

FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING
Department of Computer Engineering

1. Course , Subject & Experiment Details

Academic Year	2022-23	Estimated Time	02 - Hours
Course & Semester	B.E. (CMPN)- Sem VII	Subject Name & Code	BCT - (CSDC7022)
Chapter No.	03	Chapter Title	Programming for Blockchain

Practical No:	5
Title:	Voting Application using Solidity
Date of Performance:	05/09/2022
Date of Submission:	12/09/2022
Roll No:	8953
Name of the Student:	Brendan Lucas

Evaluation:

Sr. No	Rubric	Grade
1	On time submission Or completion (2)	
2	Preparedness(2)	
3	Skill (4)	
4	Output (2)	

Signature of the Teacher:

Date:

Code:

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

/**
 * @title Ballot
 * @dev Implements voting process along with vote delegation
 */
contract Ballot {

    struct Voter {
        uint weight; // weight is accumulated by delegation
        bool voted;  // if true, that person already voted
        address delegate; // person delegated to
        uint vote;    // index of the voted proposal
    }

    struct Proposal {
        // If you can limit the length to a certain number of bytes,
        // always use one of bytes1 to bytes32 because they are much cheaper
        bytes32 name;    // short name (up to 32 bytes)
        uint voteCount; // number of accumulated votes
    }

    address public chairperson;

    mapping(address => Voter) public voters;

    Proposal[] public proposals;

    /**
     * @dev Create a new ballot to choose one of 'proposalNames'.
     * @param proposalNames names of proposals
     */
    constructor(bytes32[] memory proposalNames) {
        chairperson = msg.sender;
        voters[chairperson].weight = 1;

        for (uint i = 0; i < proposalNames.length; i++) {
            // 'Proposal({...})' creates a temporary

```

```

        // Proposal object and 'proposals.push(...)'
        // appends it to the end of 'proposals'.
        proposals.push(Proposal({
            name: proposalNames[i],
            voteCount: 0
        }));
    }
}

/**
 * @dev Give 'voter' the right to vote on this ballot. May only be called
by 'chairperson'.
 * @param voter address of voter
 */
function giveRightToVote(address voter) public {
    require(
        msg.sender == chairperson,
        "Only chairperson can give right to vote."
    );
    require(
        !voters[voter].voted,
        "The voter already voted."
    );
    require(voters[voter].weight == 0);
    voters[voter].weight = 1;
}

/**
 * @dev Delegate your vote to the voter 'to'.
 * @param to address to which vote is delegated
 */
function delegate(address to) public {
    Voter storage sender = voters[msg.sender];
    require(!sender.voted, "You already voted.");
    require(to != msg.sender, "Self-delegation is disallowed.");

    while (voters[to].delegate != address(0)) {
        to = voters[to].delegate;

        // We found a loop in the delegation, not allowed.
        require(to != msg.sender, "Found loop in delegation.");
    }
}

```

```

    }
    sender.voted = true;
    sender.delegate = to;
    Voter storage delegate_ = voters[to];
    if (delegate_.voted) {
        // If the delegate already voted,
        // directly add to the number of votes
        proposals[delegate_.vote].voteCount += sender.weight;
    } else {
        // If the delegate did not vote yet,
        // add to her weight.
        delegate_.weight += sender.weight;
    }
}

/**
 * @dev Give your vote (including votes delegated to you) to proposal
'proposals[proposal].name'.
 * @param proposal index of proposal in the proposals array
 */
function vote(uint proposal) public {
    Voter storage sender = voters[msg.sender];
    require(sender.weight != 0, "Has no right to vote");
    require(!sender.voted, "Already voted.");
    sender.voted = true;
    sender.vote = proposal;

    // If 'proposal' is out of the range of the array,
    // this will throw automatically and revert all
    // changes.
    proposals[proposal].voteCount += sender.weight;
}

/**
 * @dev Computes the winning proposal taking all previous votes into
account.
 * @return winningProposal_ index of winning proposal in the proposals
array
 */
function winningProposal() public view
    returns (uint winningProposal_)

```

```

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

/**
 * @dev Calls winningProposal() function to get the index of the winner
contained in the proposals array and then
 * @return winnerName_ the name of the winner
 */
function winnerName() public view
    returns (bytes32 winnerName_)
{
    winnerName_ = proposals[winningProposal()].name;
}
}

```

Output:
Contract Creation:

REMIX
 IDE

remix.ethereum.org/#optimize=false&runs=2008&evmVersion=soljson-v0.8.7+commit.e28d00a7.js

Home 2_Owners.sol 3_Ballot.sol

DEPLOY & RUN TRANSACTIONS

Transactions recorded 1

Deployed Contracts

BALLOT at 0x091...39138 (MEMORY)

Balance: 0 ETH

delegate address to

giveRightT... address voter

vote uint256 proposal

chairperson

proposals uint256

voters address

winnerName

winningPr...

Low level interactions
 CALLDATA

