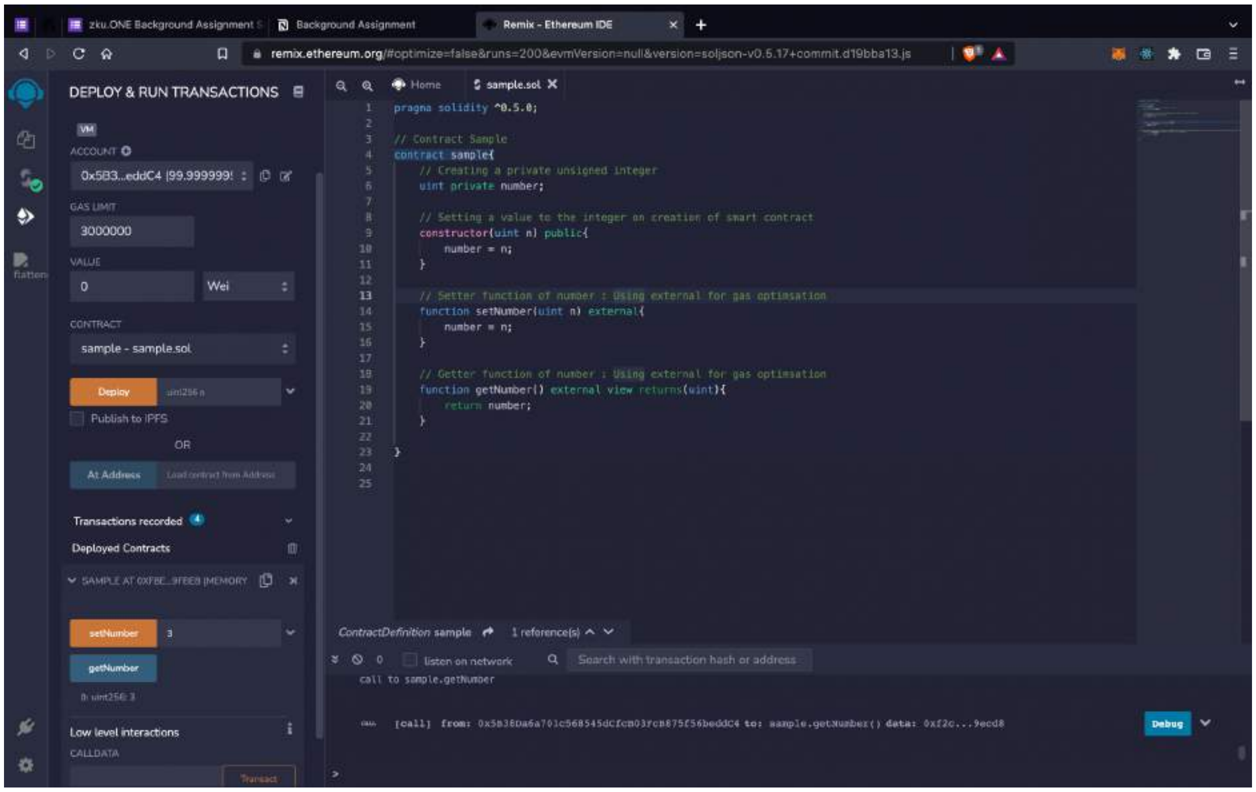
ZKU.One Background Assignment

SHIVAM SHARMA
shivam691999@gmail.com

Q1)

Screenshot :


zku.ONE Background Assignment
Background Assignment
Remix - Ethereum IDE
remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js
DEPLOY & RUN TRANSACTIONS
Home
sample.sol
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded 4
Deployed Contracts
setNumber
getNumber
Low level interactions
CALLDATA
Transact
ContractDefinition sample
1 reference(s)
listen on network
Search with transaction hash or address
Debug

VM
ACCOUNT: 0x5B3...eddC4 (99.999999!
GAS LIMIT: 3000000
VALUE: 0 Wei
CONTRACT: sample - sample.sol
Deploy: uint256 n
SAMPLE AT 0XFBE...9FEE8 (MEMORY)
setNumber: 3
getNumber
0: uint256: 3

```
pragma solidity ^0.5.0;

// Contract Sample
contract sample{
    // Creating a private unsigned integer
    uint private number;

    // Setting a value to the integer on creation of smart contract
    constructor(uint n) public{
        number = n;
    }

    // Setter function of number : Using external for gas optimsation
    function setNumber(uint n) external{
        number = n;
    }

    // Getter function of number : Using external for gas optimsation
    function getNumber() external view returns(uint){
        return number;
    }

}
```

```
call to sample.getNumber

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: sample.getNumber() data: 0xf2c...9ecd8
```

