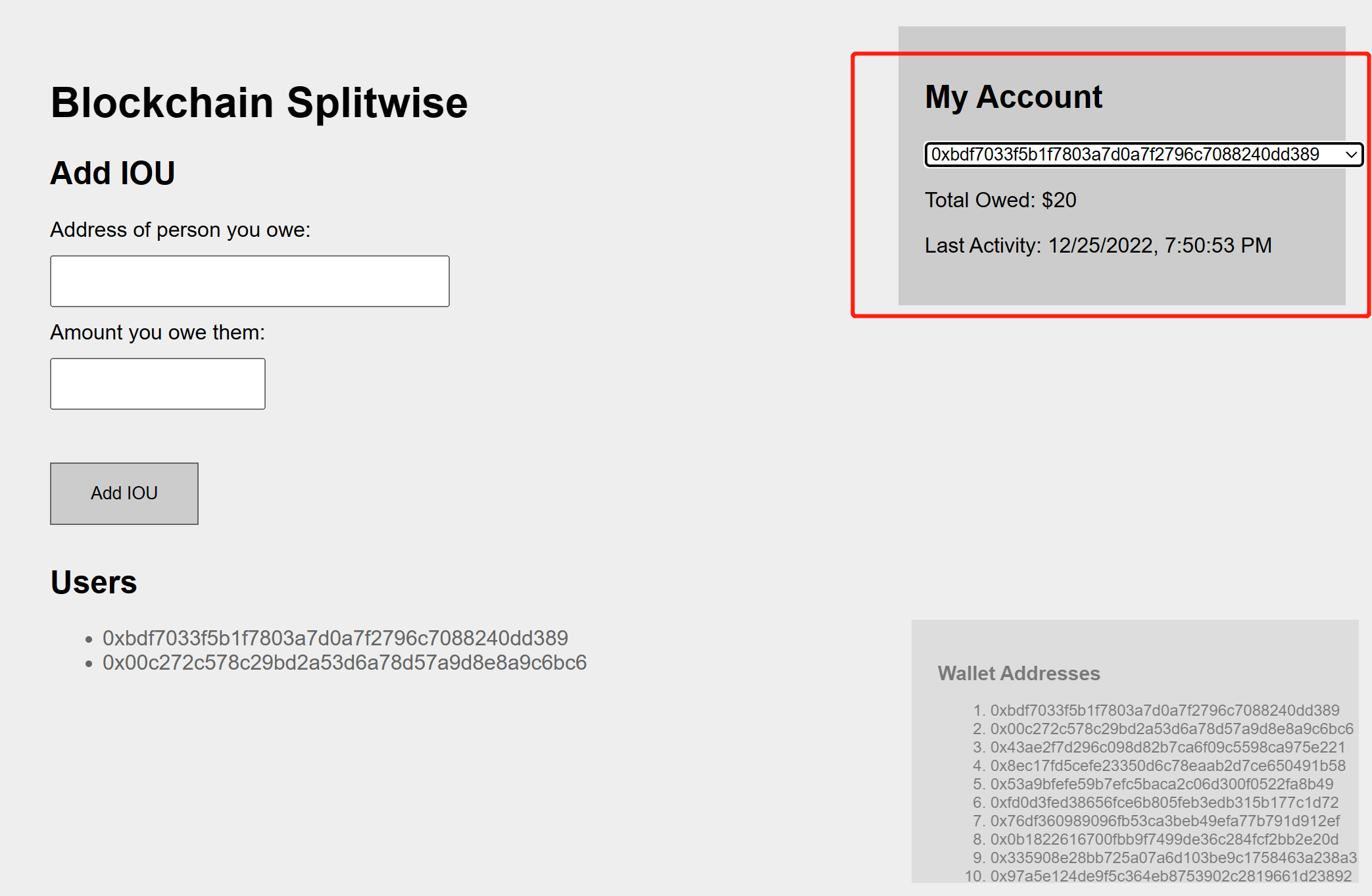


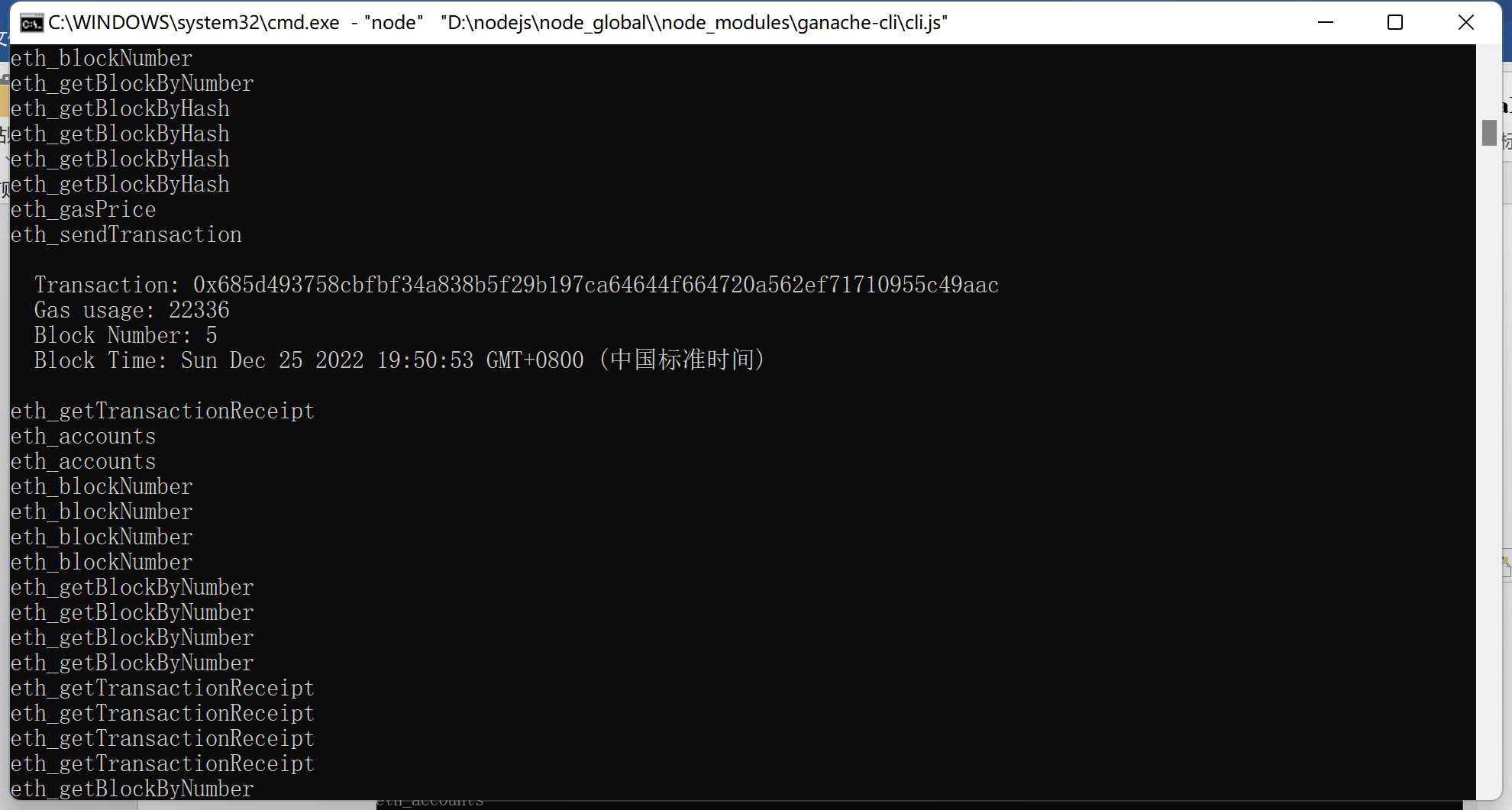
