2CSDE93- Blockchain Technology

Lab 9: To write a Solidity contract that implements a distributed ticket sales system. Anybody can create an event (specifying the initial price and number of tickets). Anybody can then purchase one of the initial tickets or sell those tickets peer-topeer. At the event, gate agents will check that each attendee is listed in the final attendees list on the blockchain. (Ethereum programming)

Outline

- Prerequisites
- Ticket sales system
- Steps for the Implementation
- Conclusion

PREREQUISITES

PREREQUISITES

- Truffle is a framework for developing these Ethereum applications (dApps, or distributed apps). Installed via npm, it'll create a local test blockchain and manage your contract deployment to the blockchain.

- Web3.js is a JavaScript library that provides an interface for reading from and sending data to the contract once its deployed.
- You include it in your Truffle application and it'll get included with your HTML and other assets. It's also available in test mode, so you can write unit tests in JavaScript that modify and assert the state of your contract.

How Smart Contract Can Boost the Ticketing Industry

- The biggest issue in the ticketing industry is the secondary resale market with its fraud schemes.
- When an event is really interesting for consumers, tickets are often sourced on the secondary market.
- In this case, there are no benefits for event participants in terms of value creation, so no one gets a share of the final price.
- A smart contract on Solidity is the solution for fair management of events and ticketing because it tracks each transaction on the blockchain and identifies resellers.
- You will definitely like the way tickets can be sold and managed.

How Smart Contract Can Boost the Ticketing Industry

- We will see how to write a smart contract which governs the relations between different parties in the industry and splits the final price of the smart contract between all participants.

STEPS FOR THE IMPLEMENTATION

Steps to deploy Ethereum Smart Contracts

1. To make your smart contract live, switch to the main ethereum network at metamask.
2. Add some real ethers.
3. Now again, deploy your smart contract using remix.
4. When a smart contract is deployed successfully, visit http://www.etherscan.io and search your smart contract address there. Select your smart contract.
5. Now you need to verify your smart contract here, click “verify contract.”

Steps to deploy Ethereum Smart Contracts

6. Copy your smart contract code and paste it at Etherscan. Select the same compiler version that you selected at remix to compile your code.

