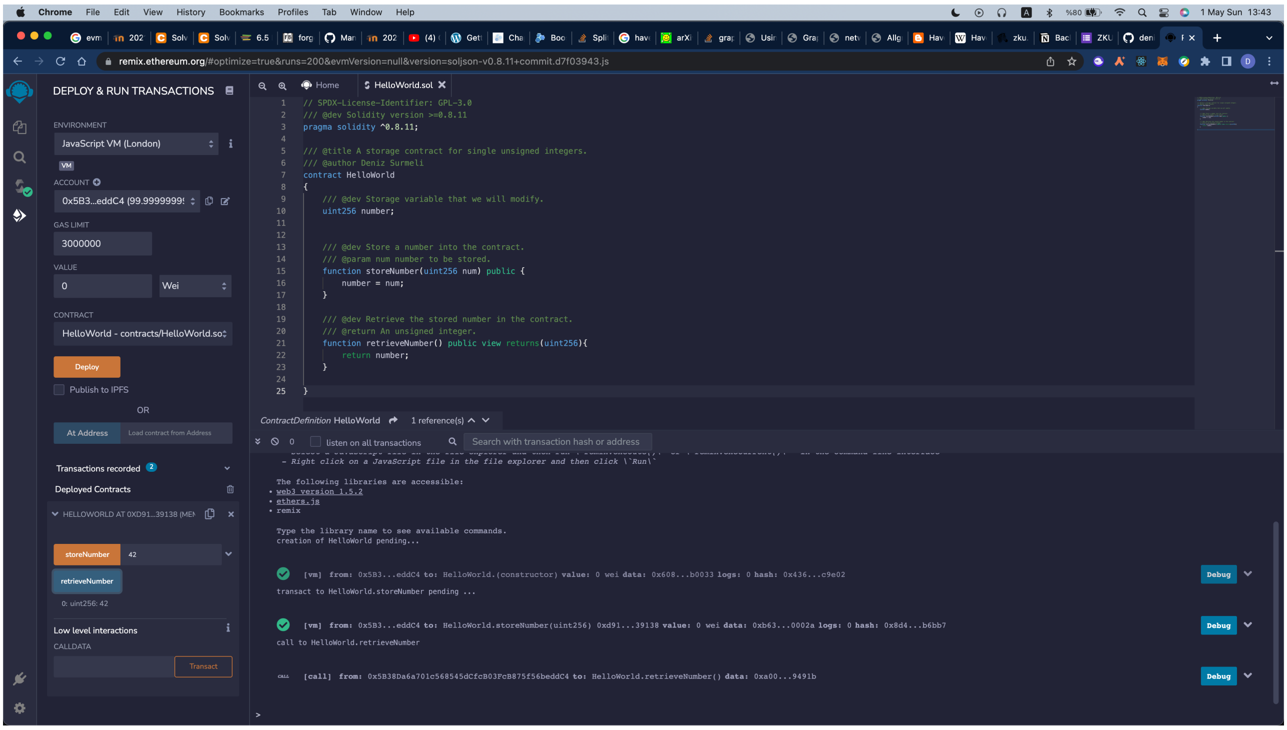
A.Conceptual Knowledge

- 1. A smart contract is an ethereum account where it's actions are regulated by code, not individuals.Just like any ethereum account it can have balance and send/receive transactions, but also it has EVM code and storage. Remember, these accounts are regulated by EVM code. Contract creation is a form of transaction, since we change the ethereum's global state, we deploy contracts by compiling the source code first, where we will achieve a bytecode. Then we will create a transaction by sending a transaction to the empty address with bytecode as data field. After these steps, we have deployed our contract.
- 2. Gas is the unit cost of the computational operation on the network. We need gas for several reasons.First of all, we should incentivize the miners so that they stay honest and continue to produce blocks. Secondly, we should find a way of ordering and accepting transactions. There is a finite amount of blockspace in a block, this is where the gas comes to the play. Miners accept and order the transactions by the fee that transactions yield. This way, they have a "fair" ordering and accepting the number of transactions. Gas optimization is very important for several reasons. Gas optimization yields advantage for each end of the network. Users spend less ether as transaction fees, miners store less data and compute less, thus their costs are reduced.Also since blocks have limited spaces, optimized calls can allow more transactions to be included in a block. At the end, every party wins.
- 3. A hash is a a fixed-size output for an arbitrary-sized input mapped by a cryptographic function. Hashing information can help you to standardize operations, help for faster and more secure lookups. Think this case. A website calcutates the users password hash in the browser, checks the hash on their database in constant time since it will be a hash table. It's faster since hash table lookups are constant time operations, it is more secure because raw password is not transmitted through network and it's standardized since every key is fixed size.
- 4. This is an instance of a problem that can be solved by using Zero Knowledge Proofs. Prover, say Allice, wants to prove to the verifier, say Bob, that two pens she's holding is in different colors, but Bob is colorblind. Alice gives the pens Bob and Bob switches them on their back, shows them to Alice. Bob asks to Alice that whether they are switched or not. The probability of identifiying switch is 1/2. If you perform ,say 100 rounds, the probability of she is catching consecutive all switches for 100 rounds is (1/2)^100 = 7.888609052210118e-31 which is a neglible possibility.

