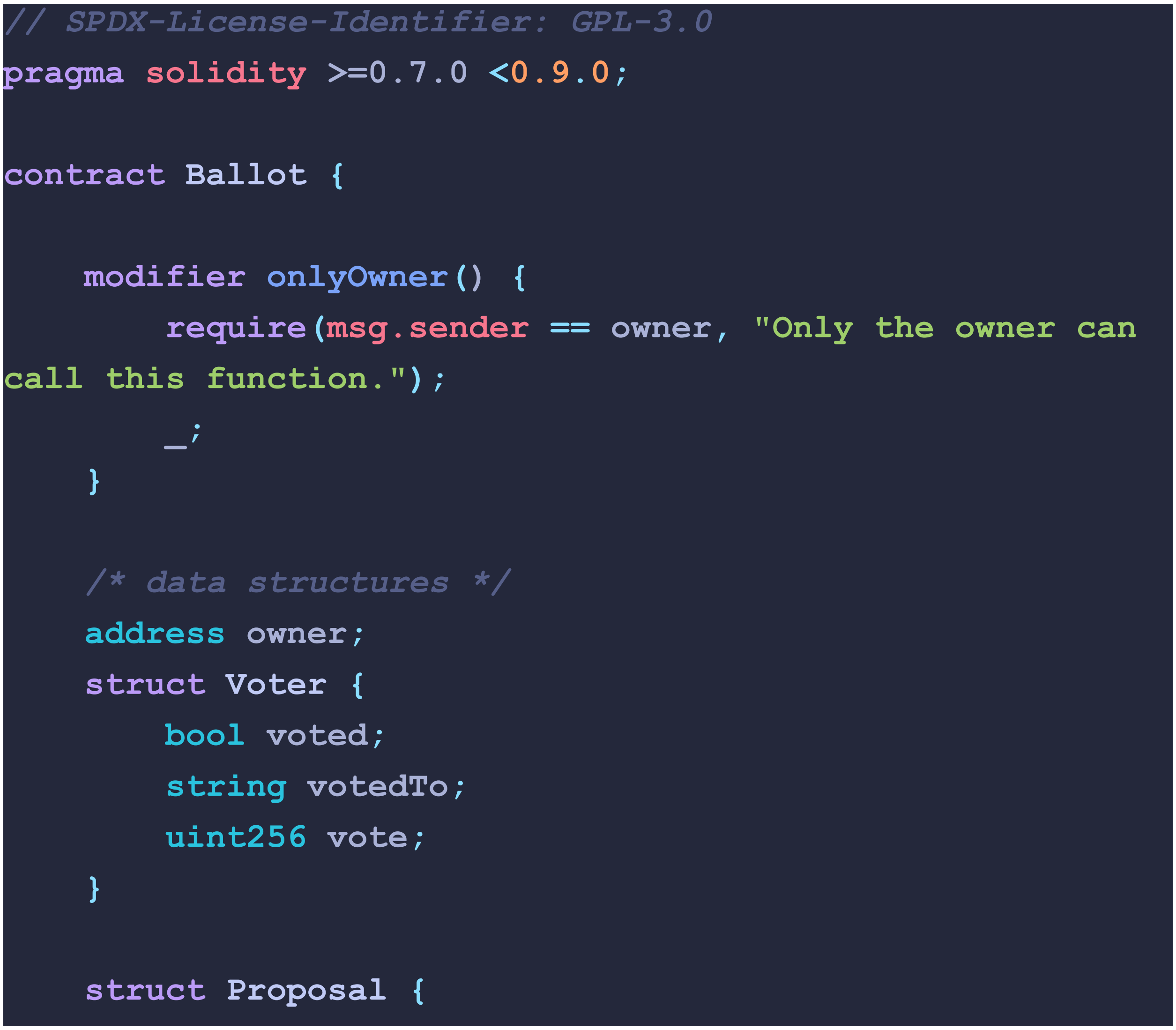
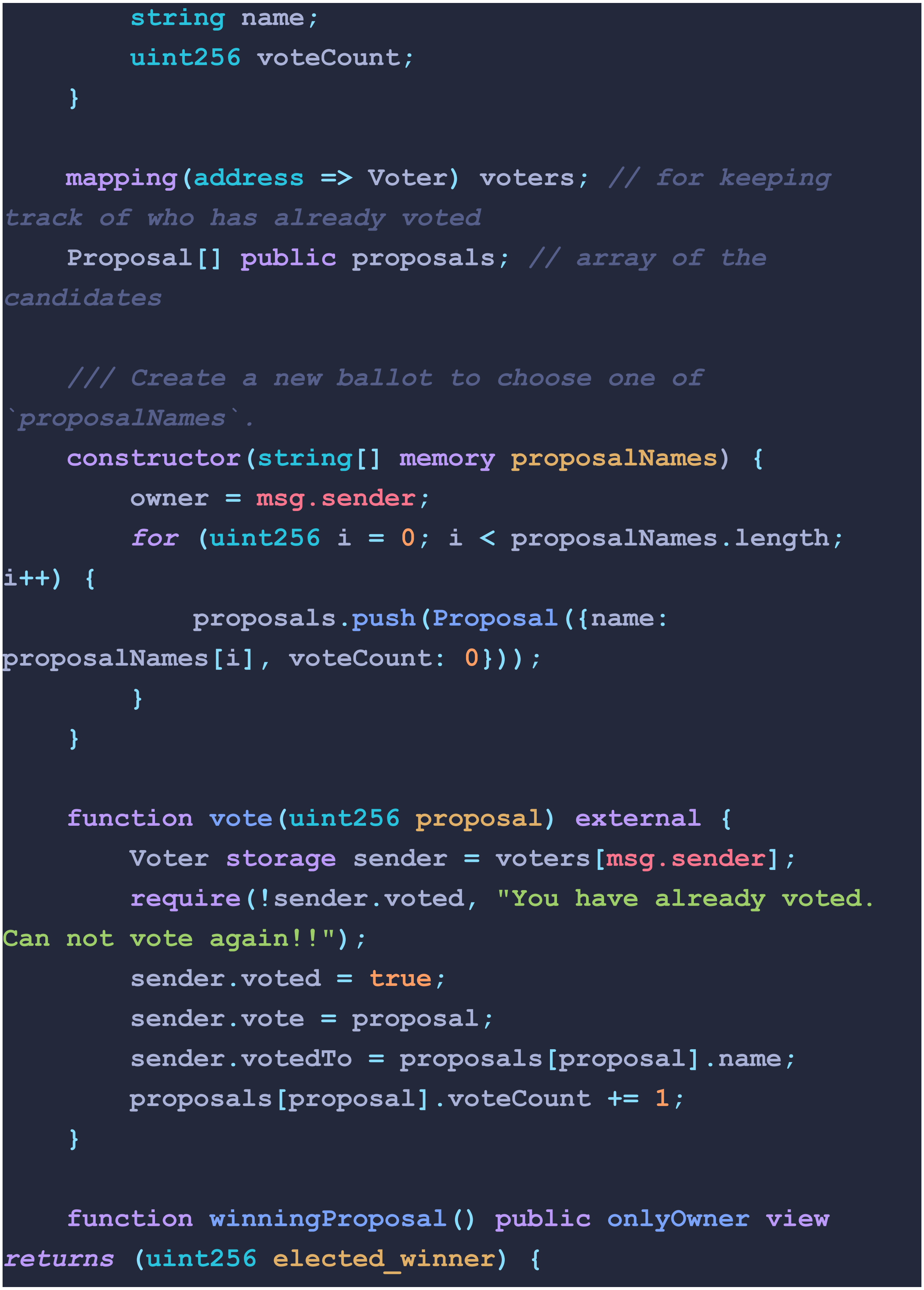
Name: Raj K Patel
Roll No: 20BCE218
Course: 2CSDE93 - Blockchain Technology
Practical No: 6
Aim: To build, implement and test voting mechanisms using Ethereum Blockchain. First, list the contestants on the screen and the vote they got. Whenever the user tries to vote for a particular contestant, the count of the votes for the particular contestant should increase by 1. Also, the user who has already voted should be marked. Marked means “the user has already voted once and will not be allowed to vote again”.

