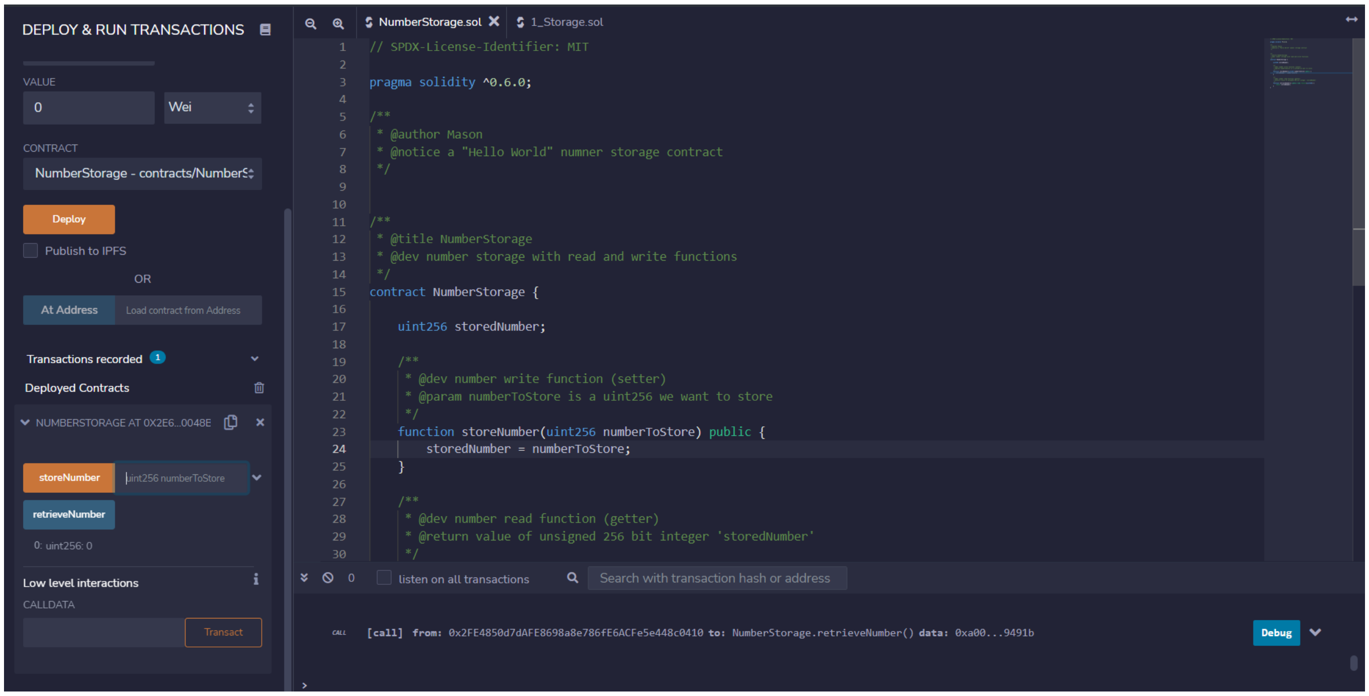
Part A:

1. A smart contract is a self-executing program that runs on and changes a blockchain. Smart contracts are deployed via a transaction to no specified address. The program is included in its bytecode form (compiled beforehand) as part of the transactions data; because it is a transaction we also need to have a connection to the network, adequate gas, and a deployment script. On the Ethereum network, the bytecode will be processed by the EVM.
2. Gas is how the blockchain charges for the ability to do computations. The collected gas is used to compensate node operators or is burned. A transaction originator chooses how much to pay per gas, paying more means your transaction is more attractive for node operators. Optimization is important because if you run out before a transaction is complete your gas is lost and changes reverted. Also, inefficient contracts make the block size bigger, and a sustained large size will increase the base fee.
3. A hash is a fixed-length unique identifier for a certain set of input data. You acquire a hash by passing your input to a hashing function. It's used to hide information because it's no longer plain text and stills allows for authentication. This is because the same input will always create the same hash.
4. Because we cannot simply show the difference in color to the person we have to show that we can consistently identify the object's difference. To prove the difference we can look at an object and then out of our view the object can be possibly switched. The person can then ask us to look at the object and ask if the object has changed. If this is repeated many times successfully it should be convincing to the person that there is a difference. If we misidentify a switch then the person knows that we are not able to see a difference.

