周亚男 2020131062

完善合约代码

TokenA.sol

```
// SPDX-License-Identifier: SEE LICENSE IN LICENSE
pragma solidity ^0.8.12;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import
"@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.
sol";

contract TokenA is ERC20, Ownable, ERC20Burnable {
    constructor() ERC20("TokenA", "TokenA") {}

    function mint(address reciever, uint256 amount) public
onlyOwner {
        _mint(reciever, amount);
    }

    function _burn(uint256 amount) public onlyOwner {
        burn(amount);
    }
}
```

TokenB.sol

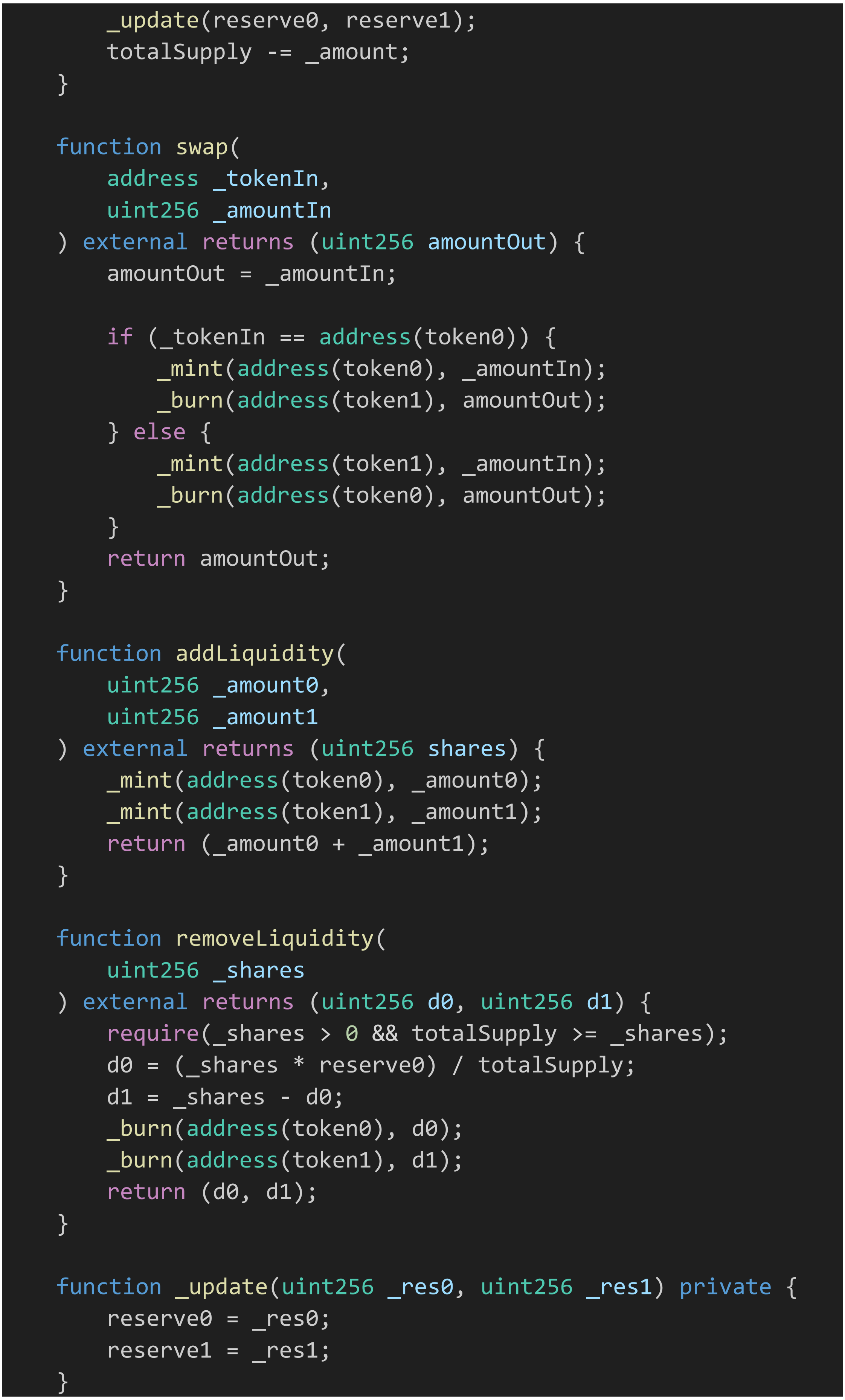
```
// SPDX-License-Identifier: SEE LICENSE IN LICENSE
pragma solidity ^0.8.12;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import
"@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.
sol";

contract TokenB is ERC20, Ownable, ERC20Burnable {
    constructor() ERC20("TokenB", "TokenB") {}
    function mint(address reciever, uint256 amount) public onlyOwner {
        _mint(reciever, amount);
    }

    function _burn(uint256 amount) public onlyOwner {
        burn(amount);
    }
}
```

