ASSIGNMENT – 3

(Blockchain Technology)

Submitted to: Mr. Shashikant

Name: Ishit Singh

Class: 3NC1

Roll No.: 102115023

Semester: Jan’24-May’24

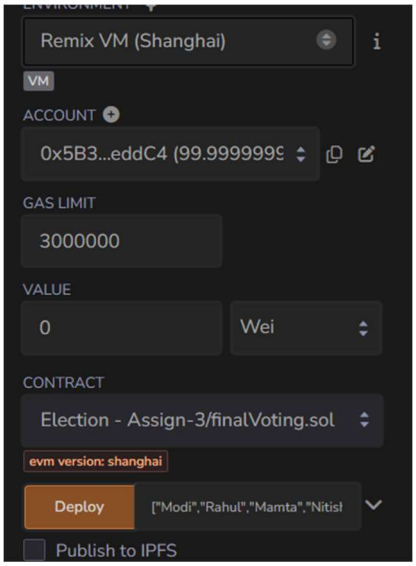
CODE:

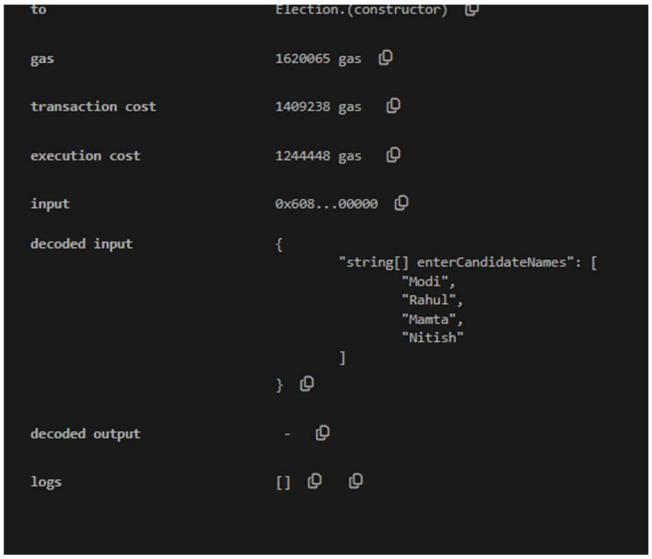
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.13;

contract Election{

    // STEP - 1: Initialize the components of candidate and voter
    struct Voter{
        uint candidateIndexVote;
        address delegatedPerson;
        uint voteWeight;
        bool isVoted;
        uint voterAge;
    }

    struct Candidate{
        string nameOfCandidate;
        uint voteCount;
    }


    // STEP - 2: Define the "State Variables" required
    address public electionHeadAddr;
    Candidate[] public candidateList;
    mapping(address => Voter) public allVotersMapped;


    // STEP - 3: Initialise the "State Variables" (define weight of vote of
"electionHead" as well)
    constructor(string[] memory enterCandidateNames){
        electionHeadAddr = msg.sender;
        allVotersMapped[electionHeadAddr].voteWeight = 1;

        for(uint i=0; i<enterCandidateNames.length; i++){
            candidateList.push(Candidate({nameOfCandidate:
enterCandidateNames[i], voteCount: 0}));
        }
    }


    // STEP - 4: Modifier to check if a particular function is only controlled
by the "ELECTION HEAD"
    modifier onlyElectionHead(){
        require(msg.sender == electionHeadAddr, "Only ELECTION HEAD can run
this function");
        _;
    }
```

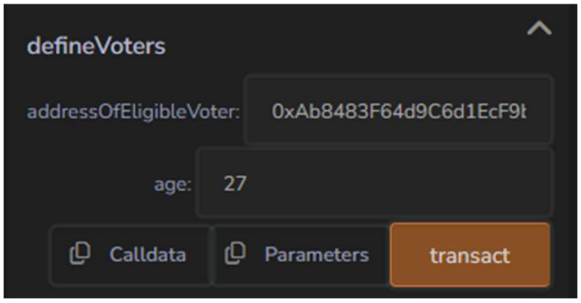
```
    // STEP - 5: Function that gives right to people who can vote [Controlled
by "ELECTION HEAD" only (using modifier)]
    function defineVoters(address addressOfEligibleVoter, uint age) public
onlyElectionHead{
        allVotersMapped[addressOfEligibleVoter].voterAge = age;
        require(allVotersMapped[addressOfEligibleVoter].voterAge >= 18, "Voter
is under-18 and Ineligible");
        require(allVotersMapped[addressOfEligibleVoter].isVoted == false,
"Voter has already voted");
        require(allVotersMapped[addressOfEligibleVoter].voteWeight == 0);
        allVotersMapped[addressOfEligibleVoter].voteWeight = 1;
    }