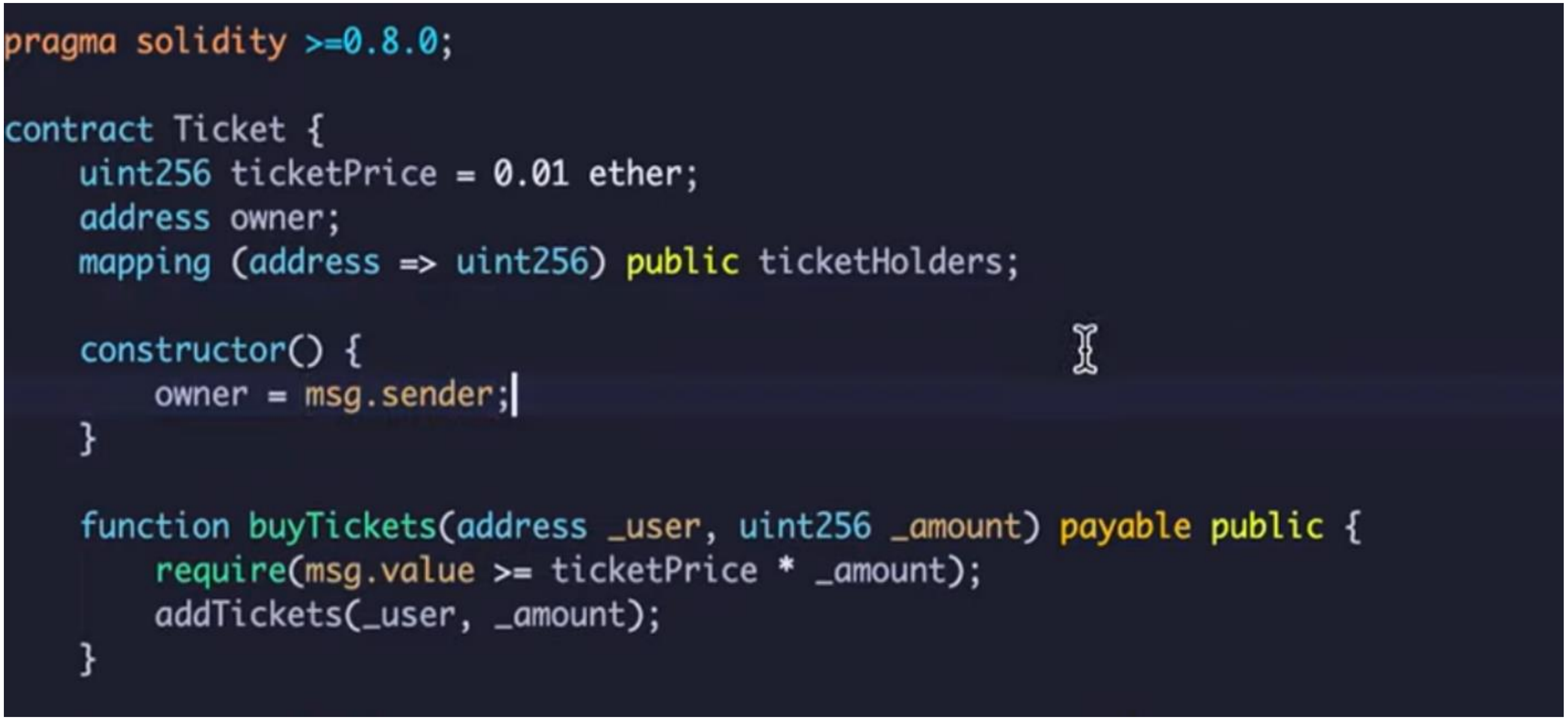
7.Check “optimization” to Yes, if you had selected optimization at remix; otherwise, select No.

8. Click Verify.

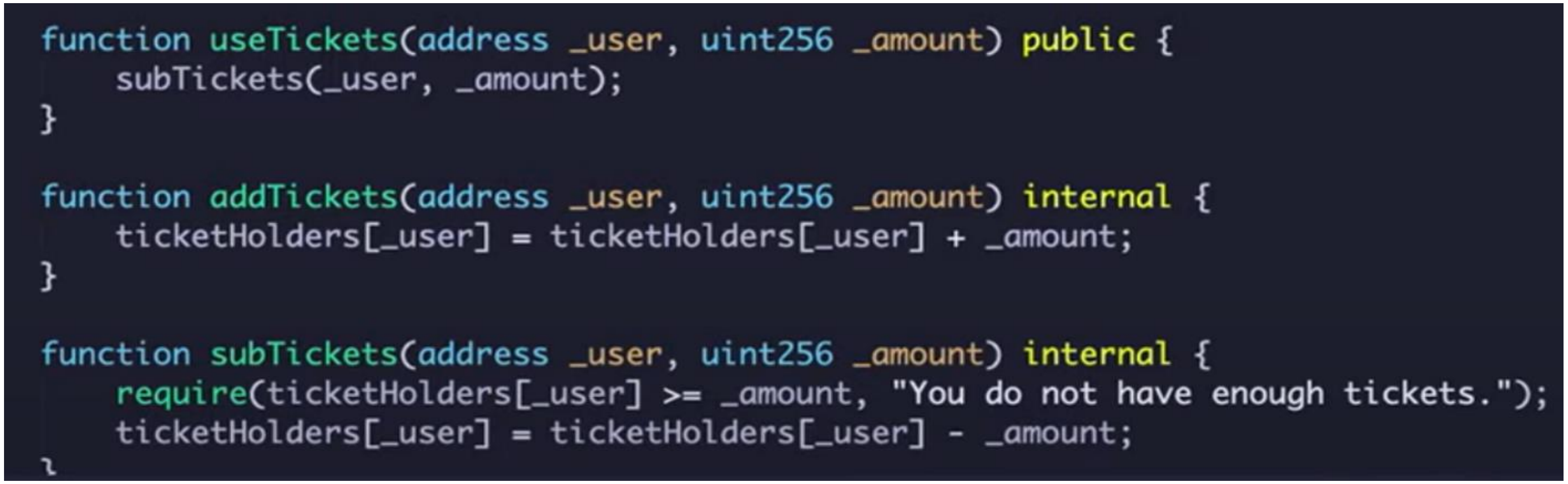
9. It will take a few minutes and your smart contract will be live if no issue occurs.