第一步：1号欠2号20元：

之前：



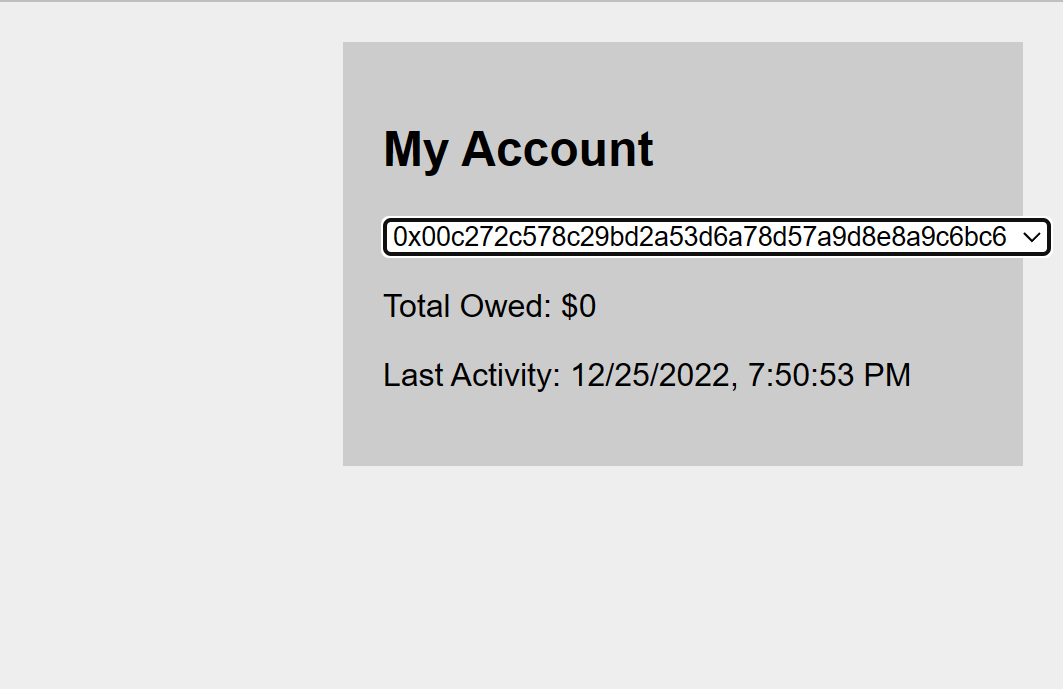