```

1 // SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity >=0.7.0 <0.9.0;
4
5 /**
6  * @title Ballot
7  * @dev Implements voting process along with vote delegation
8  */
9 contract Ballot {
10
11     struct Voter {
12         uint weight; // weight is accumulated by delegation
13         bool voted; // if true, that person already voted
14         address delegate; // person delegated to
15         uint vote; // index of the voted proposal
16     }

```

☐ listen on all transactions

[vm] from: 0x5B3...eddC4 to: Ballot.(constructor) value: 0 wei data: 0x608...00000 logs: 0 hash: 0x940...85ba1

status

transaction hash

from

to

gas

transaction cost

execution cost

input

decoded input

true

0x9404b5fc1c3fcd1c5acc6b192b89b9ffe8be88406578768eadce72446de2b85ba1

0x5B38Da6a701c568545dcfcb03c8875f56beddC4

Ballot.(constructor)

1229422 gas

1069062 gas

1069062 gas

0x608...00000

{
 "bytes32[] proposalNames": [
 "0x6d4c4f4f0a00"
]
 }

Debug

Delegation of Authority:

DEPLOY & RUN TRANSACTIONS

At Address Load contract from address

Transactions recorded **2**

Deployed Contracts

- BALLOT at 0xD91...39138 (MEMORY)

Balance: 0 ETH

delegate	0x617f2e2d072fd9d5503197092
giveRightT...	address voter
vote	uint256 proposal
chairperson	
proposals	uint256
voters	address
winnerName	
winningPr...	

Low level interactions
CALLDATA Transact

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

/**
 * @title Ballot
 * @dev Implements voting process along with vote delegation
 */
contract Ballot {

    struct Voter {
        uint weight; // weight is accumulated by delegation
        bool voted;  // if true, that person already voted
        address delegate; // person delegated to
        uint vote;   // index of the voted proposal
    }

    val
    0 wei

transaction to Ballot.delegate pending ...

[vm] from: 0x5B3...eddC4 to: Ballot.delegate(address) 0xd91...39138 value: 0 wei data: 0x5c1...5e7f2 logs: 0 hash: 0x959...5127e Debug ^

status      true Transaction mined and execution succeed
transaction hash  0x9597132fdb49e732ba302d5128f8dd3014905b6695dc99034cbdda678c5127e ⓘ
from          0x5B38Da6a701c568545dcFcf803Fc8875F56beddC4 ⓘ
to            Ballot.delegate(address) 0xd9145CCCE52086f254917e481e84ae9943f39138 ⓘ
gas           82534 gas ⓘ
transaction cost 71768 gas ⓘ
execution cost   71768 gas ⓘ
```

Giving Right to Vote:

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is active, displaying the 'Ballot' contract deployed at address 0x091...39138. The contract's state is shown with fields like 'delegate', 'giveRightToVote', 'vote', 'chairperson', 'proposals', 'voters', 'winnerName', and 'winningPr...'. The 'Low level interactions' section shows a 'CALLDATA' field and a 'Transact' button. The main editor displays the Solidity code for the 'Ballot' contract, which includes a 'Voter' struct and a 'giveRightToVote' function. The bottom panel shows the transaction details for the deployment, including the transaction hash, from, to, gas, and cost.

Voting:

The screenshot shows the Remix IDE interface with the 'Ballot' contract deployed. The 'DEPLOY & RUN TRANSACTIONS' panel on the left shows the 'Ballot' contract at address 0x091...39138. The 'Low level interactions' section shows a 'CALLDATA' field and a 'Transact' button. The main editor displays the Solidity code for the 'Ballot' contract, which includes a 'Voter' struct and a 'vote' function. The bottom panel shows the transaction details for the voting process, including the transaction hash, from, to, gas, and cost.

Winning Proposal:

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel displays the account balance (0 ETH) and transaction details for a call to `winningProposal_0`. The transaction is successful, with a gas cost of 29079. The main editor shows the Solidity code for `2_Owner.sol` and `3_Ballot.sol`. The `3_Ballot.sol` code includes a `winningProposal_0` function that returns the winning proposal name. The transaction log shows the call to `Ballot.winningProposal()` with data `0x609...ff1bd`.

Winning Proposal Name:

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel displays the account balance (0 ETH) and transaction details for a call to `winningProposal_0`. The transaction is successful, with a gas cost of 31477. The main editor shows the Solidity code for `2_Owner.sol` and `3_Ballot.sol`. The `3_Ballot.sol` code includes a `winningProposal_0` function that returns the winning proposal name. The transaction log shows the call to `Ballot.winningProposal()` with data `0xe2b...a53f0`.