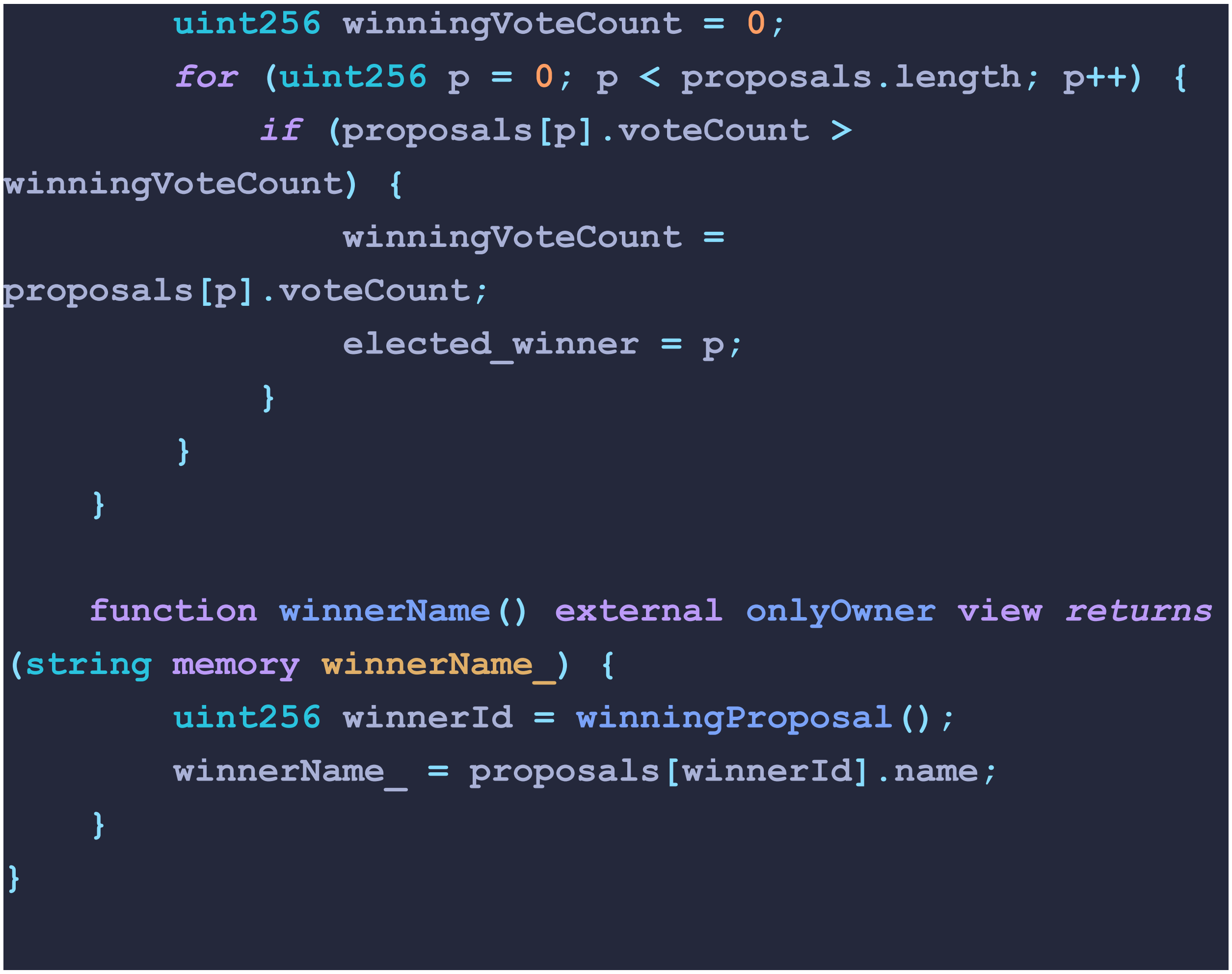
Code:



```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;

contract Ballot {

    modifier onlyOwner() {
        require(msg.sender == owner, "Only the owner can call this function.");
        _;
    }

    /* data structures */
    address owner;
    struct Voter {
        bool voted;
        string votedTo;
        uint256 vote;
    }

    struct Proposal {
```