之后：



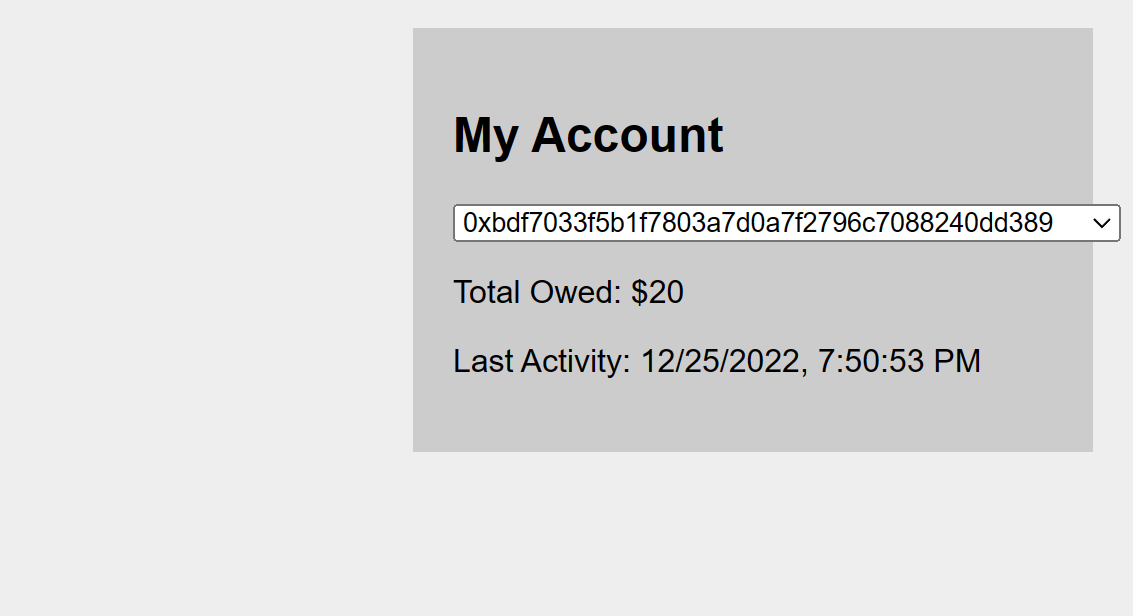


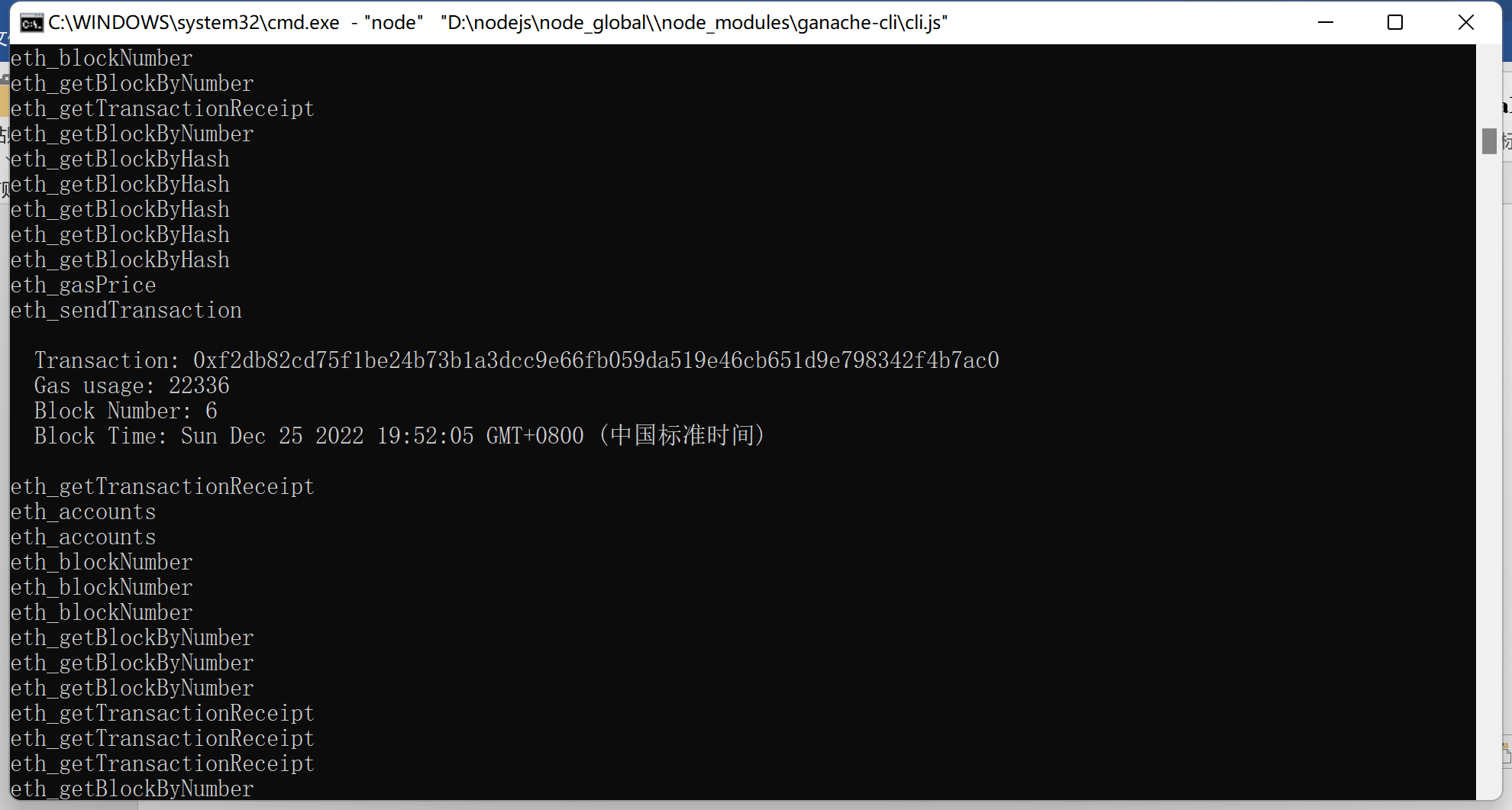