10.You can now run your smart contract methods at Etherscan.

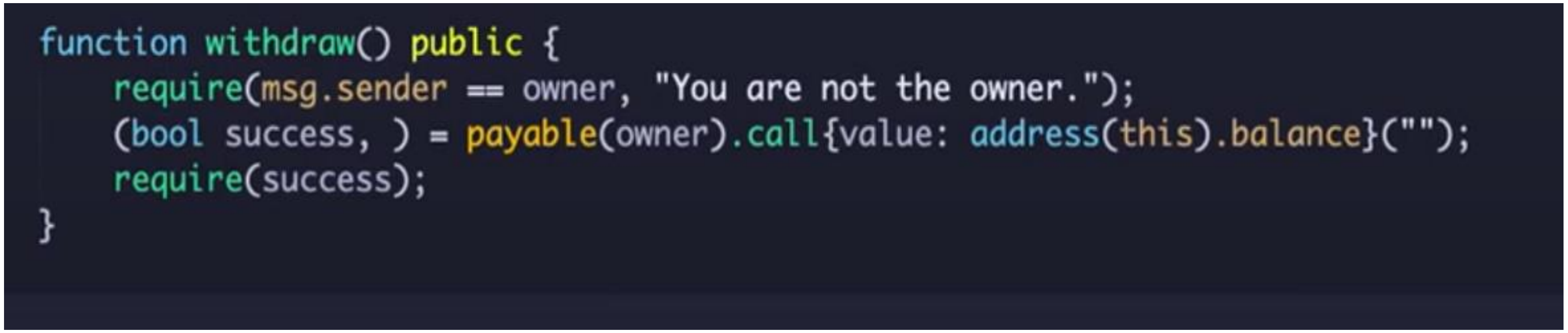
A Sample ticket system smart contract in Solidity



```
pragma solidity >=0.8.0;

contract Ticket {
    uint256 ticketPrice = 0.01 ether;
    address owner;
    mapping (address => uint256) public ticketHolders;

    constructor() {
        owner = msg.sender;
    }

    function buyTickets(address _user, uint256 _amount) payable public {
        require(msg.value >= ticketPrice * _amount);
        addTickets(_user, _amount);
    }
```



```
function useTickets(address _user, uint256 _amount) public {
    subTickets(_user, _amount);
}

function addTickets(address _user, uint256 _amount) internal {
    ticketHolders[_user] = ticketHolders[_user] + _amount;
}

function subTickets(address _user, uint256 _amount) internal {
    require(ticketHolders[_user] >= _amount, "You do not have enough tickets.");
    ticketHolders[_user] = ticketHolders[_user] - _amount;
}
```



```
function withdraw() public {
    require(msg.sender == owner, "You are not the owner.");
    (bool success, ) = payable(owner).call{value: address(this).balance}("");
    require(success);
}
```

Take Home Assignment

- The following Solidity contract in (ticket.sol)implements the distributed ticket sales system. Anyone can create a transaction, specifying the initial price and the number of tickets.

- Then, anyone can buy one of the initial tickets or sell tickets point-to-point.

- In this event, the agent will check whether each participant is listed in the final participant list of the blockchain.

- Find at least 5 bugs that cause loss of specific functions or security vulnerabilities; explain their meanings and write down the correct code after removing the bugs.

Thank You