    // STEP - 6: Function to assign delegate , check delegate contraints and
add to vote count
    /*      CHECK FOR
        1. Sender hasn't voted
        2. Sender not self delgating
        3. Loop to check for any delegation chain
        ----------------------------------------
            ASSIGN
        1. Delegate to sender
        2. Sender has now voted
        3. Check if delegate has already voted or not and accordings either
change the (weight of delegate vote) OR (change candidate vote count)
    */
    function delegationProcess(address delegated) public{
        Voter storage sender = allVotersMapped[msg.sender];

        require(sender.isVoted = false, "Voter has already voted");
        require(delegated != msg.sender,"Self-Delegation NOT ALLOWED");
        while(allVotersMapped[delegated].delegatedPerson != address(0)){
            delegated = allVotersMapped[delegated].delegatedPerson;

            require(delegated != msg.sender,"Found Delegation Loop");
        }

        sender.isVoted = true;
        sender.delegatedPerson = delegated;
        if(allVotersMapped[delegated].isVoted == true){
            candidateList[sender.candidateIndexVote].voteCount +=
sender.voteWeight;
        }
        else{
            allVotersMapped[delegated].voteWeight += sender.voteWeight;
        }
```

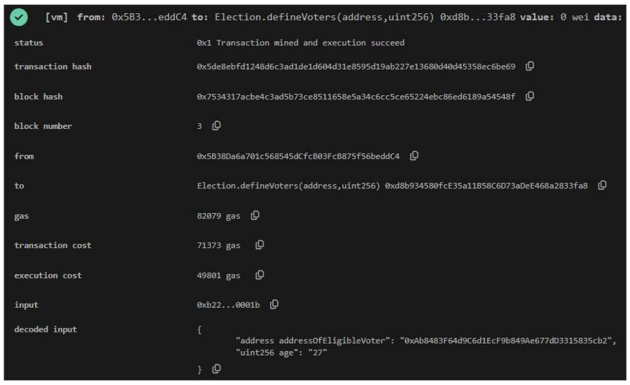
```
    }


    // STEP - 7: Function for people to vote
    /*          CHECK FOR
        1. Current account owner doesn't have "0" voteWeight
        2. Cuurent account owner hasn't already voted
        ---------------------------------------------
                ASSIGN
        1. Current account has now voted
        2. Current account voted for candidate at index --> IDX in the
candidateList
        3. Add the voting account's vote to the total vote of the respective
candidate
    */
    function vote(uint idx) public{
        Voter storage sender = allVotersMapped[msg.sender];

        require(sender.voteWeight != 0, "Not eligible to vote");
        require(sender.isVoted == false, "You have already voted");

        sender.isVoted = true;
        sender.candidateIndexVote = idx;
        candidateList[idx].voteCount += sender.voteWeight;
    }


    // STEP - 8: Function to find the winning candidate by constantly checking
total votes of all candidates
    function winningCandidate() public view returns(uint winnerCandidate){
        uint maxVoteCount = 0;
        for(uint c=0; c<candidateList.length; c++){
            if(candidateList[c].voteCount > maxVoteCount){
                maxVoteCount = candidateList[c].voteCount;
                winnerCandidate = c;
            }
        }
    }