第二步，2号欠3号20元：

之前：



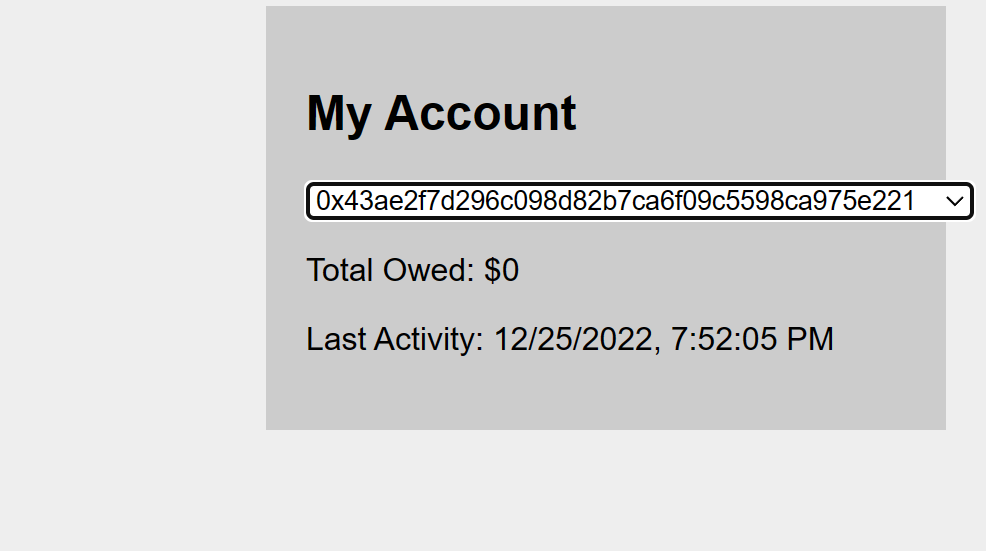
