BLOCK TECHNOLOGY

Assignment 1:

Step 1: Go to [Chrome Web Store Extensions Section].

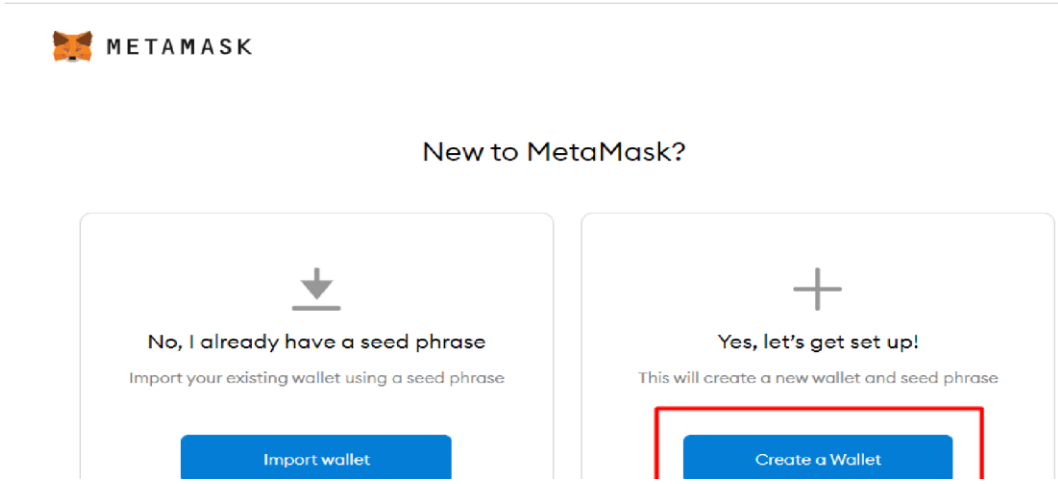
Step 2: Search *MetaMask.*

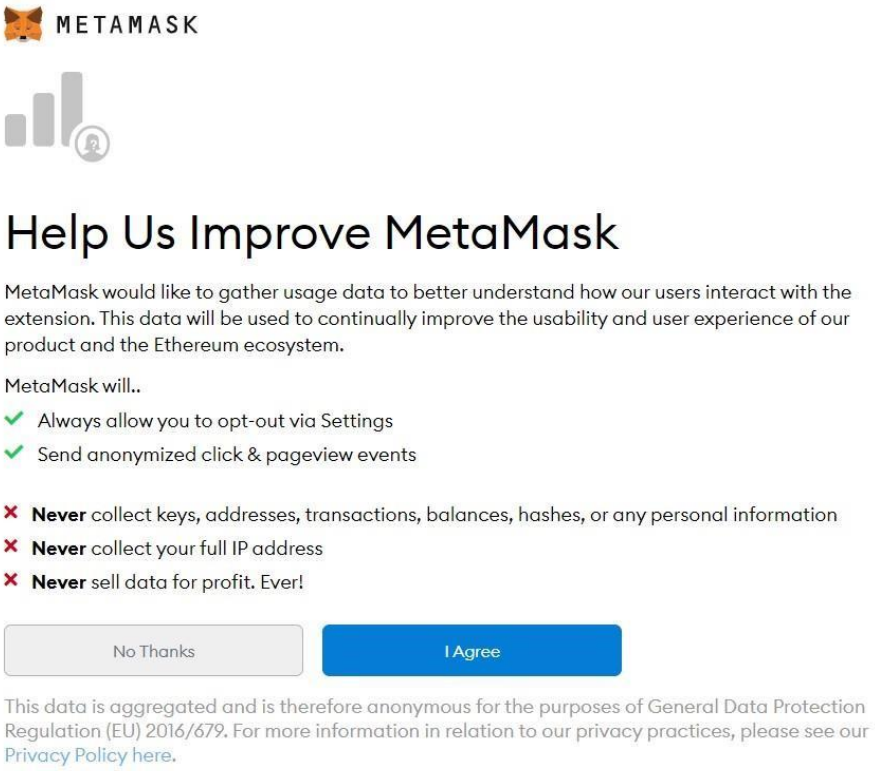
Step 3: Check the number of downloads to make sure that the legitimate MetaMask is being installed, as hackers might try to make clones of it.

Step 4: Click the *Add to Chrome* button.