002_CSAMM.sol

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.16;

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

contract CSAMM {
    IERC20 immutable token0;
    IERC20 immutable token1;

    uint public reserve0;
    uint public reserve1;

    uint public totalSupply;
    mapping(address => uint) public balanceOf;

    constructor(address _token0, address _token1) {
        token0 = IERC20(_token0);
        token1 = IERC20(_token1);
    }

    function _mint(address _to, uint _amount) private {

        require(_amount>0);
        balanceOf[_to] += _amount;
        totalSupply += _amount;
    }

    function _burn(address _from, uint _amount) private {

        require(_amount>0);
        require(balanceOf[_from]>=_amount);
        balanceOf[_from] -= _amount;
        totalSupply -= _amount;
    }

    function swap(
        address _tokenIn,
        uint _amountIn
    ) external returns (uint amountOut) {

        require(_amountIn>0);

        if (_tokenIn == address(token0)){
            token0.transferFrom(msg.sender, address(this), _amountIn);
            token1.transfer(msg.sender, amountOut);
            reserve0+=_amountIn;
            reserve1+=_amountIn;
        }else {
            token1.transferFrom(msg.sender, address(this), _amountIn);
            token0.transfer(msg.sender, _amountIn);
            reserve1 += _amountIn;
            reserve0 -= _amountIn;
        }

    }

    function addLiquidity(
        uint _amount0,
        uint _amount1
    ) external returns (uint shares) {

        require(_amount0>0&&_amount1>0);

        if (totalSupply==0){
            shares = _amount0+_amount1;
        }else {
            shares =
(_amount0+_amount1)*totalSupply/(reserve0+reserve1);

        }
        token0.transferFrom(msg.sender, address(this), _amount0);
        token1.transferFrom(msg.sender, address(this), _amount1);
        reserve0+=_amount0;
        reserve1+=_amount1;
        _mint(msg.sender, shares);
    }

    function removeLiquidity(uint _shares) external returns (uint d0, 
uint d1) {

        require(_shares>0);
        require((balanceOf[msg.sender]>=_shares));
        d0 = reserve0*_shares/totalSupply;
        d1 = reserve1*_shares/totalSupply;
        require(token0.balanceOf(address(this)) >= d0 && 
token1.balanceOf(address(this)) >= d1);
        token0.transfer(msg.sender, d0);
        token1.transfer(msg.sender, d1);
        reserve0 -= d0;
        reserve1 -= d1;
        _burn(msg.sender, _shares);
    }

    function _update(uint _res0, uint _res1) private {
        reserve0 = _res0;
        reserve1 = _res1;
    }
}
```

002_CSAMM.sol(错误版本，此时没理解 shares)

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.16;

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

contract CSAMM {
    IERC20 immutable token0;
    IERC20 immutable token1;

    uint256 public reserve0;
    uint256 public reserve1;

    uint256 public totalSupply;

    // mapping(address => uint) public balanceOf;
    constructor(address _token0, address _token1) {
        token0 = IERC20(_token0);
        token1 = IERC20(_token1);
    }

    function _mint(address _to, uint256 _amount) private {
        // 此处补全
        require(_to == address(token0) || _to == address(token1));
        require(_amount > 0);
        //mint 并不是 private 为什么不能调用
        // IERC20(_to).mint(msg.sender, address(this), _amount);
        if (_to == address(token0)) {
            reserve0 += _amount;
            token0.transferFrom(msg.sender, address(this), _amount);
        } else {
            reserve1 += _amount;
            token1.transferFrom(msg.sender, address(this), _amount);
        }
        _update(reserve0, reserve1);
        totalSupply += _amount;
    }

    function _burn(address _from, uint256 _amount) private {
        require(_from == address(token0) || _from == address(token1));
        require(
            _amount > 0 && IERC20(_from).balanceOf(address(this)) >= _amount
        );

        // IERC20(_from)._burn(address(this), _amount);

        if (_from == address(token0)) {
            reserve0 -= _amount;
            token0.transfer(msg.sender, _amount);
        } else {
            reserve1 -= _amount;
            token1.transfer(msg.sender, _amount);
        }
```