之后：



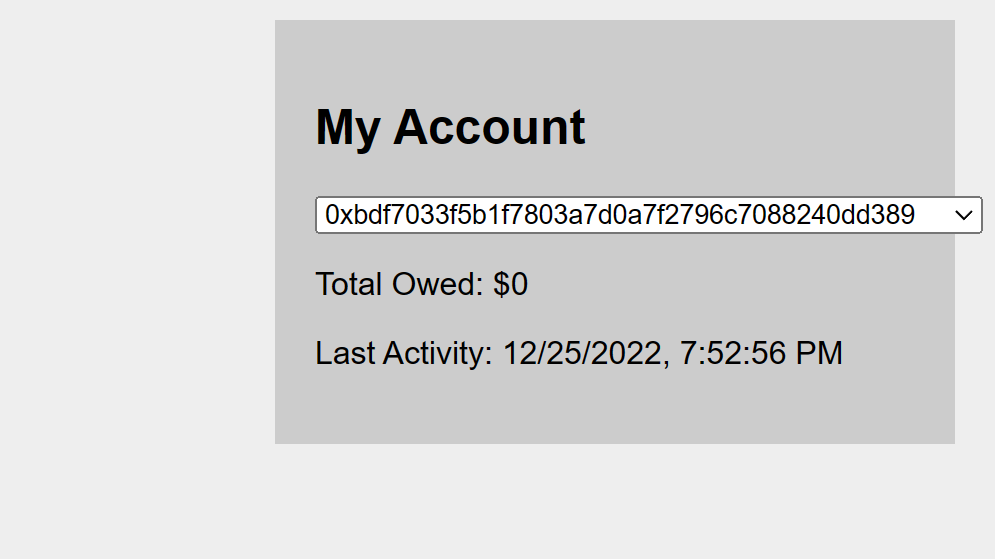