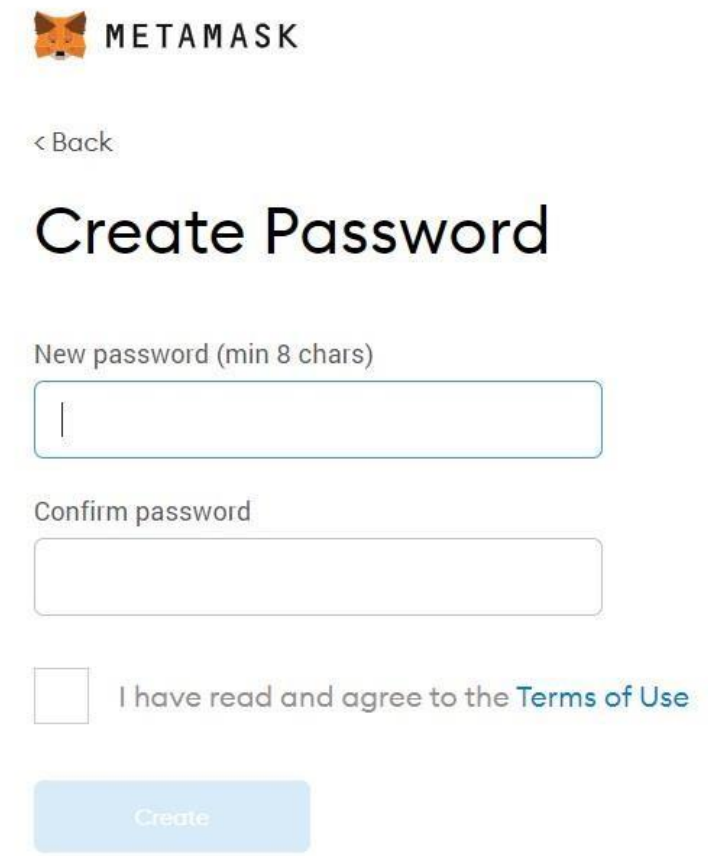
Step 5: Once installation is complete this page will be displayed. Click on the *Get Started* button.

Step 6: This is the first time creating a wallet, so click the *Create a Wallet* button. If there is already a wallet then import the already created using the *Import Wallet* button.

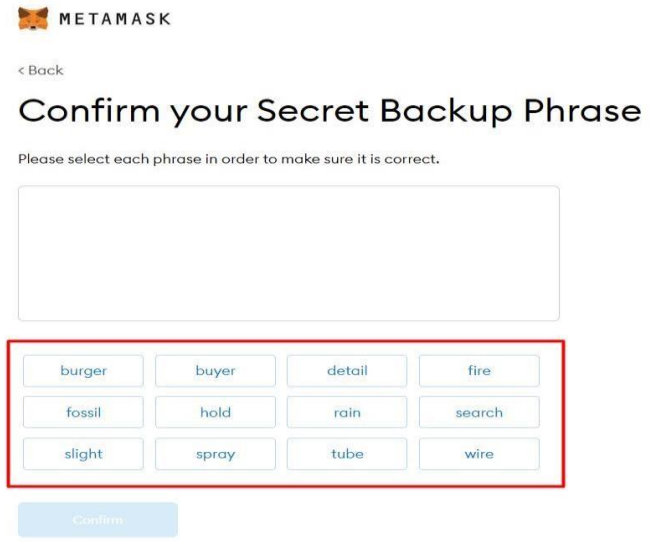




Step 7: Create a password for your wallet. This password is to be entered every time the browser is launched and wants to use MetaMask. A new password needs to be created if chrome is uninstalled or if there is a switching of browsers. In that case, go through the *Import Wallet* button. This is because MetaMask stores the keys in the browser. Agree to *Terms of Use*.



Step 8: Click the buttons respective to the order of the words in your seed phrase. In other words, type the seed phrase using the button on the screen. If done correctly the *Confirm* button should turn blue.



Click the *Confirm* button. Please follow the tips mentioned.

METAMASK

Congratulations

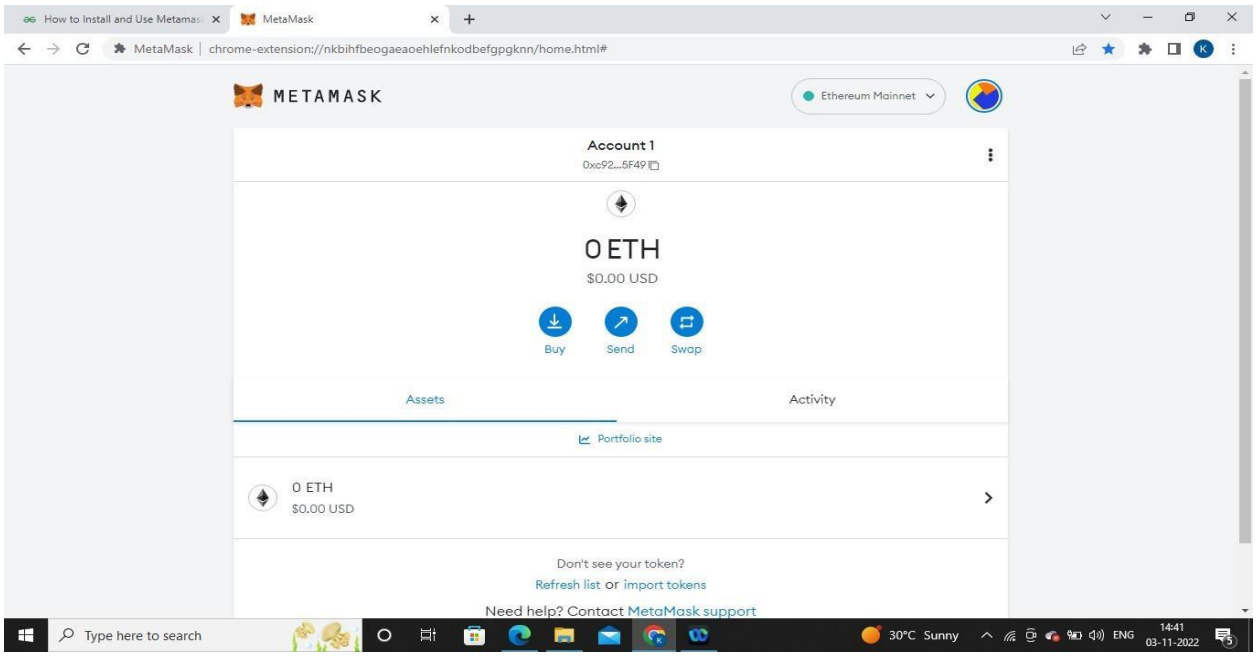
You passed the test - keep your seedphrase safe, it's your responsibility!