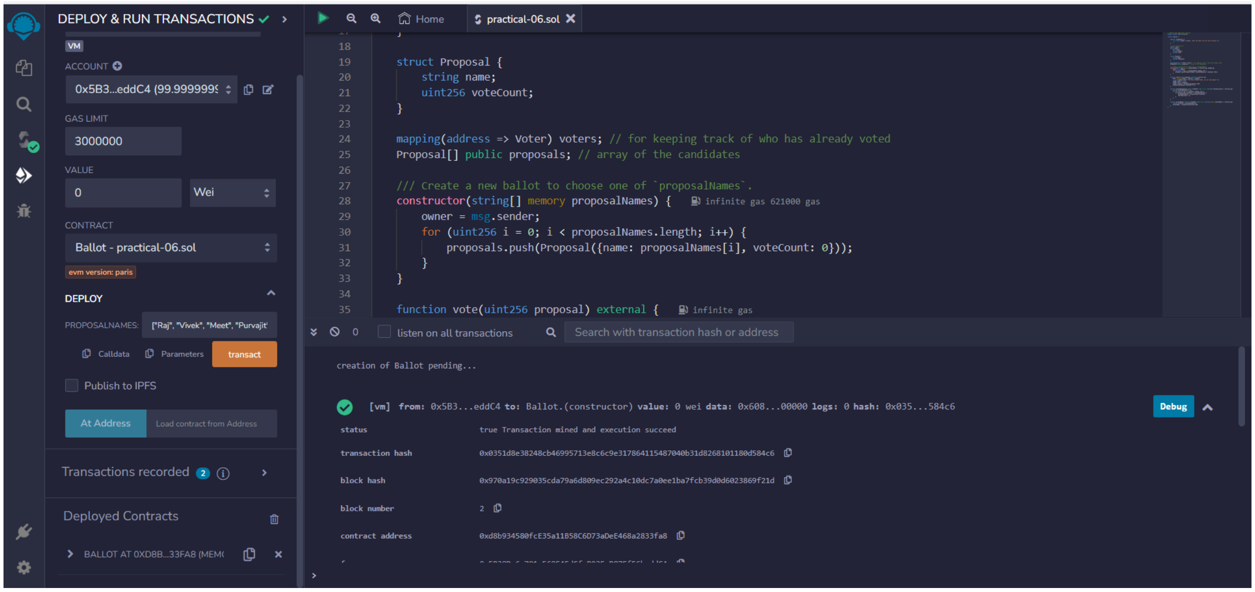
```
        string name;
        uint256 voteCount;
    }

    mapping(address => Voter) voters; // for keeping
track of who has already voted
    Proposal[] public proposals; // array of the
candidates

    /// Create a new ballot to choose one of
`proposalNames`.
    constructor(string[] memory proposalNames) {
        owner = msg.sender;
        for (uint256 i = 0; i < proposalNames.length;
i++) {
            proposals.push(Proposal({name:
proposalNames[i], voteCount: 0}));
        }
    }

    function vote(uint256 proposal) external {
        Voter storage sender = voters[msg.sender];
        require(!sender.voted, "You have already voted.
Can not vote again!!");
        sender.voted = true;
        sender.vote = proposal;
        sender.votedTo = proposals[proposal].name;
        proposals[proposal].voteCount += 1;
    }

    function winningProposal() public onlyOwner view
returns (uint256 elected_winner) {
```



```
        uint256 winningVoteCount = 0;
        for (uint256 p = 0; p < proposals.length; p++) {
            if (proposals[p].voteCount >
winningVoteCount) {
                winningVoteCount =
proposals[p].voteCount;
                elected_winner = p;
            }
        }
    }

    function winnerName() external onlyOwner view returns
(string memory winnerName_) {
        uint256 winnerId = winningProposal();
        winnerName_ = proposals[winnerId].name;
    }
}
```