Part B:

link to projects on github

hello world



VALUE 0 Wei

CONTRACT NumberStorage - contracts/NumberS

NUMBERSTORAGE AT 0X2E6...0048E

storeNumber uint256 numberToStore

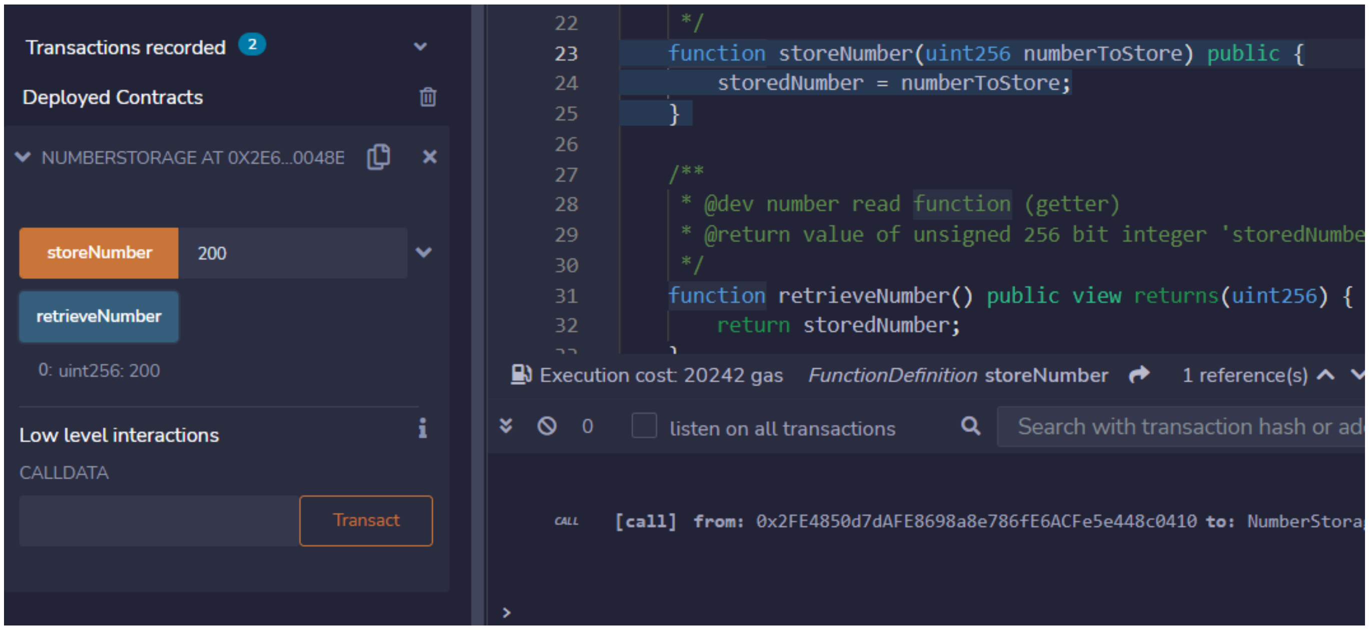
0: uint256: 0

CALLDATA

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.6.0;

/**
 * @author Mason
 * @notice a "Hello World" numner storage contract
 */


/**
 * @title NumberStorage
 * @dev number storage with read and write functions
 */
contract NumberStorage {

    uint256 storedNumber;

    /**
     * @dev number write function (setter)
     * @param numberToStore is a uint256 we want to store
     */
    function storeNumber(uint256 numberToStore) public {
        storedNumber = numberToStore;
    }

    /**
     * @dev number read function (getter)
     * @return value of unsigned 256 bit integer 'storedNumber'
     */
```

listen on all transactions

```
[call] from: 0x2FE4850d7dAFE8698a8e786fE6ACFe5e448c0410 to: NumberStorage.retrieveNumber() data: 0xa00...9491b
```



NUMBERSTORAGE AT 0X2E6...0048E

storeNumber 200

0: uint256: 200

CALLDATA