Tips on storing it safely

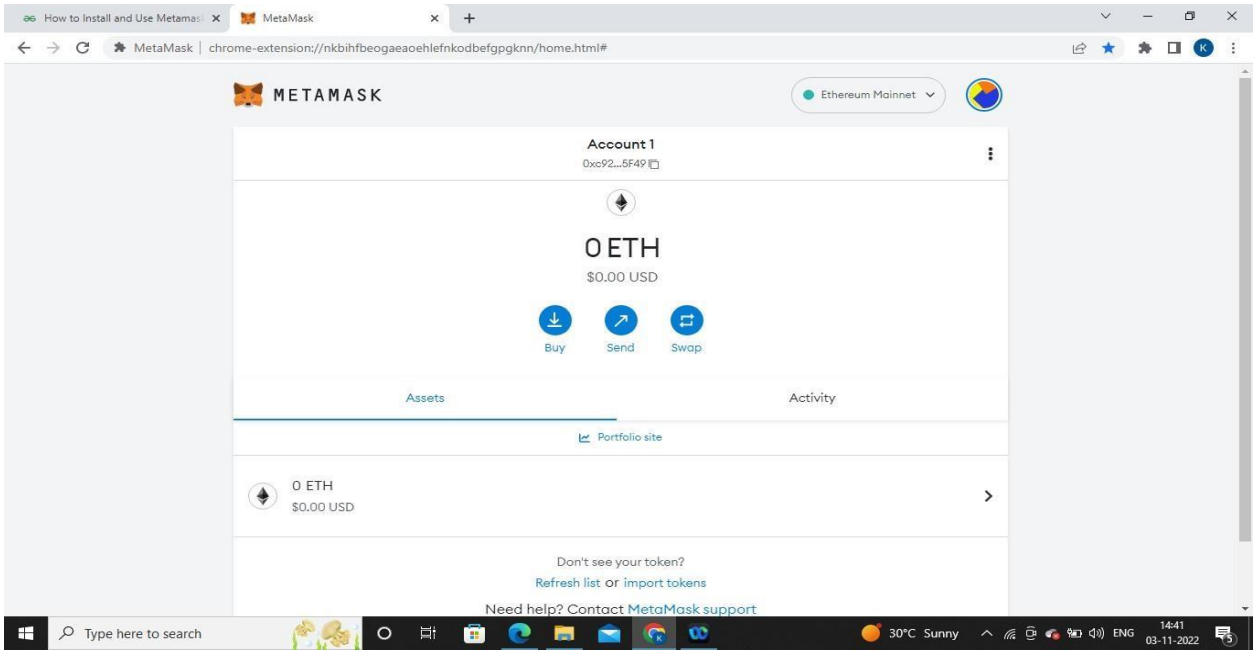
- Save a backup in multiple places.
- Never share the phrase with anyone.
- Be careful of phishing! MetaMask will never spontaneously ask for your seed phrase.
- If you need to back up your seed phrase again, you can find it in Settings -> Security.
- If you ever have questions or see something fishy, contact our support here.

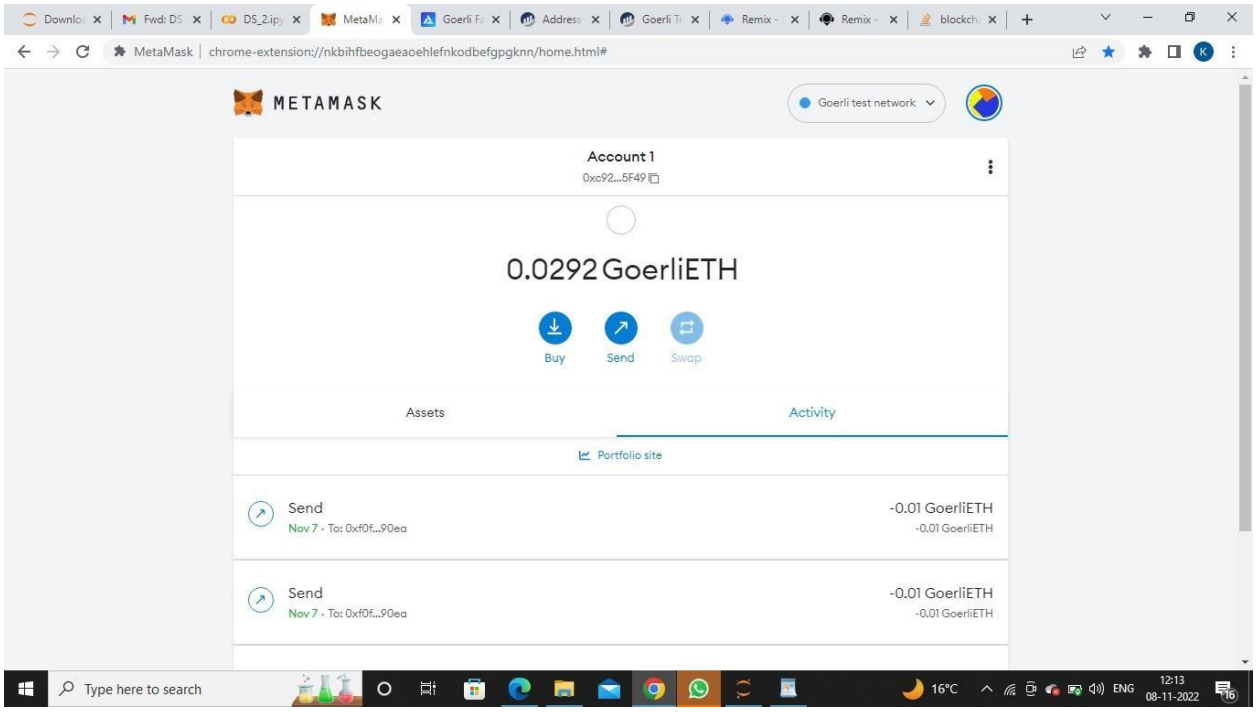
*MetaMask cannot recover your seedphrase. Learn more.