Output:

1. Contact deployment



ACCOUNT: 0x5B3...eddC4 (99.999999

GAS LIMIT: 3000000

VALUE: 0 Wei

CONTRACT: Ballot - practical-06.sol

evm version: paris

DEPLOY

PROPOSALNAMES: ["Raj", "Vivek", "Meet", "Purvajit'

☐ Publish to IPFS

Transactions recorded 2

Deployed Contracts

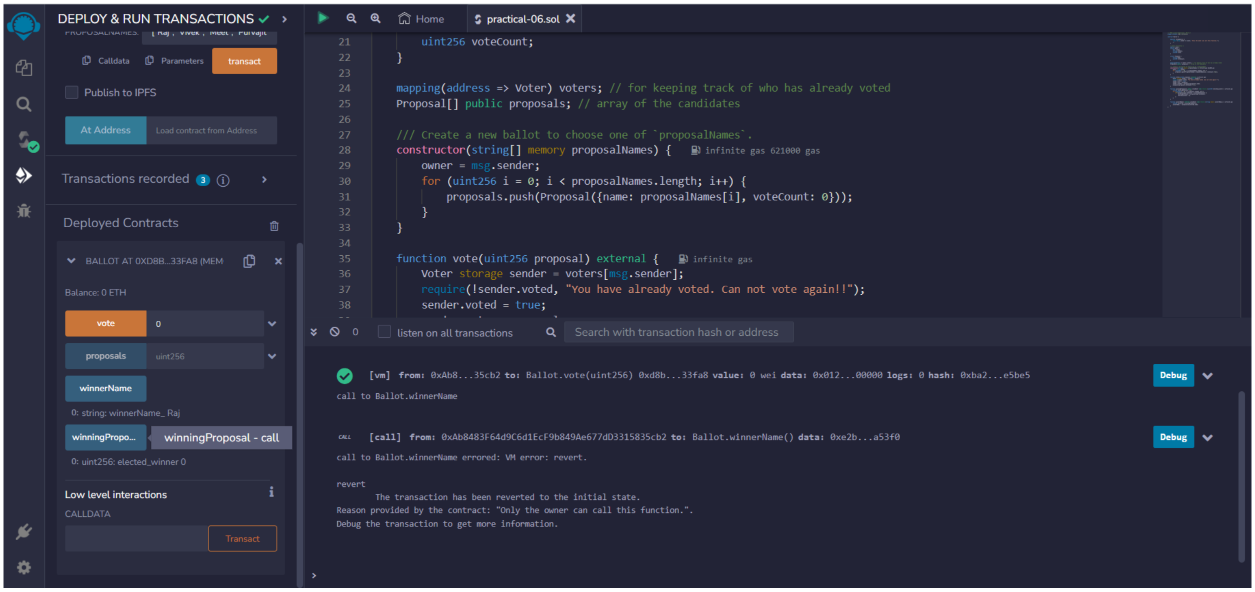
BALLOT AT 0XD8B...33FA8 (MEM(

```
    struct Proposal {
        string name;
        uint256 voteCount;
    }

    mapping(address => Voter) voters; // for keeping track of who has already voted
    Proposal[] public proposals; // array of the candidates

    /// Create a new ballot to choose one of `proposalNames`.
    constructor(string[] memory proposalNames) {    infinite gas 621000 gas
        owner = msg.sender;
        for (uint256 i = 0; i < proposalNames.length; i++) {
            proposals.push(Proposal({name: proposalNames[i], voteCount: 0}));
        }
    }

    function vote(uint256 proposal) external {    infinite gas
```

0 ☐ listen on all transactions — Search with transaction hash or address

```
creation of Ballot pending...

[vm] from: 0x5B3...eddC4 to: Ballot.(constructor) value: 0 wei data: 0x608...00000 logs: 0 hash: 0x035...584c6
status               true Transaction mined and execution succeed
transaction hash     0x0351d8e38248cb46995713e8c6c9e317864115487040b31d8268101180d584c6
block hash           0x970a19c929035cda79a6d809ec292a4c10dc7a0ee1ba7fcb39d0d6023869f21d
block number         2
contract address     0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8
```