```
    */
    function storeNumber(uint256 numberToStore) public {
        storedNumber = numberToStore;
    }

    /**
     * @dev number read function (getter)
     * @return value of unsigned 256 bit integer 'storedNumbe
     */
    function retrieveNumber() public view returns(uint256) {
        return storedNumber;
```

Execution cost: 20242 gas *FunctionDefinition* storeNumber 1 reference(s)

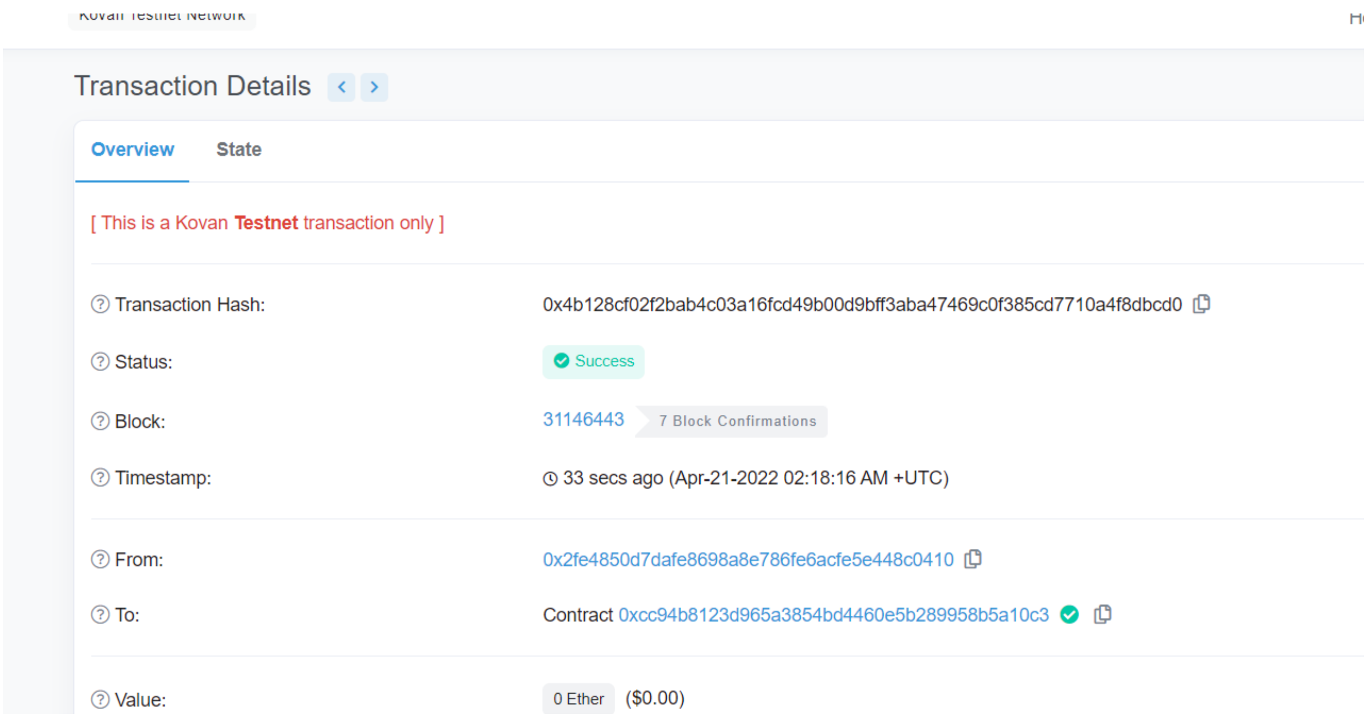
listen on all transactions