All Done



Assignment 2:





Assignment-3

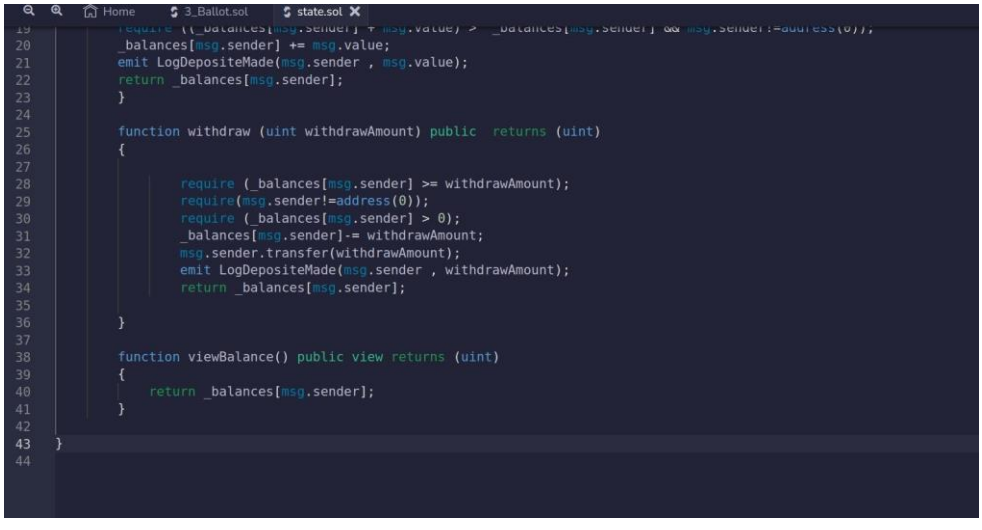
Code:

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0; contract
MyBank
{
   mapping(address=> uint ) private _balances;     address public
owner;     event LogDepositeMade(address accountHoder, uint
amount );
   constructor () public
   {
      owner=msg.sender;         emit
LogDepositeMade(msg.sender, 1000);
   }
      function deposite() public payable  returns (uint)
      {
      require ((_balances[msg.sender] + msg.value) >  _balances[msg.sender] &&
msg.sender!=address(0));
      _balances[msg.sender] += msg.value;        emit
LogDepositeMade(msg.sender , msg.value);
return _balances[msg.sender];
      }
      function withdraw (uint withdrawAmount) public  returns (uint)
      {
            require (_balances[msg.sender] >= withdrawAmount);
require(msg.sender!=address(0));              require
(_balances[msg.sender] > 0);              _balances[msg.sender]-=
withdrawAmount;
msg.sender.transfer(withdrawAmount);              emit
```

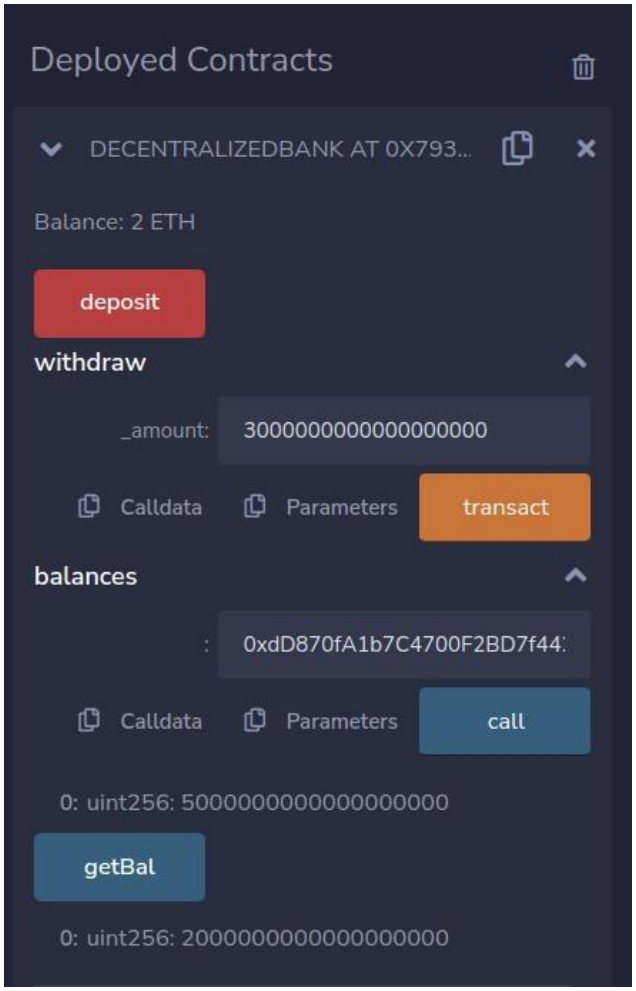
```
LogDepositeMade(msg.sender , withdrawAmount);
return _balances[msg.sender];

