**周亚男 2020131062**

## 完善合约代码

CPAMM.sol

// SPDX-License-Identifier: MIT

pragma solidity 0.8.16;

import "@openzeppelin/contracts/token/ERC20/IERC20.sol";

contract CSAMM {

    // 表示dx 和 dy

    IERC20 immutable token0;

    IERC20 immutable token1;

    //用于存储持有币在合约的数量，对应X和Y

    uint256 public reserve0;

    uint256 public reserve1;

    uint256 public totalSupply;

    //balanceOf每个地址在流动性中持有的share

    mapping(address => uint256) public balanceOf;

    constructor(address \_token0, address \_token1) {

        token0 = IERC20(\_token0);

        token1 = IERC20(\_token1);

    }

    // 可以进行复用，用于更新余额表

    function \_update() private {

        reserve0 = token0.balanceOf(address(this));

        reserve1 = token1.balanceOf(address(this));

    }

    function \_sqrt(uint256 y) internal pure returns (uint256 z) {

        if (y > 3) {

            z = y;

            uint256 x = y / 2 + 1;

            while (x < z) {

                z = x;

                x = (y / x + x) / 2;

            }

        } else if (y != 0) {

            z = 1;

        }

    }

    function \_min(uint256 \_x, uint256 \_y) private pure returns (uint256) {

        return \_x > \_y ? \_y : \_x;

    }

    // 添加流动性，增加shared到目的地址

    function \_mint(address \_to, uint256 \_amount) private {

        // 此处补全

        balanceOf[\_to] += \_amount;

        totalSupply += \_amount;

    }

    function \_burn(address \_from, uint256 \_amount) private {

        // 此处补全

        balanceOf[\_from] -= \_amount;

        totalSupply -= \_amount;

    }

    function swap(

        address \_tokenIn,

        uint256 \_amountIn

    ) external returns (uint256 amountOut) {

        // 此处补全

        // 前两个进行验证

        require(\_amountIn > 0, "Invalid Amount");

        // 为交易所支持的币

        require(

            \_tokenIn == address(token0) || \_tokenIn == address(token1),

            "Invalid token"

        );

        // 判断是换的那种币

        bool isToken0 = \_tokenIn == address(token0);

        (IERC20 tokenIn, IERC20 tokenOut) = isToken0

            ? (token0, token1)

            : (token1, token0);

        (uint256 reserveIn, uint256 reserveOut) = isToken0

            ? (reserve0, reserve1)

            : (reserve1, reserve0);

        //转币到合约

        tokenIn.transferFrom(msg.sender, address(this), \_amountIn);

        // 计算输出的数量

        amountOut = (\_amountIn \* reserveOut) / (reserveIn + \_amountIn);

        // 转币给用户

        tokenOut.transfer(msg.sender, amountOut);

        // 更新自己的余额表

        \_update();

    }

    function addLiquidity(

        uint256 \_amount0,

        uint256 \_amount1

    ) external returns (uint256 shares) {

        // 此处补全

        // 避免输入无效值

        require(\_amount0 > 0 && \_amount1 > 0, "Invalid amount");

        // 把token0 和 token1 转入到合约

        token0.transferFrom(msg.sender, address(this), \_amount0);

        token1.transferFrom(msg.sender, address(this), \_amount1);

        // 计算并mint share 给用户（改）

        // 已经添加过流动性，验证是否满足比例

        if (reserve0 > 0 || reserve1 > 0) {

            require(\_amount0 \* reserve1 == \_amount1 \* reserve0, "dy != y/x");

        }

        // 还未被添加过，取平平方

        if (totalSupply == 0) {

            shares = \_sqrt(\_amount0 \* \_amount1);

            // 已经添加过

        } else {

            // 选择较小的值，比较安全

            shares = \_min(

                (\_amount0 \* totalSupply) / reserve0,

                (\_amount1 \* totalSupply) / reserve1

            );

        }

        require(shares > 0, "share is zero");

        \_mint(msg.sender, shares);

        // 更新余额表

        \_update();

    }

    // 根据share，将币转回给用户

    function removeLiquidity(

        uint256 \_shares

    ) external returns (uint256 amount0, uint256 amount1) {

        // 此处补全

        require(\_shares > 0, "Invalid shares");

        // 计算dx 和dy的数量

        amount0 = (\_shares \* reserve0) / totalSupply;

        amount1 = (\_shares \* reserve1) / totalSupply;

        // 销毁用户的share

        \_burn(msg.sender, \_shares);

        // 把两个币转回给用户

        token0.transfer(msg.sender, amount0);

        token1.transfer(msg.sender, amount1);

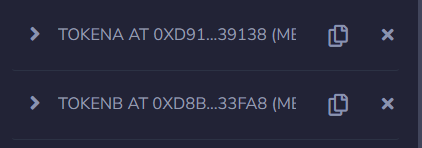
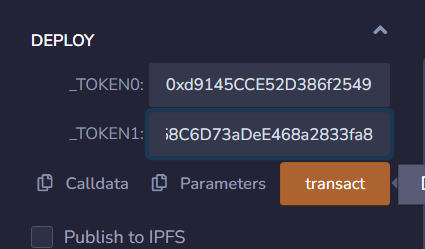
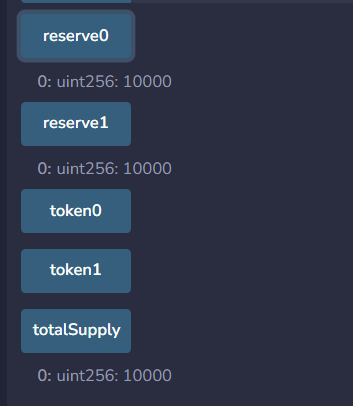
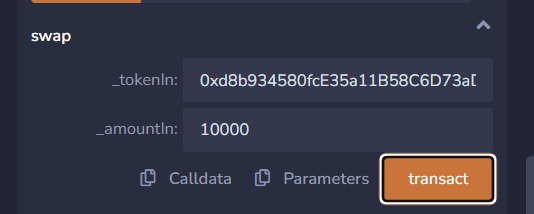
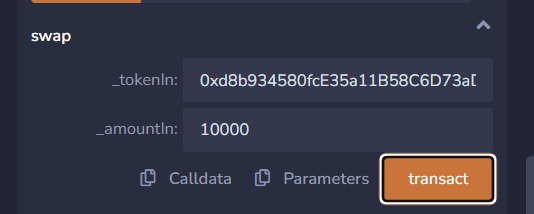
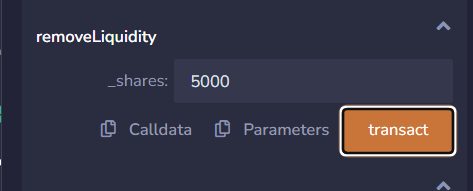
        // 更新余额表

        \_update();

    }

}

## 实验过程

1. 部署tokenA和tokenB  
   
2. 部署cpamm  
   
3. 查看状态  
   
4. 买10000tokenA  
   
5. 查看状态（梁老师的好像是错的？为什么是反的）  
   
6. 移除流动性  
   
7. 查看状态  
   