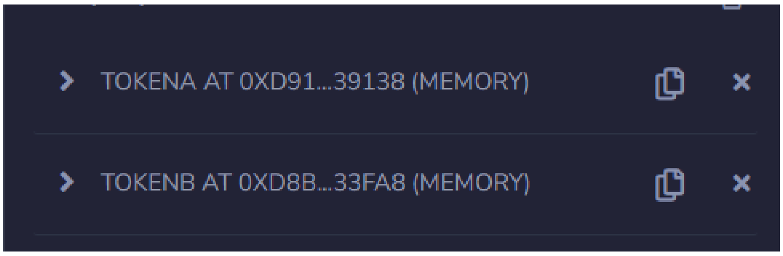


```
        _update(reserve0, reserve1);
        totalSupply -= _amount;
    }

    function swap(
        address _tokenIn,
        uint256 _amountIn
    ) external returns (uint256 amountOut) {
        amountOut = _amountIn;

        if (_tokenIn == address(token0)) {
            _mint(address(token0), _amountIn);
            _burn(address(token1), amountOut);
        } else {
            _mint(address(token1), _amountIn);
            _burn(address(token0), amountOut);
        }
        return amountOut;
    }

    function addLiquidity(
        uint256 _amount0,
        uint256 _amount1
    ) external returns (uint256 shares) {
        _mint(address(token0), _amount0);
        _mint(address(token1), _amount1);
        return (_amount0 + _amount1);
    }

    function removeLiquidity(
        uint256 _shares
    ) external returns (uint256 d0, uint256 d1) {
        require(_shares > 0 && totalSupply >= _shares);
        d0 = (_shares * reserve0) / totalSupply;
        d1 = _shares - d0;
        _burn(address(token0), d0);
        _burn(address(token1), d1);
        return (d0, d1);
    }

    function _update(uint256 _res0, uint256 _res1) private {
        reserve0 = _res0;
        reserve1 = _res1;
    }
```

```
}
```

实验步骤

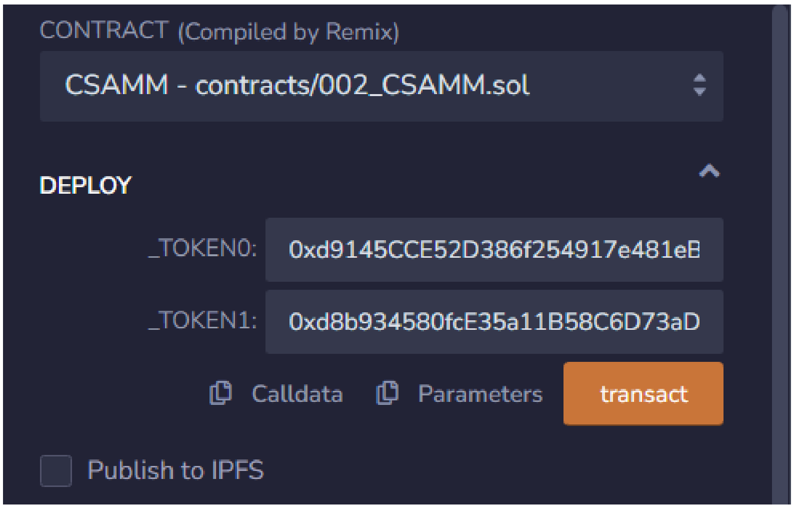
1. 部署 tokenA 和 tokenB



TOKENA AT 0XD91...39138 (MEMORY)

TOKENB AT 0XD8B...33FA8 (MEMORY)