    // STEP - 9: Function to declare name of winning candidate i.e. Candidate
with maximum vote count
    function  winnerName() public view returns(string memory){
        return(candidateList[winningCandidate()].nameOfCandidate);
    }
}
```

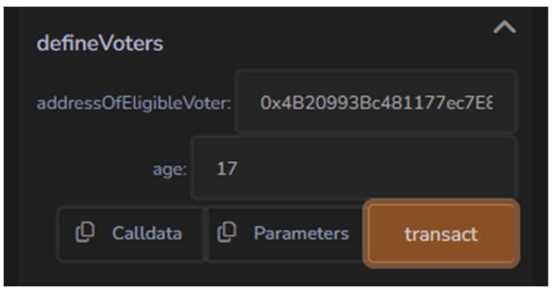
OUTPUT:


Remix VM (Shanghai)
VM
ACCOUNT
0x5B3...eddC4 (99.999999
GAS LIMIT
3000000
VALUE
0
Wei
CONTRACT
Election - Assign-3/finalVoting.sol
evm version: shanghai
Deploy
["Modi","Rahul","Mamta","Nitis
Publish to IPFS

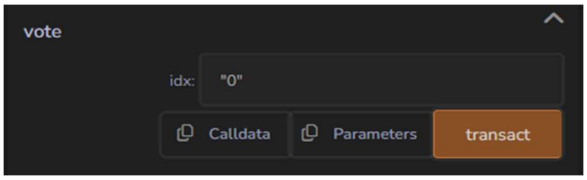

to
Election.(constructor)
gas
1620065 gas
transaction cost
1409238 gas
execution cost
1244448 gas
input
0x608...00000
decoded input
{
"string[] enterCandidateNames": [
"Modi",
"Rahul",
"Mamta",
"Nitish"
]
}
decoded output
-
logs
[]

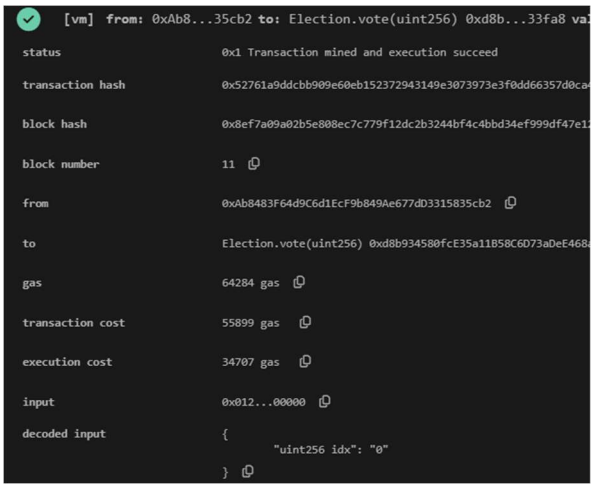

defineVoters
addressOfEligibleVoter:
0xAb8483F64d9C6d1EcF9t
age:
27
Calldata
Parameters
transact

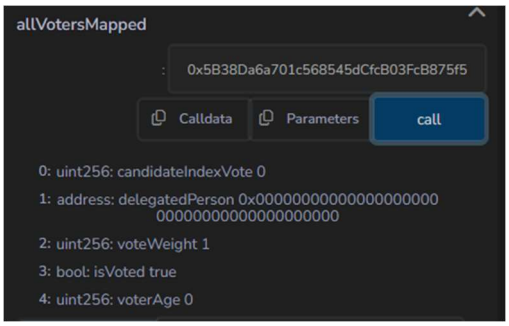

[vm] from: 0x5B3...eddC4 to: Election.defineVoters(address,uint256) 0xd8b...33fa8 value: 0 wei data:
status
0x1 Transaction mined and execution succeed
transaction hash
0x5de8ebfd1248d6c3ad1de1d604d31e8595d19ab227e13680d40d45358ec6be69
block hash
0x7534317acbe4c3ad5b73ce8511658e5a34c6cc5ce65224ebc86ed6189a54548f
block number
3
from
0x5B38Da6a701c568545dCfcB03Fc8875f56beddC4
to
Election.defineVoters(address,uint256) 0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8
gas
82079 gas
transaction cost
71373 gas
execution cost
49801 gas
input
0xb22...0001b
decoded input
{
"address addressOfEligibleVoter": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
"uint256 age": "27"
}


defineVoters
addressOfEligibleVoter:
0x4B20993Bc481177ec7E8
age:
17
Calldata
Parameters
transact

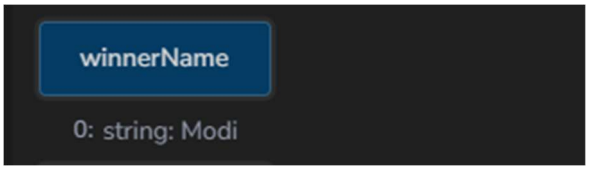