.sol file

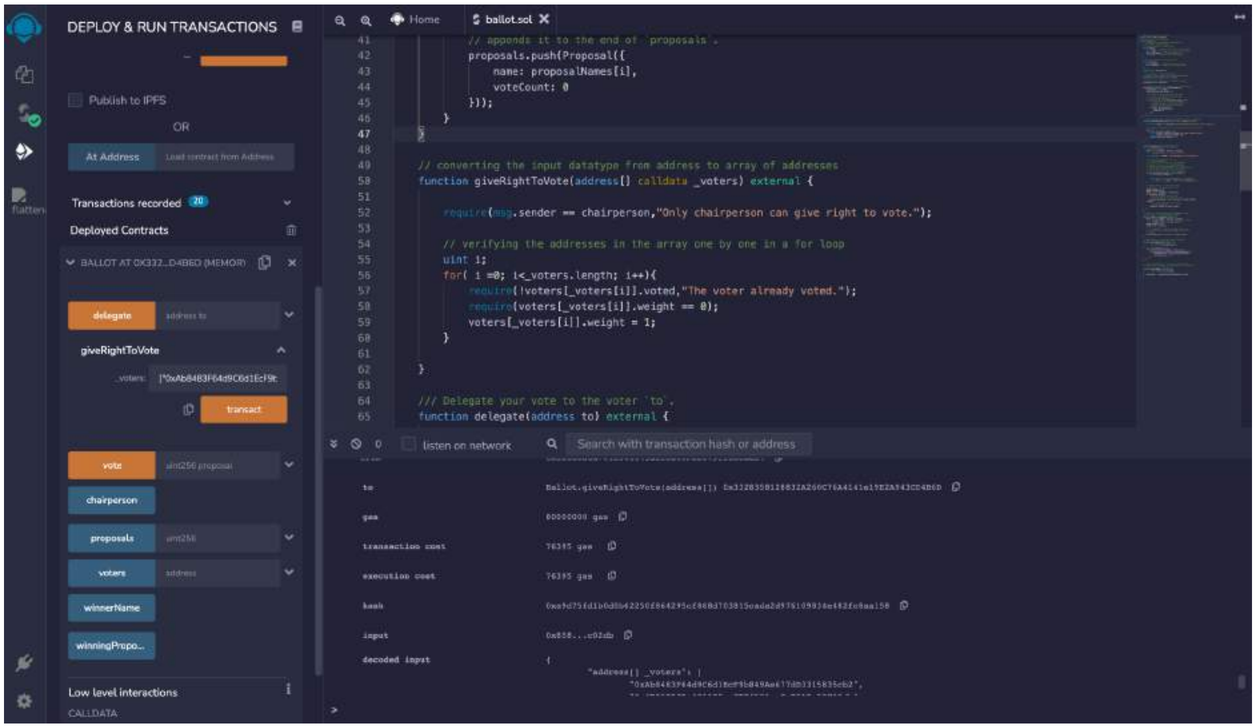
```
pragma solidity ^0.5.0;

// Contract Sample
contract sample{
   // Creating a private unsigned integer
   uint private number;

   // Setting a value to the integer on creation of smart contract
   constructor(uint n) public{
       number = n;
```

```
    }

    // Setter function of number : Using external for gas optimsation
    function setNumber(uint n)  external{
        number = n;
    }

    // Getter function of number : Using external for gas optimsation
    function getNumber() external view returns(uint){
        return number;
    }

}
```

Q2)

Gas Before = 10 * 48657 = ~486570
Gas After Optimisation = 76395

<u>**Screenshot :**</u>



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

// converting the input datatype from address to array of addresses
function giveRightToVote(address[] calldata _voters) external {

    require(msg.sender == chairperson,"Only chairperson can give right to vote.");

    // verifying the addresses in the array one by one in a for loop
    uint i;
    for( i =0; i<_voters.length; i++){
        require(!voters[_voters[i]].voted,"The voter already voted.");
        require(voters[_voters[i]].weight == 0);
        voters[_voters[i]].weight = 1;
    }

}

/// Delegate your vote to the voter `to`.
function delegate(address to) external {
```

```
to                  Ballot.giveRightToVote(address[]) 0x32283581288322A260C76A4141e19E2A943CD4B6D
gas                 80000000 gas
transaction cost    76395 gas
execution cost      76395 gas
hash                0xe9d75fd1b0d8b42250f864295cf868d703815eada2d97610983 4e482fe8aa158
input               0x858...c02db
decoded input       {
                        "address[] _voters": [
                            "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
```

Solution: converting the input argument data-type from address to array of addresses and verifying the addresses in the array one by one in a ‘for’ loop.
Also, comments added in the code.

.sol file

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

contract Ballot {

    struct Voter {
        uint weight;
        bool voted;
        address delegate;
        uint vote;
    }


    struct Proposal {
        bytes32 name;
        uint voteCount;
    }

    address public chairperson;

    mapping(address => Voter) public voters;


    Proposal[] public proposals;


    constructor(bytes32[] memory proposalNames) {
        chairperson = msg.sender;
        voters[chairperson].weight = 1;


        for (uint i = 0; i < proposalNames.length; i++) {

            proposals.push(Proposal({
                name: proposalNames[i],
                voteCount: 0
            }));
        }
    }

    // converting the input datatype from address to array of addresses
    function giveRightToVote(address[] calldata _voters) external {

        require(msg.sender == chairperson,"Only chairperson can give right to vote.");

        // verifying the addresses in the array one by one in a for loop
        uint i;
        for( i =0; i<_voters.length; i++){
            require(!voters[_voters[i]].voted,"The voter already voted.");
            require(voters[_voters[i]].weight == 0);
            voters[_voters[i]].weight = 1;
        }

    }


    function delegate(address to) external {

        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;

            require(to != msg.sender, "Found loop in delegation.");
        }


        sender.voted = true;
        sender.delegate = to;
        Voter storage delegate_ = voters[to];
        if (delegate_.voted) {

            proposals[delegate_.vote].voteCount += sender.weight;
        } else {
            delegate_.weight += sender.weight;
        }
    }


    function vote(uint proposal) external {
        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;

        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() external view
            returns (bytes32 winnerName_)
    {
        winnerName_ = proposals[winningProposal()].name;
    }
}
```