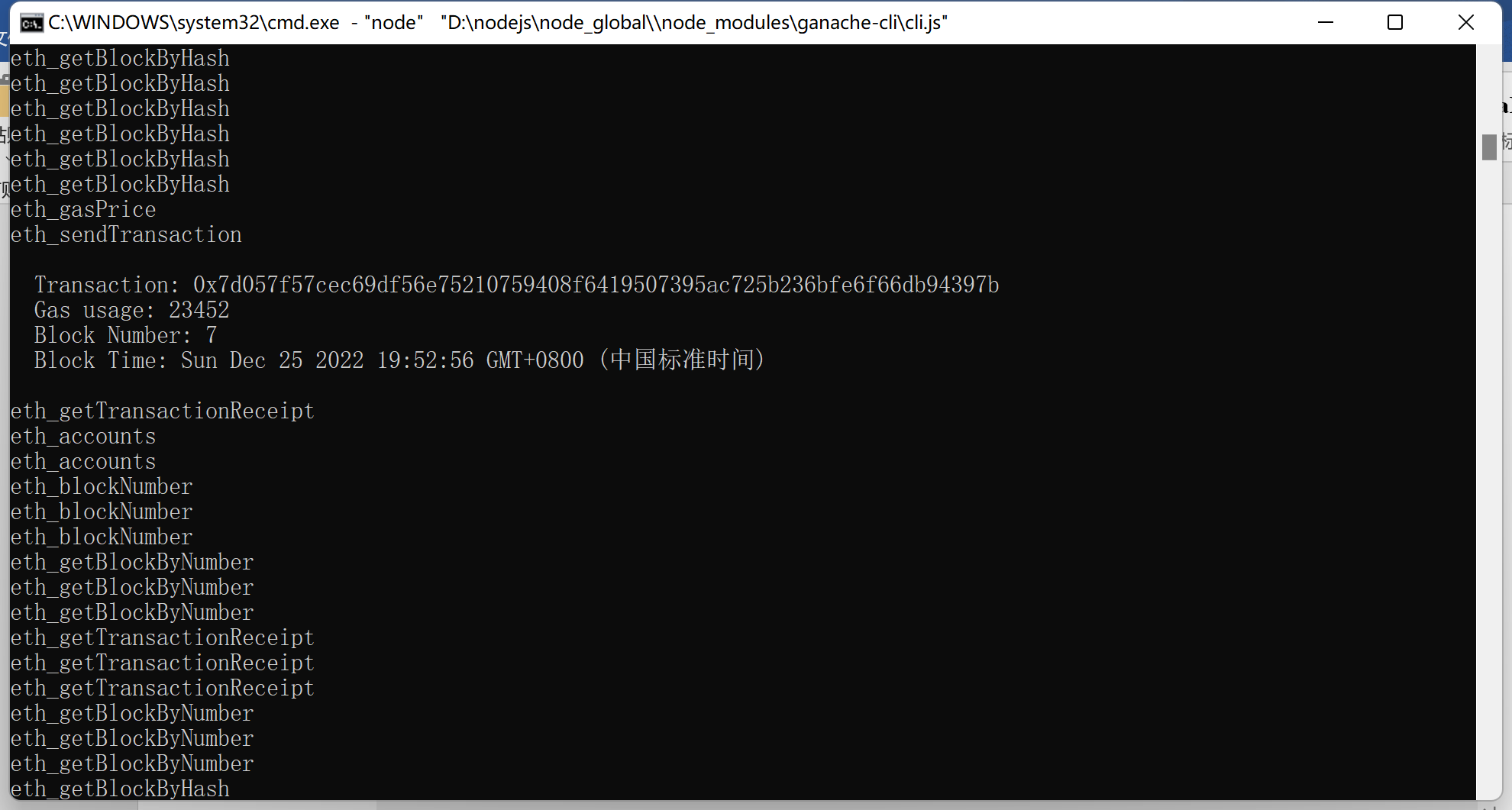
第三步：3号欠1号20元：

之前：

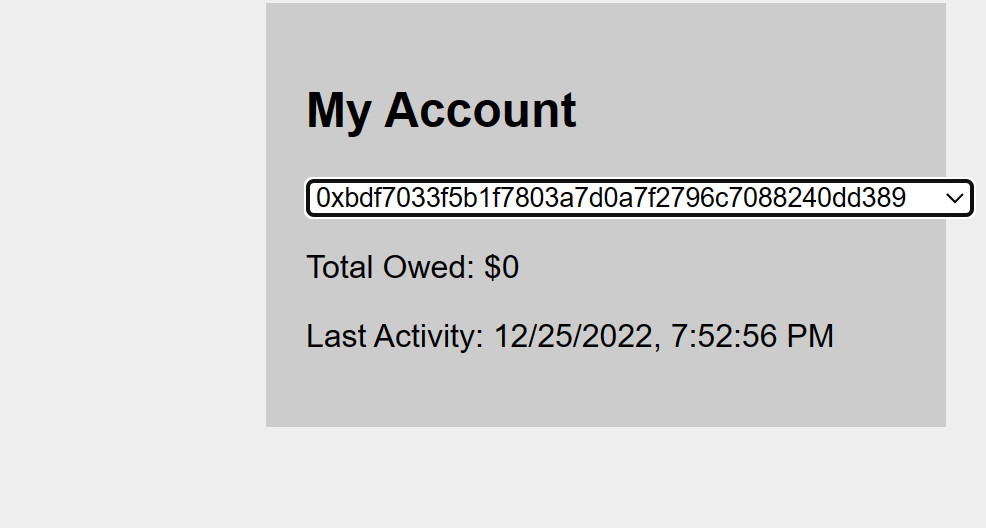