[vm] from: 0x5B3...eddC4 to: Election.defineVoters(address,uint256) 0xd8b...33fa8 value: 0 wei data: 0xb22
status
0x0 Transaction mined but execution failed
transaction hash
0xab83d6cf88584d75516102044ad663fafea94dd616e71c651dbf4d7ec85a8fa5
block hash
0x7f7ee73a6a404bbfa31e8fe2058b0e84b562ab0ebae985f0e21af37ed8c53511
block number
4
from
0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to
Election.defineVoters(address,uint256) 0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8
gas
3000000 gas
transaction cost
47066 gas
execution cost
25494 gas
input
0xb22...00011
decoded input
{
"address addressOfEligibleVoter": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db",
"uint256 age": "17"
}

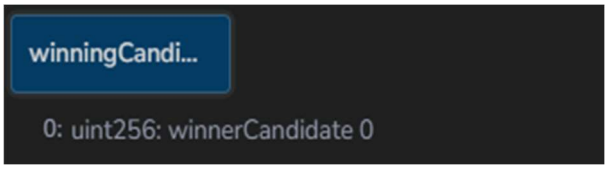

vote
idx:
"0"
Calldata
Parameters
transact

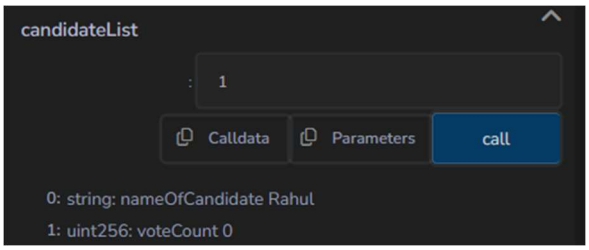

[vm] from: 0xAb8...35cb2 to: Election.vote(uint256) 0xd8b...33fa8
status
0x1 Transaction mined and execution succeed
transaction hash
block hash
block number
11
from
0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
to
Election.vote(uint256) 0xd8b934580fcE35a11B58C6D73aDeE468
gas
64284 gas
transaction cost
55899 gas
execution cost
34707 gas
input
0x012...00000
decoded input
{
"uint256 idx": "0"
}

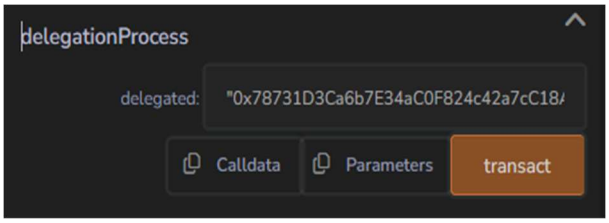

allVotersMapped
0x5B38Da6a701c568545dCfcB03FcB875f5
Calldata
Parameters
call
0: uint256: candidateIndexVote 0
1: address: delegatedPerson 0x0000000000000000000000000000000000000000
2: uint256: voteWeight 1
3: bool: isVoted true
4: uint256: voterAge 0


electionHeadA...
0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56
beddC4


winnerName
0: string: Modi


winningCandi...
0: uint256: winnerCandidate 0


candidateList
1
Calldata
Parameters
call
0: string: nameOfCandidate Rahul
1: uint256: voteCount 0


delegationProcess
delegated:
"0x78731D3Ca6b7E34aC0F824c42a7cC18
Calldata
Parameters
transact