    }
    function viewBalance() public view returns (uint)
    {
       return _balances[msg.sender];
    }
}
```

Output:

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

contract MyBank
{
    mapping(address=> uint ) private _balances;
    address public owner;
    event LogDepositeMade(address accountHoder, uint amount );

    constructor () public
    {
        owner=msg.sender;
        emit LogDepositeMade(msg.sender, 1000);
    }

        function deposite() public payable  returns (uint)
        {

        require ((_balances[msg.sender] + msg.value) >  _balances[msg.sender] && msg.sender!=address(0));
        _balances[msg.sender] += msg.value;
        emit LogDepositeMade(msg.sender , msg.value);
        return _balances[msg.sender];
        }

        function withdraw (uint withdrawAmount) public  returns (uint)
        {

                require (_balances[msg.sender] >= withdrawAmount);
                require(msg.sender!=address(0));
                require (_balances[msg.sender] > 0);
```



```
        _balances[msg.sender] += msg.value;
        emit LogDepositeMade(msg.sender , msg.value);
        return _balances[msg.sender];
        }

        function withdraw (uint withdrawAmount) public  returns (uint)
        {

                require (_balances[msg.sender] >= withdrawAmount);
                require(msg.sender!=address(0));
                require (_balances[msg.sender] > 0);
                _balances[msg.sender]-= withdrawAmount;
                msg.sender.transfer(withdrawAmount);
                emit LogDepositeMade(msg.sender , withdrawAmount);
                return _balances[msg.sender];

        }

        function viewBalance() public view returns (uint)
        {
            return _balances[msg.sender];
        }

}
```


Deployed Contracts
DECENTRALIZEDBANK AT 0X793...
Balance: 2 ETH
deposit
withdraw
_amount:
3000000000000000000
Calldata
Parameters
transact
balances
:
0xdD870fA1b7C4700F2BD7f44:
Calldata
Parameters
call
0: uint256: 5000000000000000000
getBal
0: uint256: 2000000000000000000

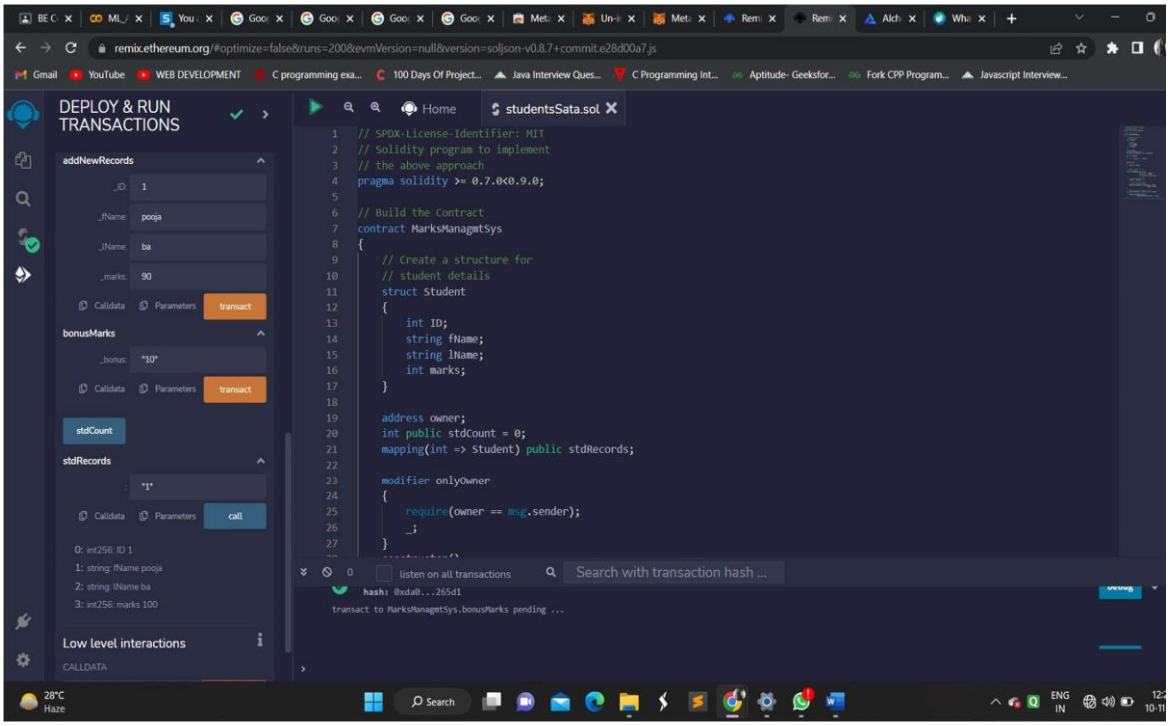
Assignment 4 :