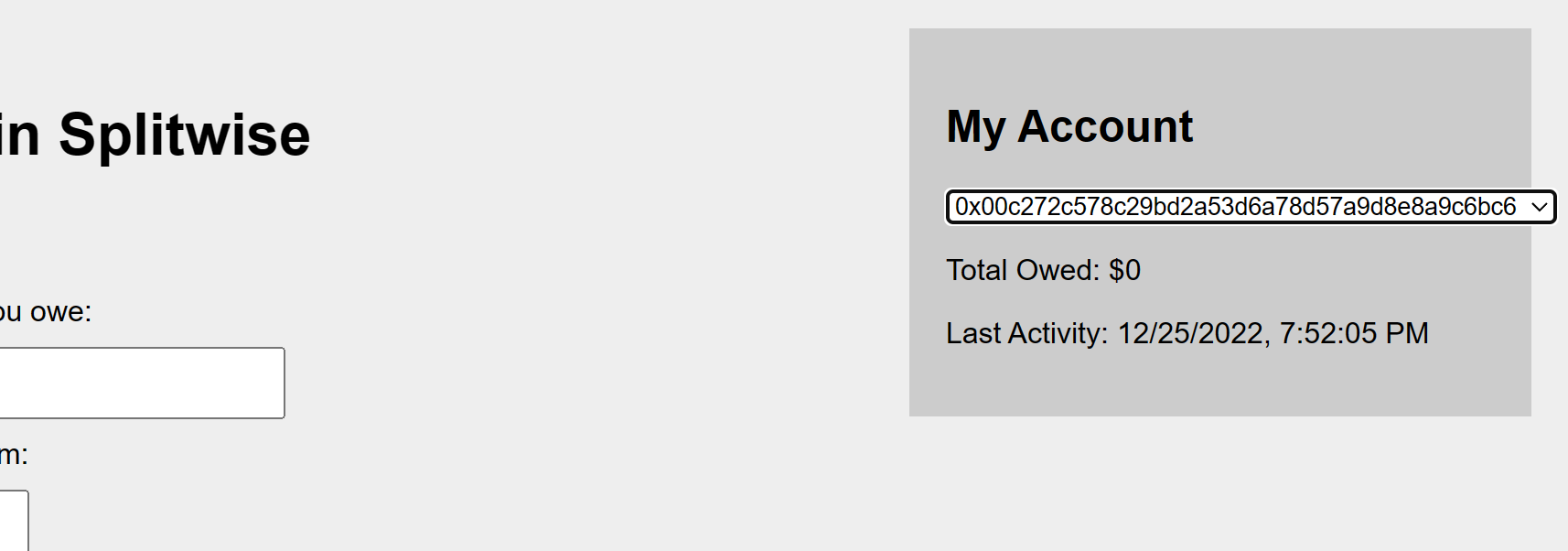
之后：