2. 部署 csamm 合约



CONTRACT (Compiled by Remix)

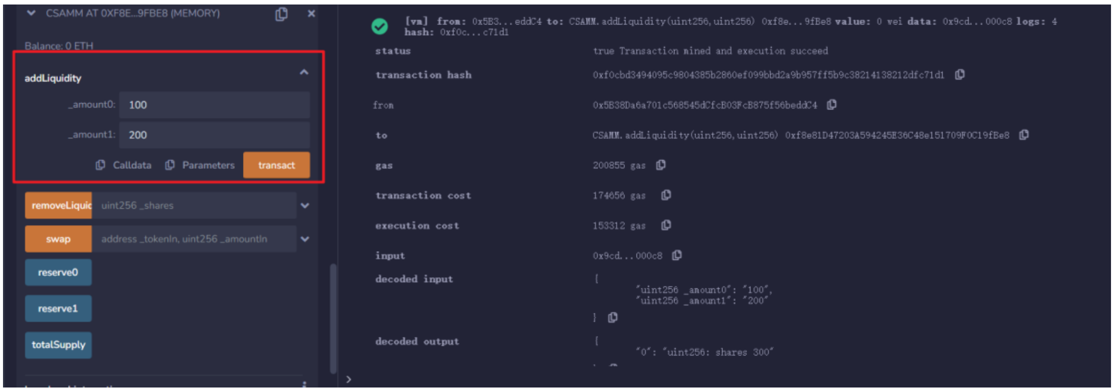
CSAMM - contracts/002_CSAMM.sol

DEPLOY

_TOKEN0: 0xd9145CCE52D386f254917e481eB

_TOKEN1: 0xd8b934580fcE35a11B58C6D73aD

3. 添加流动性



CSAMM AT 0XF8E...9FBE8 (MEMORY)

Balance: 0 ETH

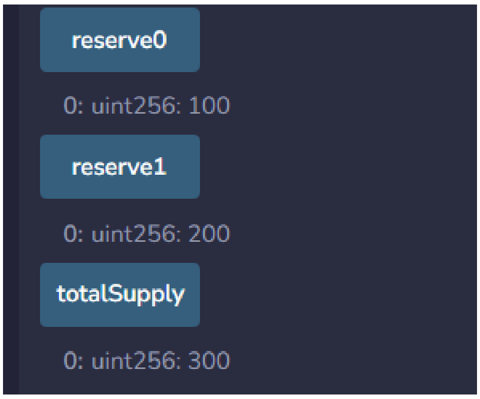
addLiquidity

_amount0: 100

_amount1: 200

```
[vm] from: 0x5B3...eddC4 to: CSAMM.addLiquidity(uint256,uint256) 0xf8e...9fBe8 value: 0 wei data: 0x9cd...000c8 logs: 4
hash: 0xf0c...c71d1
status              true Transaction mined and execution succeed
transaction hash    0xf0cbd3494095c9804385b2860ef099bbd2a9b957ff5b9c38214138212dfc71d1
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  CSAMM.addLiquidity(uint256,uint256) 0xf8e81D47203A594245E36C48e151709F0C19fBe8
gas                 200855 gas
transaction cost    174656 gas
execution cost      153312 gas
input               0x9cd...000c8
decoded input       {
                        "uint256 _amount0": "100",
                        "uint256 _amount1": "200"
                    }
decoded output      {
                        "0": "uint256: shares 300"
```