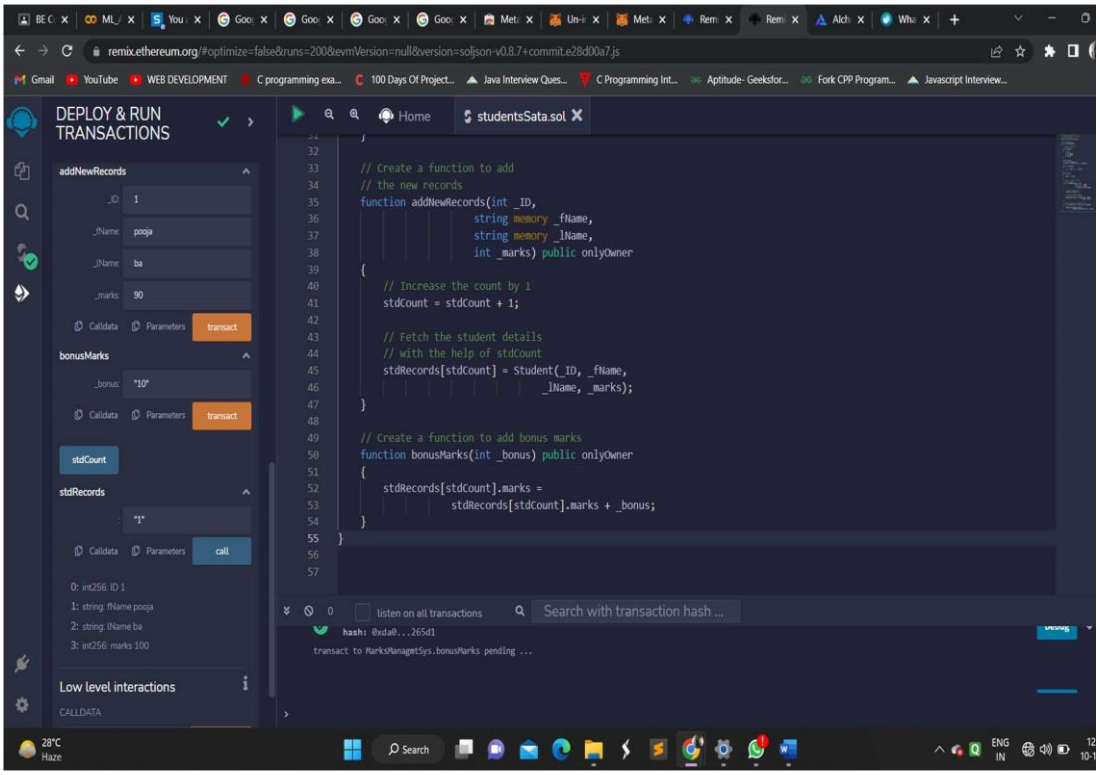
Input :

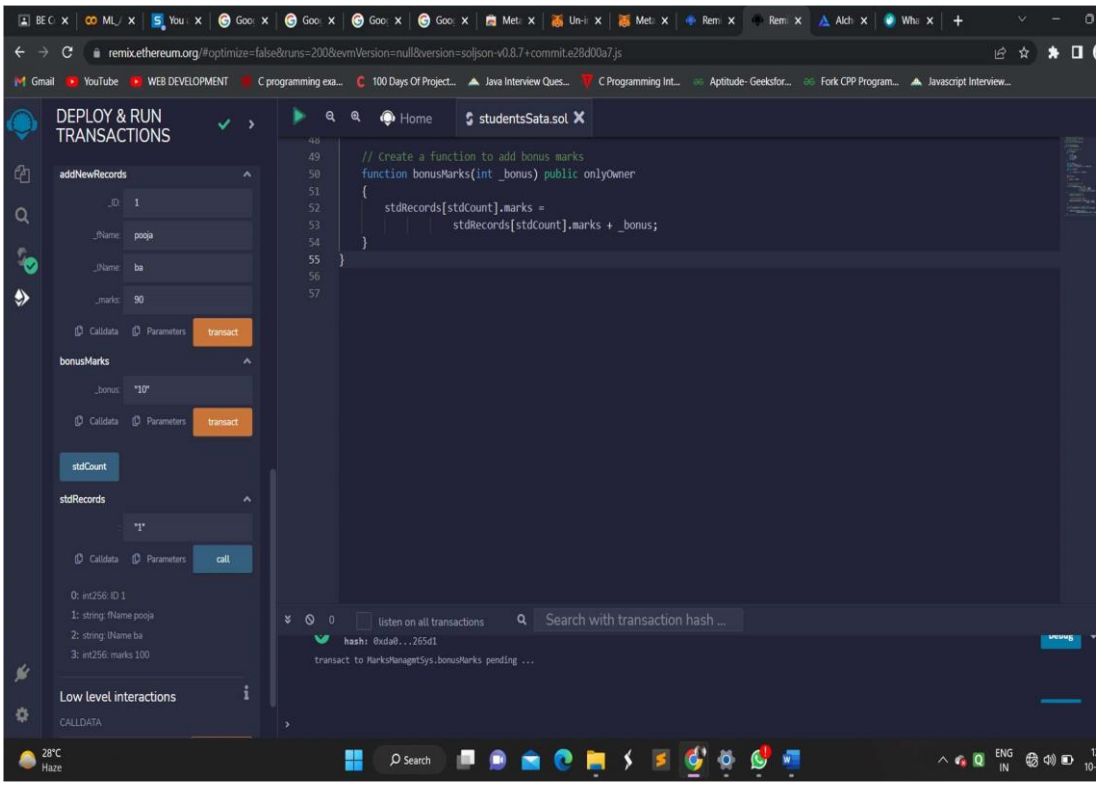
```
// SPDX-License-Identifier: MIT
// Solidity program to implement // the above
approach pragma solidity >= 0.7.0<0.9.0;

// Build the Contract contract
MarksManagmtSys
{
        // Create a structure for          //
student details  struct Student
  {
                int ID;          string
fName;                  string lName;
        int marks;
        }

        address owner;          int public stdCount = 0;
mapping(int => Student) public stdRecords;      modifier
onlyOwner
        {
                require(owner == msg.sender);_;
        }
         constructor()
        {
                owner=msg.sender;
        }
```

```
        // Create a function to add         // the
new records
        function addNewRecords(int _ID, string memory _fName, string memory _lName, int
_marks) public onlyOwner
        {
                // Increase the count by 1
stdCount = stdCount + 1;

                // Fetch the student details
// with the help of stdCount
                        stdRecords[stdCount] = Student(_ID, _fName, _lName, _marks);
        }

        // Create a function to add bonus marks             function
bonusMarks(int _bonus) public onlyOwner
        {
                stdRecords[stdCount].marks =
                                        stdRecords[stdCount].marks + _bonus;
        }
}
```