```
[call] from: 0x2FE4850d7dAFE8698a8e786fE6ACFe5e448c0410 to: NumberStora
```

voting

Note: for purposes of not having to wait 5 minutes I used a timeout of 1 minute for this demonstration

first I deployed the contract
then I voted with the chairperson account
i added another account to those who can vote
waited for 1 minute after deployment
tried to vote from second account but transaction failed due to the timer
I check that no extra vote was added



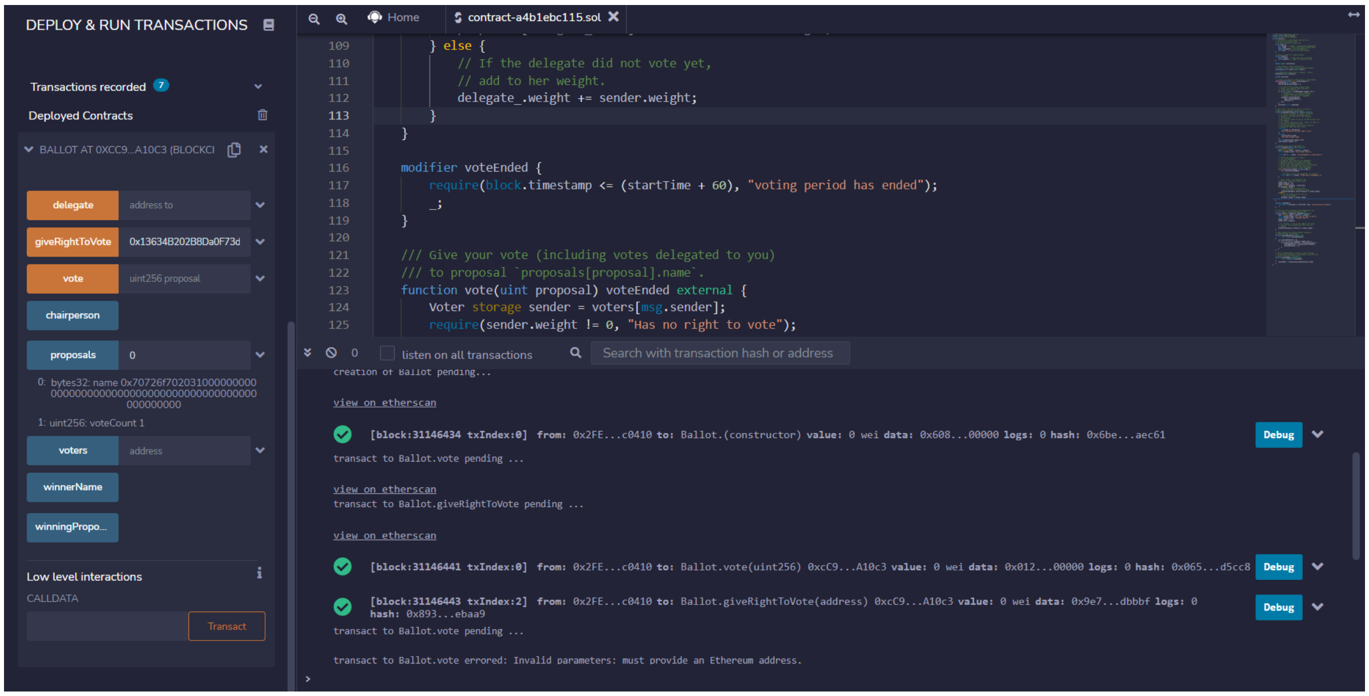
Kovan Testnet Network

Transaction Details

Overview State

[This is a Kovan **Testnet** transaction only]

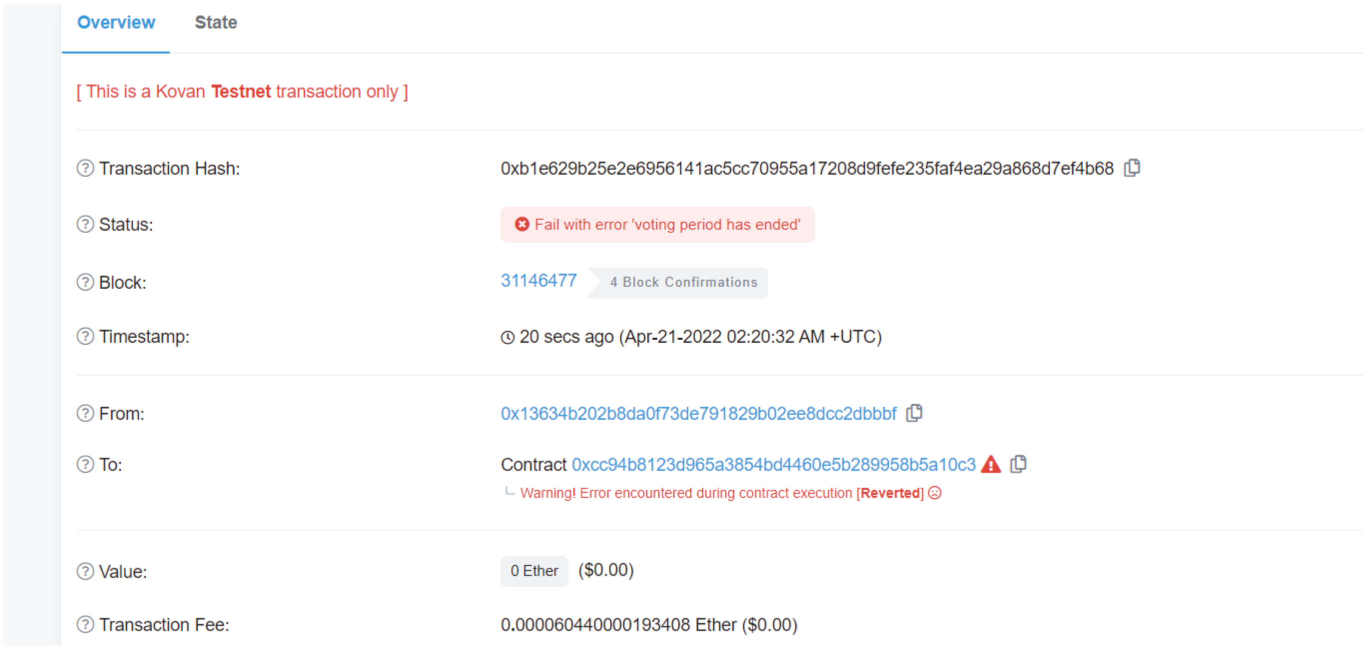