2. Voting for a candidate



PROPOSALNAMES: ["Raj", "Vivek", "Meet", "Purvajit

☐ Publish to IPFS

Transactions recorded 3

Deployed Contracts

BALLOT AT 0XD8B...33FA8 (MEM

Balance: 0 ETH

vote: 0

proposals: uint256

0: string: winnerName_ Raj

0: uint256: elected_winner 0

Low level interactions

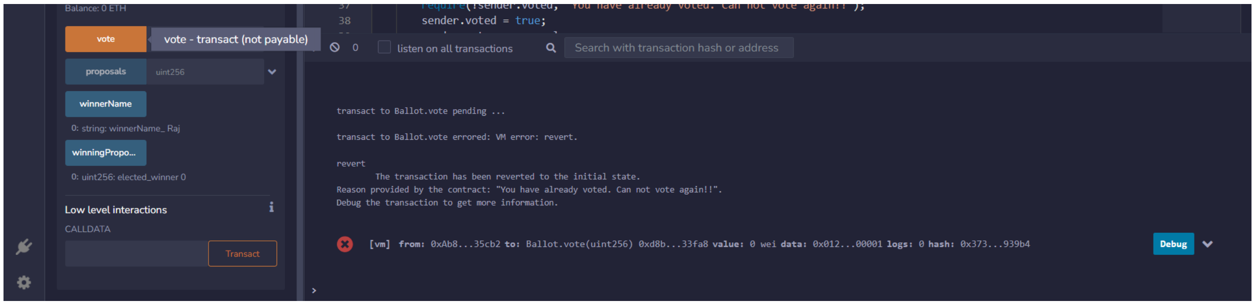
CALLDATA

```
        uint256 voteCount;
    }

    mapping(address => Voter) voters; // for keeping track of who has already voted
    Proposal[] public proposals; // array of the candidates

    /// Create a new ballot to choose one of `proposalNames`.
    constructor(string[] memory proposalNames) {    infinite gas 621000 gas
        owner = msg.sender;
        for (uint256 i = 0; i < proposalNames.length; i++) {
            proposals.push(Proposal({name: proposalNames[i], voteCount: 0}));
        }
    }

    function vote(uint256 proposal) external {    infinite gas
        Voter storage sender = voters[msg.sender];
        require(!sender.voted, "You have already voted. Can not vote again!!");
        sender.voted = true;
```

0 ☐ listen on all transactions — Search with transaction hash or address

```
[vm] from: 0xAb8...35cb2 to: Ballot.vote(uint256) 0xd8b...33fa8 value: 0 wei data: 0x012...00000 logs: 0 hash: 0xba2...e5be5
call to Ballot.winnerName

CALL [call] from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 to: Ballot.winnerName() data: 0xe2b...a53f0
call to Ballot.winnerName errored: VM error: revert.

revert
        The transaction has been reverted to the initial state.
Reason provided by the contract: "Only the owner can call this function.".
Debug the transaction to get more information.
```

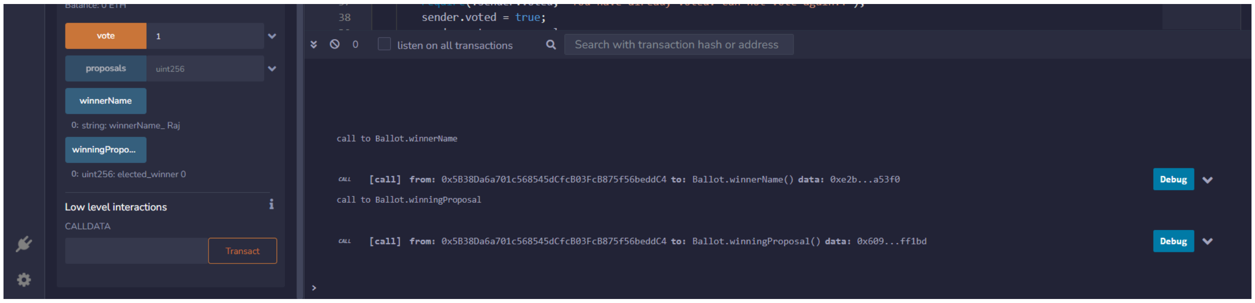
3.Can not vote again


Balance: 0 ETH
vote
vote - transact (not payable)
proposals
uint256
winnerName
0: string: winnerName_ Raj
winningPropo...
0: uint256: elected_winner 0
Low level interactions
CALLDATA
Transact
0
listen on all transactions
Search with transaction hash or address
Debug

```
        require(!sender.voted, "You have already voted. Can not vote again!!");
        sender.voted = true;
```

```
transact to Ballot.vote pending ...

transact to Ballot.vote errored: VM error: revert.

revert
	The transaction has been reverted to the initial state.
Reason provided by the contract: "You have already voted. Can not vote again!!".
Debug the transaction to get more information.

[vm] from: 0xAb8...35cb2 to: Ballot.vote(uint256) 0xd8b...33fa8 value: 0 wei data: 0x012...00001 logs: 0 hash: 0x373...939b4
```

4. Only owner can announce the results


Balance: 0 ETH
vote
1
proposals
uint256
winnerName
0: string: winnerName_ Raj
winningPropo...
0: uint256: elected_winner 0
Low level interactions
CALLDATA
Transact
0
listen on all transactions
Search with transaction hash or address
Debug
Debug

```
        sender.voted = true;
```

```
call to Ballot.winnerName

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Ballot.winnerName() data: 0xe2b...a53f0
call to Ballot.winningProposal

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Ballot.winningProposal() data: 0x609...ff1bd
```