Output :


remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js
DEPLOY & RUN TRANSACTIONS
addNewRecords
_ID: 1
_fName: pooja
_lName: ba
_marks: 90
Calldata
Parameters
transact
bonusMarks
_bonus: "10"
Calldata
Parameters
transact
stdCount
stdRecords
"1"
Calldata
Parameters
call
0: int256: ID 1
1: string: fName pooja
2: string: lName ba
3: int256: marks 100
Low level interactions
CALLDATA
Home
studentsSata.sol
// SPDX-License-Identifier: MIT
// Solidity program to implement
// the above approach
pragma solidity >= 0.7.0<0.9.0;
// Build the Contract
contract MarksManagmtSys
{
// Create a structure for
// student details
struct Student
{
int ID;
string fName;
string lName;
int marks;
}
address owner;
int public stdCount = 0;
mapping(int => Student) public stdRecords;
modifier onlyOwner
{
require(owner == msg.sender);
_;
}
listen on all transactions
Search with transaction hash ...
hash: 0xda0...265d1
transact to MarksManagmtSys.bonusMarks pending ...
28°C
Haze
Search
ENG
IN


DEPLOY & RUN TRANSACTIONS
addNewRecords
_ID: 1
_fName: pooja
_lName: ba
_marks: 90
Calldata
Parameters
transact
bonusMarks
_bonus: "10"
Calldata
Parameters
transact
stdCount
stdRecords
"1"
Calldata
Parameters
call
0: int256: ID 1
1: string: fName pooja
2: string: lName ba
3: int256: marks 100
Low level interactions
CALLDATA
Home
studentsSata.sol
// Create a function to add
// the new records
function addNewRecords(int _ID,
string memory _fName,
string memory _lName,
int _marks) public onlyOwner
{
// Increase the count by 1
stdCount = stdCount + 1;
// Fetch the student details
// with the help of stdCount
stdRecords[stdCount] = Student(_ID, _fName,
_lName, _marks);
}
// Create a function to add bonus marks
function bonusMarks(int _bonus) public onlyOwner
{
stdRecords[stdCount].marks =
stdRecords[stdCount].marks + _bonus;
}
}
listen on all transactions
Search with transaction hash ...
hash: 0xda0...265d1
transact to MarksManagmtSys.bonusMarks pending ...


DEPLOY & RUN TRANSACTIONS
addNewRecords
_ID: 1
_fName: pooja
_lName: ba
_marks: 90
Calldata
Parameters
transact
bonusMarks
_bonus: "10"
Calldata
Parameters
transact
stdCount
stdRecords
"1"
Calldata
Parameters
call
0: int256: ID 1
1: string: fName pooja
2: string: lName ba
3: int256: marks 100
Low level interactions
CALLDATA
Home
studentsSata.sol
// Create a function to add bonus marks
function bonusMarks(int _bonus) public onlyOwner
{
stdRecords[stdCount].marks =
stdRecords[stdCount].marks + _bonus;
}
}
listen on all transactions
Search with transaction hash ...
hash: 0xda0...265d1
transact to MarksManagmtSys.bonusMarks pending ...