Transaction Hash:	0x4b128cf02f2bab4c03a16fcd49b00d9bff3aba47469c0f385cd7710a4f8dbcd0
Status:	Success
Block:	31146443 7 Block Confirmations
Timestamp:	33 secs ago (Apr-21-2022 02:18:16 AM +UTC)
From:	0x2fe4850d7dafe8698a8e786fe6acfe5e448c0410
To:	Contract 0xcc94b8123d965a3854bd4460e5b289958b5a10c3
Value:	0 Ether ($0.00)



0: bytes32: name 0x70726f702031000000000000000000000000000000000000000000000000000000000
1: uint256: voteCount 1

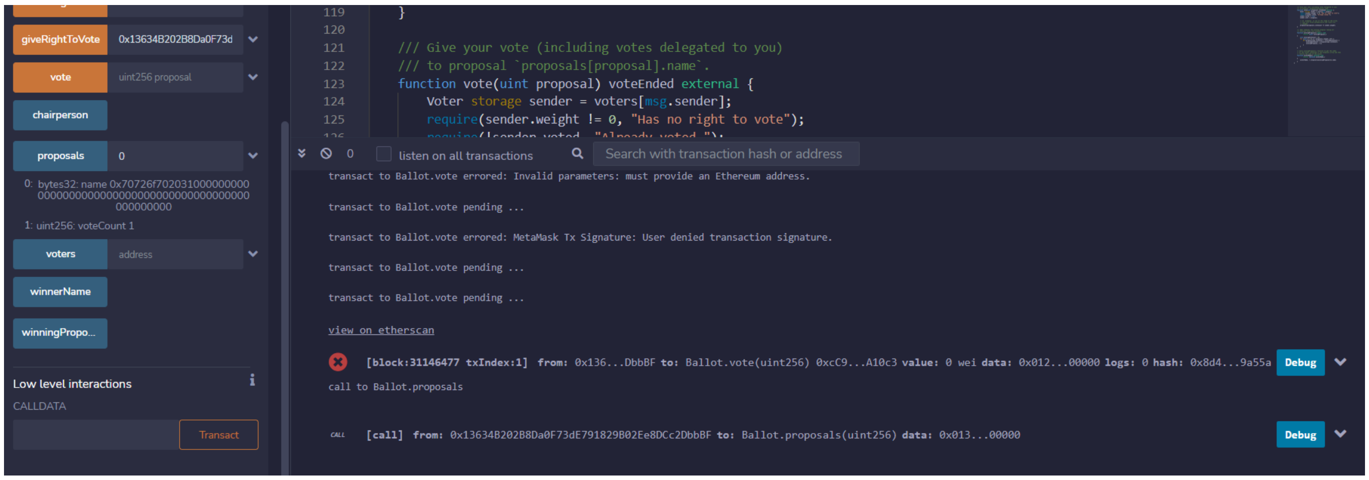
```
        } else {
            // If the delegate did not vote yet,
            // add to her weight.
            delegate_.weight += sender.weight;
        }
    }