4. 查看状态


reserve0

0: uint256: 100

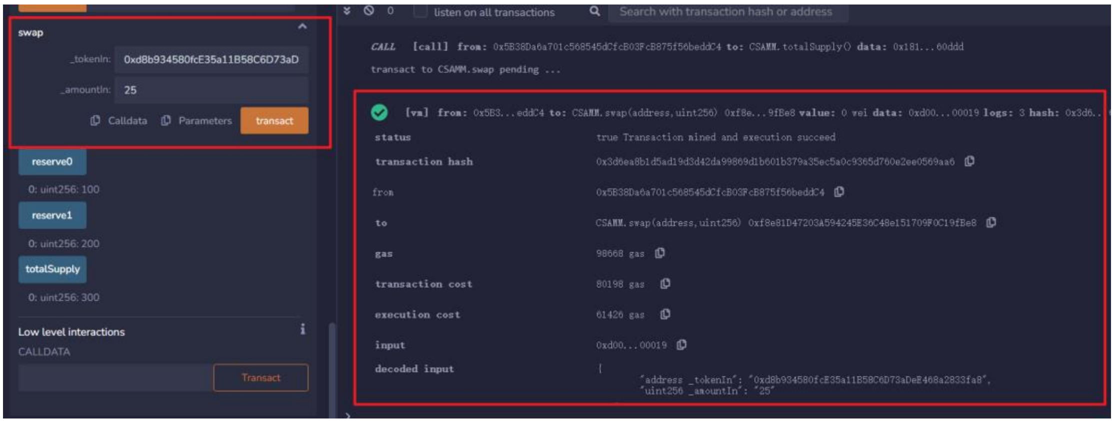
reserve1

0: uint256: 200

totalSupply

0: uint256: 300

5. 购买 25 个 tokenB


swap

_tokenIn: 0xd8b934580fcE35a11B58C6D73aD

_amountIn: 25

Calldata Parameters transact

reserve0

0: uint256: 100

reserve1

0: uint256: 200

totalSupply

0: uint256: 300

Low level interactions

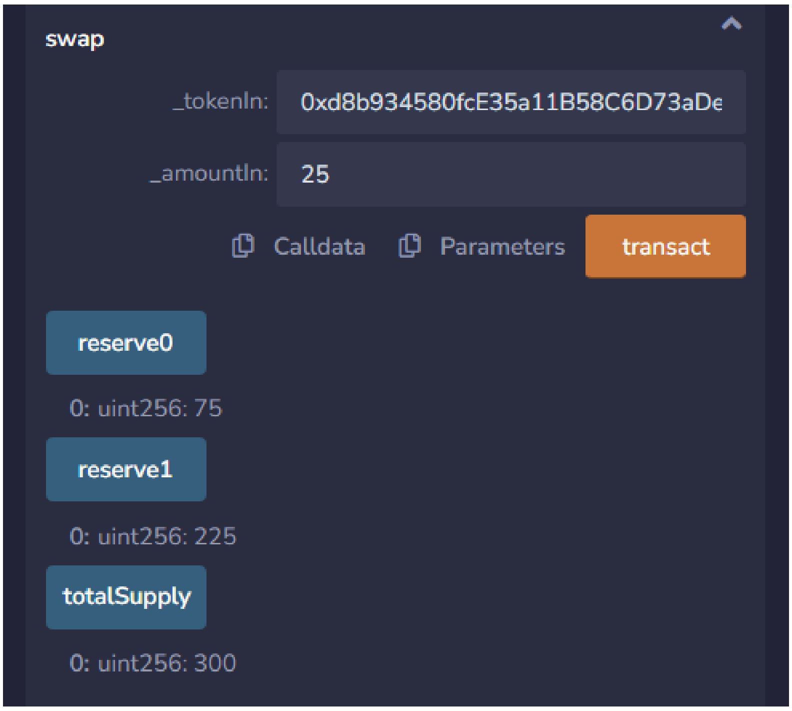
CALLDATA

Transact

0
listen on all transactions
Search with transaction hash or address

```
CALL  [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: CSAMM.totalSupply() data: 0x181...60ddd
transact to CSAMM.swap pending ...

[vm] from: 0x5B3...eddC4 to: CSAMM.swap(address,uint256) 0xf8e...9fBe8 value: 0 wei data: 0xd00...00019 logs: 3 hash: 0x3d6..
status                   true Transaction mined and execution succeed
transaction hash         0x3d6ea8b1d5ad19d3d42da99869d1b601b379a35ec5a0c9365d760e2ee0569aa6
from                     0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                       CSAMM.swap(address,uint256) 0xf8e81D47203A594245E36C48e151709F0C19fBe8
gas                      98668 gas
transaction cost         80198 gas
execution cost           61426 gas
input                    0xd00...00019
decoded input            {
                              "address _tokenIn": "0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8",
                              "uint256 _amountIn": "25"
```