Mini-Project

Code:

```
// SPDX-License-Identifier: GPL-3.0 pragma solidity
>=0.7.0 <0.9.0; 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
string name;   // short name (up to 32 bytes)        uint voteCount; //
number of accumulated votes
   }
   address public chairperson;     mapping(address
=> Voter) public voters;     Proposal[] public
proposals;     constructor(string[] 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

         }));
      }
```

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

}

Output:

Home 3_Ballot.sol

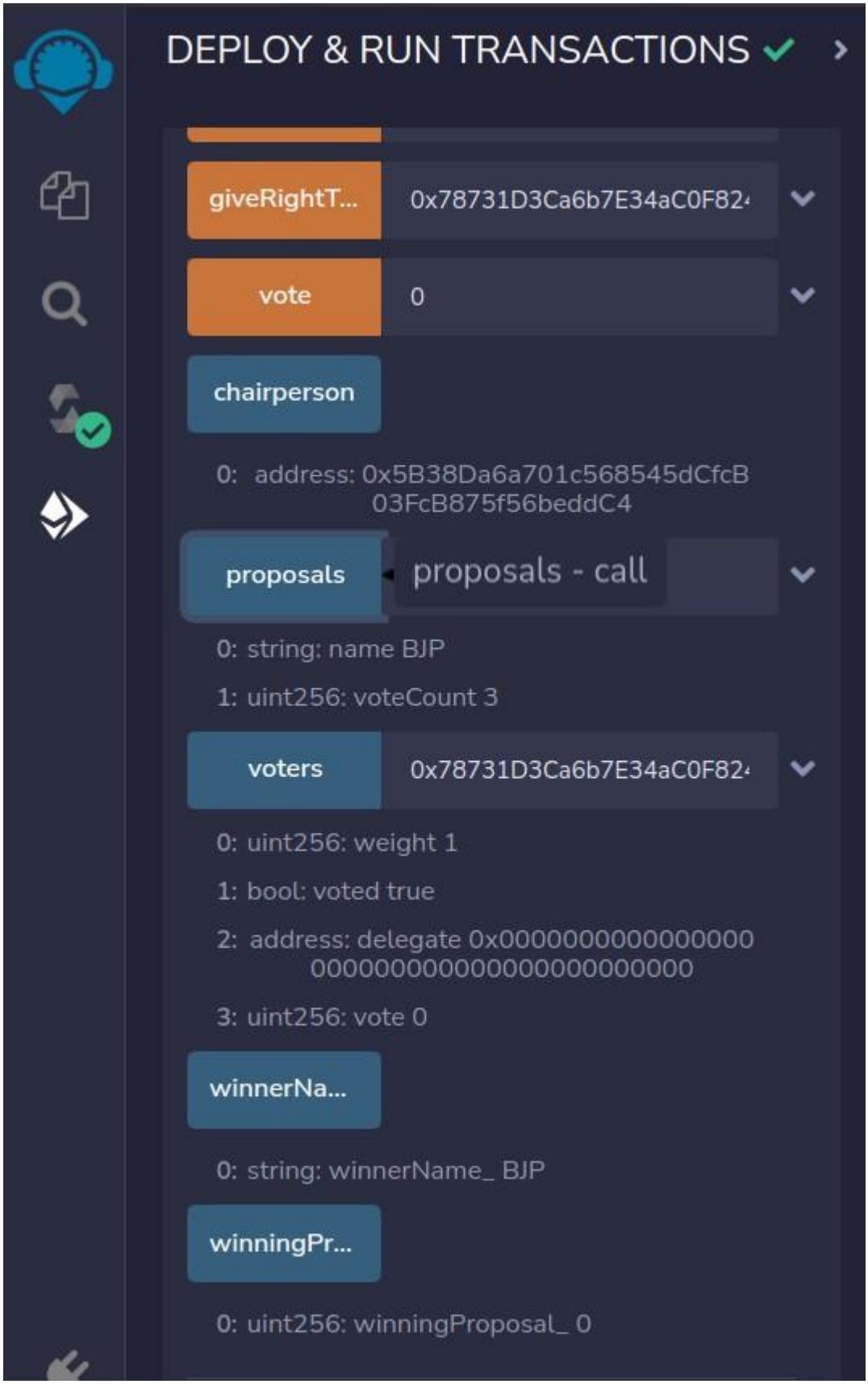
```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

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
        string name;   // short name (up to 32 bytes)
        uint voteCount; // number of accumulated votes
    }

    address public chairperson;

    mapping(address => Voter) public voters;

    Proposal[] public proposals;

    constructor(string[] memory proposalNames) {
        chairperson = msg.sender;
        voters[chairperson].weight = 1;
```

```
    Proposal[] public proposals;

    constructor(string[] 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
            }));
        }
    }


    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;
```

```
    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;
    }


    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,
```

```
    }
    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;
    }
}


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;
}
```

```
96
97      function winningProposal() public view
98              returns (uint winningProposal_)
99      {
100         uint winningVoteCount = 0;
101         for (uint p = 0; p < proposals.length; p++) {
102             if (proposals[p].voteCount > winningVoteCount) {
103                 winningVoteCount = proposals[p].voteCount;
104                 winningProposal_ = p;
105             }
106         }
107     }
108
109     function winnerName() public view
110             returns (string memory winnerName_)
111     {
112         winnerName_ = proposals[winningProposal()].name;
113     }
114 }
```


DEPLOY & RUN TRANSACTIONS
giveRightT...
0x78731D3Ca6b7E34aC0F82
vote
0
chairperson
0: address: 0x5B38Da6a701c568545dCfcB
03FcB875f56beddC4
proposals
proposals - call
0: string: name BJP
1: uint256: voteCount 3
voters
0x78731D3Ca6b7E34aC0F82
0: uint256: weight 1
1: bool: voted true
2: address: delegate 0x0000000000000000
0000000000000000000000
3: uint256: vote 0
winnerNa...
0: string: winnerName_ BJP
winningPr...
0: uint256: winningProposal_ 0