    modifier voteEnded {
        require(block.timestamp <= (startTime + 60), "voting period has ended");
        _;
    }

    /// Give your vote (including votes delegated to you)
    /// to proposal `proposals[proposal].name`.
    function vote(uint proposal) voteEnded external {
        Voter storage sender = voters[msg.sender];
        require(sender.weight != 0, "Has no right to vote");
```

```
creation of Ballot pending...

view on etherscan

[block:31146434 txIndex:0] from: 0x2FE...c0410 to: Ballot.(constructor) value: 0 wei data: 0x608...00000 logs: 0 hash: 0x6be...aec61
transact to Ballot.vote pending ...

view on etherscan
transact to Ballot.giveRightToVote pending ...

view on etherscan

[block:31146441 txIndex:0] from: 0x2FE...c0410 to: Ballot.vote(uint256) 0xcC9...A10c3 value: 0 wei data: 0x012...00000 logs: 0 hash: 0x065...d5cc8

[block:31146443 txIndex:2] from: 0x2FE...c0410 to: Ballot.giveRightToVote(address) 0xcC9...A10c3 value: 0 wei data: 0x9e7...dbbbf logs: 0
hash: 0x893...ebaa9
transact to Ballot.vote pending ...

transact to Ballot.vote errored: Invalid parameters: must provide an Ethereum address.
>
```



[This is a Kovan **Testnet** transaction only]

Transaction Hash:	0xb1e629b25e2e6956141ac5cc70955a17208d9fefe235faf4ea29a868d7ef4b68
Status:	Fail with error 'voting period has ended'
Block:	31146477 4 Block Confirmations
Timestamp:	20 secs ago (Apr-21-2022 02:20:32 AM +UTC)
From:	0x13634b202b8da0f73de791829b02ee8dcc2dbbbf
To:	Contract 0xcc94b8123d965a3854bd4460e5b289958b5a10c3 Warning! Error encountered during contract execution [**Reverted**]
Value:	0 Ether ($0.00)
Transaction Fee:	0.000060440000193408 Ether ($0.00)



0: bytes32: name 0x70726f702031000000000000000000000000000000000000000000000000000000
1: uint256: voteCount 1

```
    }

    /// Give your vote (including votes delegated to you)
    /// to proposal `proposals[proposal].name`.
    function vote(uint proposal) voteEnded external {
        Voter storage sender = voters[msg.sender];
        require(sender.weight != 0, "Has no right to vote");
```

```
transact to Ballot.vote errored: Invalid parameters: must provide an Ethereum address.

transact to Ballot.vote pending ...

transact to Ballot.vote errored: MetaMask Tx Signature: User denied transaction signature.

transact to Ballot.vote pending ...

transact to Ballot.vote pending ...

view on etherscan

[block:31146477 txIndex:1] from: 0x136...DbbBF to: Ballot.vote(uint256) 0xcC9...A10c3 value: 0 wei data: 0x012...00000 logs: 0 hash: 0x8d4...9a55a
call to Ballot.proposals

CALL [call] from: 0x13634B202B8Da0F73dE791829B02Ee8DCc2DbbBF to: Ballot.proposals(uint256) data: 0x013...00000
```