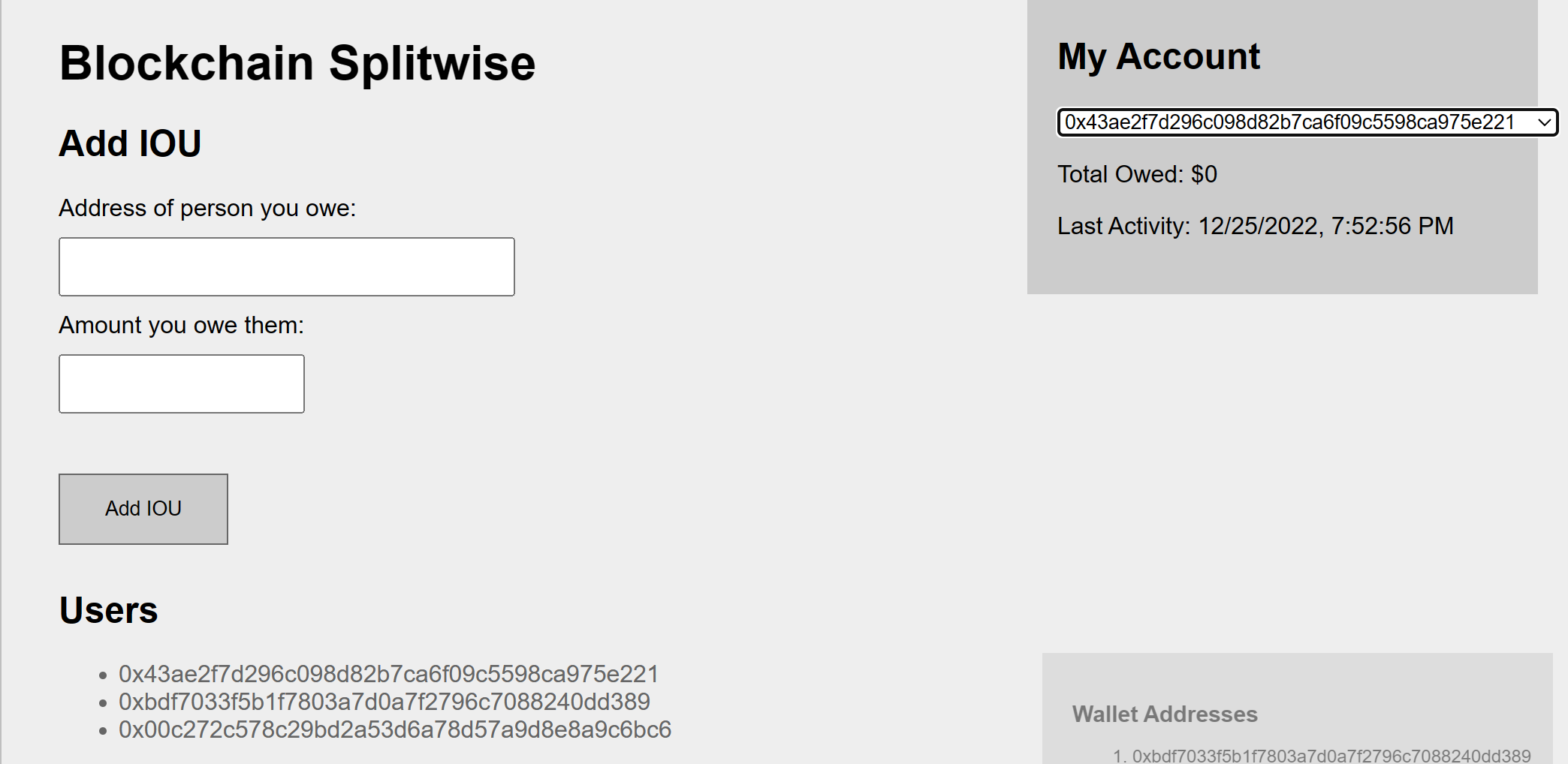




交易提交后1~3号账户信息如下：







可见形成的交易环消除，实验成功。