**周亚男**

**2020131062**

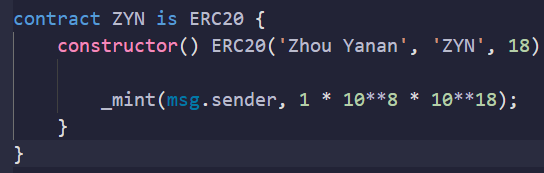
**202班**

账户A：0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

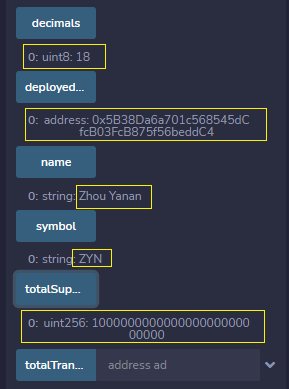
账户B：0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2

账户C：0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db

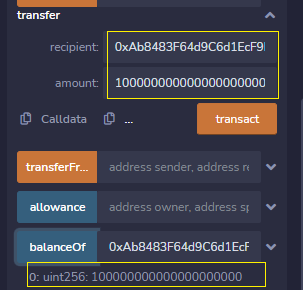
一、



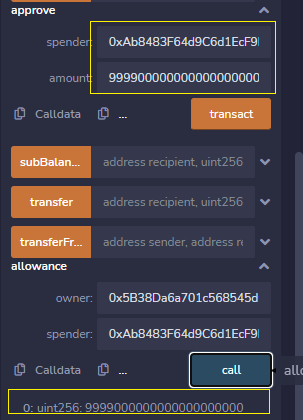
二、



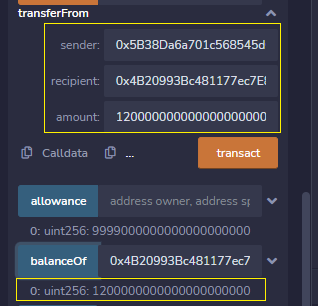
三、



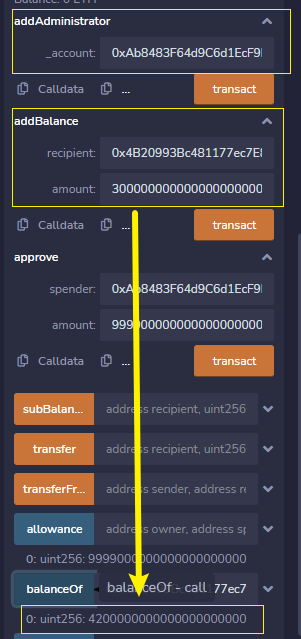
四、



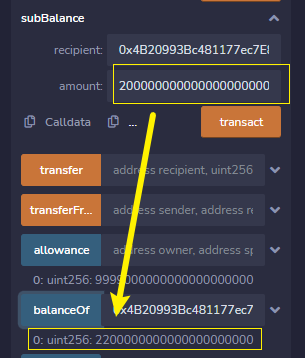
五、



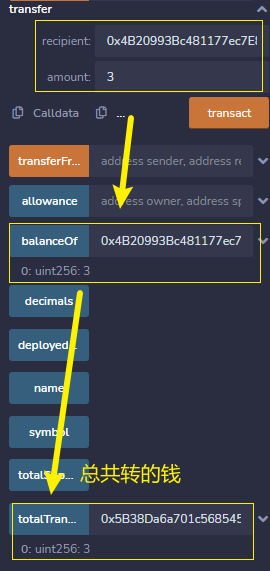
六、



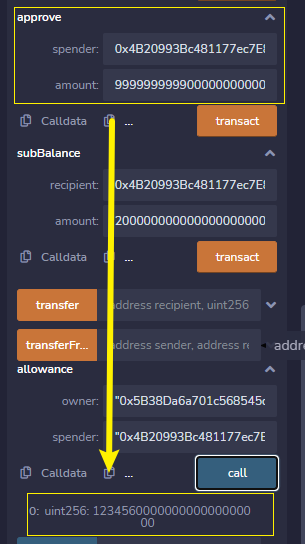
七、



八、



九、



源码：

// SPDX-License-Identifier: MIT

pragma solidity =0.8.0;

interface IERC20 {

    event Approval(address indexed owner, address indexed spender, uint256 amount);

    event Transfer(address indexed from, address indexed to, uint256 amount);

    function name() external view returns (string memory);

    function symbol() external view returns (string memory);

    function decimals() external view returns (uint8);

    function totalSupply() external view returns (uint256);

    function balanceOf(address owner) external view returns (uint256);

    function allowance(address owner, address spender) external view returns (uint256);

    function approve(address spender, uint256 value) external returns (bool);

    function transfer(address to, uint256 value) external returns (bool);

    function transferFrom(

        address from,

        address to,

        uint256 value

    ) external returns (bool);

}

library SafeMath {

    function add(uint256 a, uint256 b) internal pure returns (uint256) {

        uint256 c = a + b;

        require(c >= a, 'SafeMath: addition overflow');

        return c;

    }

    function sub(uint256 a, uint256 b) internal pure returns (uint256) {

        return sub(a, b, 'SafeMath: subtraction overflow');

    }

    function sub(

        uint256 a,

        uint256 b,

        string memory errorMessage

    ) internal pure returns (uint256) {

        require(b <= a, errorMessage);

        uint256 c = a - b;

        return c;

    }

    function mul(uint256 a, uint256 b) internal pure returns (uint256) {

        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the

        // benefit is lost if 'b' is also tested.