6. 查看状态


swap

_tokenIn: 0xd8b934580fcE35a11B58C6D73aDe

_amountIn: 25

Calldata Parameters transact

reserve0

0: uint256: 75

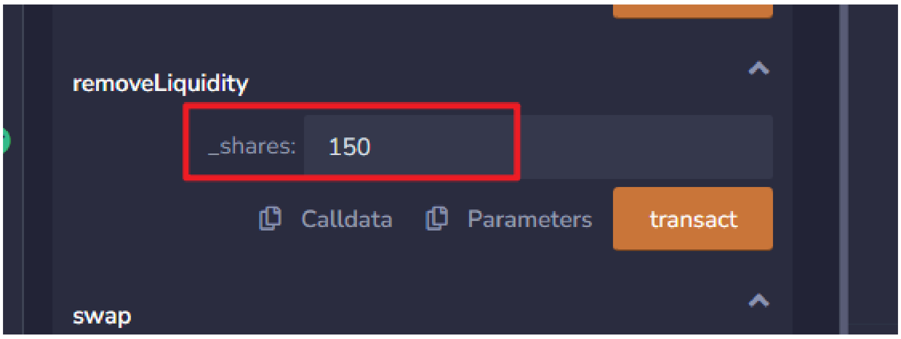
reserve1

0: uint256: 225

totalSupply

0: uint256: 300

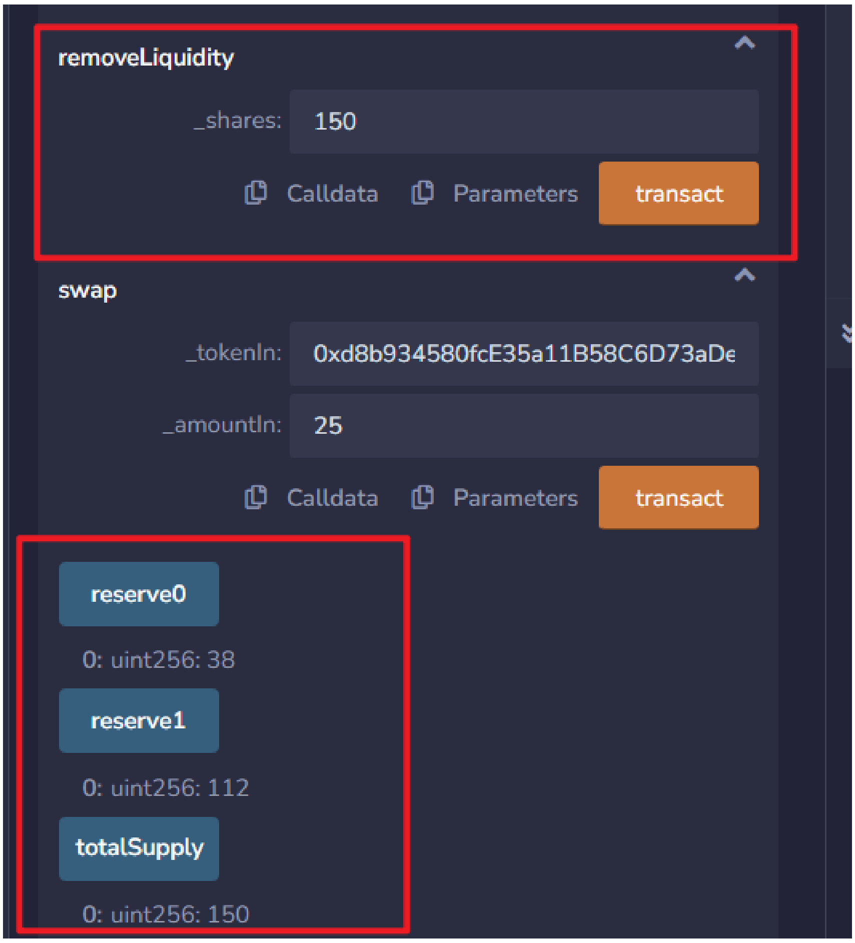
7. 移除流动性 150



removeLiquidity

_shares: 150

8. 查看状态



removeLiquidity

_shares: 150

swap

_tokenIn: 0xd8b934580fcE35a11B58C6D73aDe

_amountIn: 25

0: uint256: 38

0: uint256: 112

0: uint256: 150