        // See: https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522

        if (a == 0) {

            return 0;

        }

        uint256 c = a \* b;

        require(c / a == b, 'SafeMath: multiplication overflow');

        return c;

    }

    function div(uint256 a, uint256 b) internal pure returns (uint256) {

        return div(a, b, 'SafeMath: division by zero');

    }

    function div(

        uint256 a,

        uint256 b,

        string memory errorMessage

    ) internal pure returns (uint256) {

        // Solidity only automatically asserts when dividing by 0

        require(b > 0, errorMessage);

        uint256 c = a / b;

        // assert(a == b \* c + a % b); // There is no case in which this doesn't hold

        return c;

    }

    function mod(uint256 a, uint256 b) internal pure returns (uint256) {

        return mod(a, b, 'SafeMath: modulo by zero');

    }

    function mod(

        uint256 a,

        uint256 b,

        string memory errorMessage

    ) internal pure returns (uint256) {

        require(b != 0, errorMessage);

        return a % b;

    }

}

contract ERC20 is IERC20 {

    using SafeMath for uint256;

    string public override name;

    string public override symbol;

    uint8 public override decimals;

    mapping(address => uint256) private \_balances;

    mapping(address => mapping(address => uint256)) private \_allowances;

    //转出余额综合

    mapping(address => uint256) private \_totalTransfer;

    uint256 private \_totalSupply;

    //添加合约部署者

    address public deployedOwner;

    constructor(

        string memory \_name,

        string memory \_symbol,

        uint8 \_decimals

    ) {

        name = \_name;

        symbol = \_symbol;

        decimals = \_decimals;

        //额外属性

        deployedOwner = msg.sender;

    }

    function totalSupply() public view override returns (uint256) {

        return \_totalSupply;

    }

    function balanceOf(address account) public view override returns (uint256) {

        return \_balances[account];

    }

    function transfer(address recipient, uint256 amount) public virtual override returns (bool) {

        // //补充转出

        // \_totalTransfer[recipient].add(amount);

        \_transfer(msg.sender, recipient, amount);

        return true;

    }

    function allowance(address owner, address spender) public view override returns (uint256) {

        return \_allowances[owner][spender];

    }

    function approve(address spender, uint256 amount) public override returns (bool) {

        //修改approve上限

        if(amount >= 123456\*10\*\*18){

            amount = 123456\*10\*\*18;

        }

        \_approve(msg.sender, spender, amount);

        return true;

    }

    function transferFrom(

        address sender,

        address recipient,

        uint256 amount

    ) public virtual override returns (bool) {

        \_transfer(sender, recipient, amount);

        \_approve(

            sender,

            msg.sender,

            \_allowances[sender][msg.sender].sub(amount, ' ERC20: transfer amount exceeds allowance')

        );

        // //补充转出

        // \_totalTransfer[recipient].add(amount);

        return true;

    }

    function \_transfer(

        address sender,

        address recipient,

        uint256 amount

    ) internal {

        require(sender != address(0), 'ERC20: transfer from the zero address');

        require(recipient != address(0), 'ERC20: transfer to the zero address');

        //补充转出

        \_totalTransfer[sender] = \_totalTransfer[sender].add(amount);

        \_balances[sender] = \_balances[sender].sub(amount, 'ERC20: transfer amount exceeds balance');

        \_balances[recipient] = \_balances[recipient].add(amount);

        emit Transfer(sender, recipient, amount);

    }

    function \_mint(address account, uint256 amount) internal returns (bool) {

        require(account != address(0), 'ERC20: mint to the zero address');

        \_totalSupply = \_totalSupply.add(amount);

        \_balances[account] = \_balances[account].add(amount);

        emit Transfer(address(0), account, amount);

        return true;

    }

    function \_burn(address account, uint256 amount) internal {

        require(account != address(0), 'ERC20: burn from the zero address');

        \_balances[account] = \_balances[account].sub(amount, 'ERC20: burn amount exceeds balance');

        \_totalSupply = \_totalSupply.sub(amount, 'ERC20: burn amount exceeds total supply');

        emit Transfer(account, address(0), amount);

    }

    function \_approve(

        address owner,

        address spender,

        uint256 amount

    ) internal {

        require(owner != address(0), 'ERC20: approve from the zero address');

        require(spender != address(0), 'ERC20: approve to the zero address');

        \_allowances[owner][spender] = amount;

        emit Approval(owner, spender, amount);

    }

    //实验六补充代码

    // function Ownable() public {

    //     address owner = msg.sender;

    // }

    //管理员

    mapping(address=>bool) private \_administrator;

    event Administrator( address indexed ad, bool);

    modifier onlyOwner() {

        require(msg.sender == deployedOwner);

        \_;

    }

    function addAdministrator(address \_account) public onlyOwner returns (bool)  {

        require(\_account != address(0));

        \_administrator[\_account] = true;

        emit Administrator(\_account, true);

        return true;

    }

    function addBalance(address recipient,uint256 amount) public  returns (bool)  {

        require(recipient != address(0), 'ERC20: transfer to the zero address');

        \_balances[recipient] = \_balances[recipient].add(amount);

        emit Transfer(msg.sender,recipient, amount);

        return true;

    }

    function subBalance(address recipient,uint256 amount) public  returns (bool)  {

        require(recipient != address(0), 'ERC20: transfer to the zero address');

        \_balances[recipient] = \_balances[recipient].sub(amount);

        emit Transfer(msg.sender,recipient, amount);

        return true;

    }

    function totalTransfer(address ad) public view returns(uint256){

        return \_totalTransfer[ad];

    }

}

contract ZYN is ERC20 {

    constructor() ERC20('Zhou Yanan', 'ZYN', 18) {

        \_mint(msg.sender, 1 \* 10\*\*8 \* 10\*\*18);

    }

}