B. You sure you're solid with Solidity ?

- 1)



remix.ethereum.org/#optimize=true&runs=200&evmVersion=null&version=soljson-v0.8.11+commit.d7f03943.js

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: JavaScript VM (London)

ACCOUNT: 0x5B3...eddC4 (99.9999999!

GAS LIMIT: 3000000

VALUE: 0 Wei

CONTRACT: HelloWorld - contracts/HelloWorld.so

Transactions recorded 2

Deployed Contracts

HELLOWORLD AT 0XD91...39138 (MEN

storeNumber 42

retrieveNumber

0: uint256: 42

HelloWorld.sol

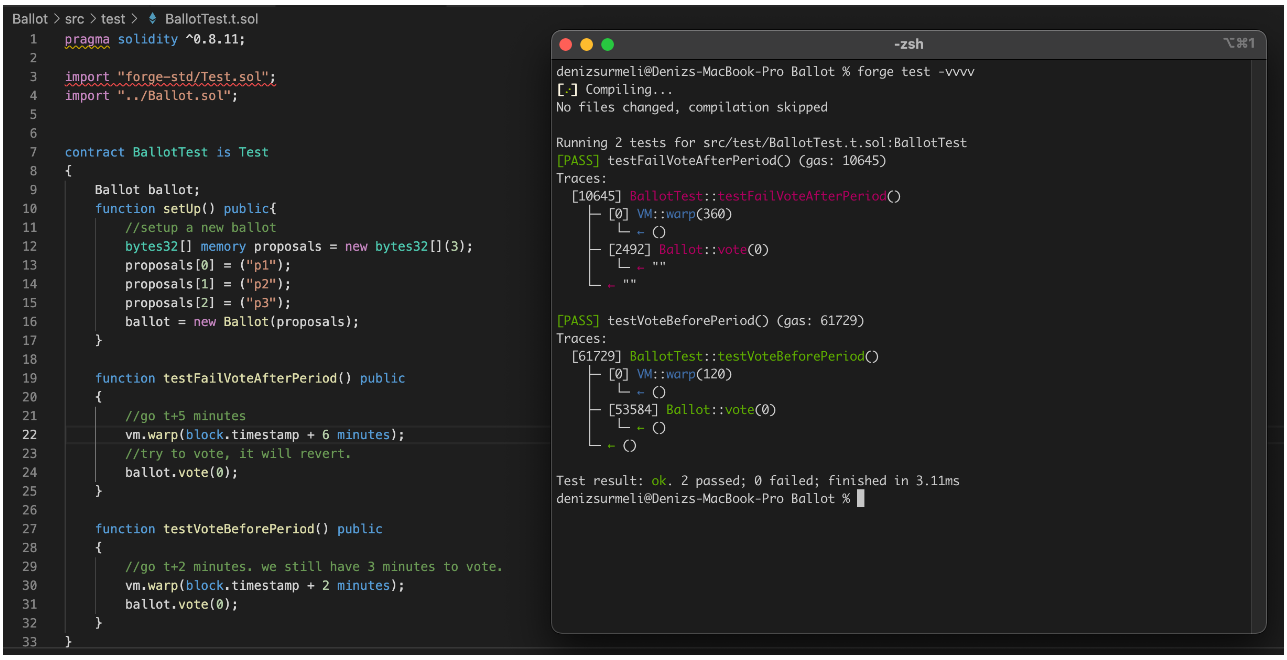
```solidity
// SPDX-License-Identifier: GPL-3.0
/// @dev Solidity version >=0.8.11
pragma solidity ^0.8.11;

/// @title A storage contract for single unsigned integers.
/// @author Deniz Surmeli
contract HelloWorld
{
    /// @dev Storage variable that we will modify.
    uint256 number;


    /// @dev Store a number into the contract.
    /// @param num number to be stored.
    function storeNumber(uint256 num) public {
        number = num;
    }

    /// @dev Retrieve the stored number in the contract.
    /// @return An unsigned integer.
    function retrieveNumber() public view returns(uint256){
        return number;
    }

}
```

```
 - Right click on a JavaScript file in the file explorer and then click \`Run\`

 The following libraries are accessible:
• web3 version 1.5.2
• ethers.js
• remix

 Type the library name to see available commands.
 creation of HelloWorld pending...

 [vm] from: 0x5B3...eddC4 to: HelloWorld.(constructor) value: 0 wei data: 0x608...b0033 logs: 0 hash: 0x436...c9e02
 transact to HelloWorld.storeNumber pending ...

 [vm] from: 0x5B3...eddC4 to: HelloWorld.storeNumber(uint256) 0xd91...39138 value: 0 wei data: 0xb63...0002a logs: 0 hash: 0x8d4...b6bb7
 call to HelloWorld.retrieveNumber

 [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: HelloWorld.retrieveNumber() data: 0xa00...9491b
```

Link to the question's answer in code.

- 2,3,4)See the source code.

- The Ballot subdirectory contains the tests for showing that modified code works as intented. Here is a secreenshot.



Ballot > src > test > BallotTest.t.sol

```solidity
pragma solidity ^0.8.11;

import "forge-std/Test.sol";
import "../Ballot.sol";


contract BallotTest is Test
{
    Ballot ballot;
    function setUp() public{
        //setup a new ballot
        bytes32[] memory proposals = new bytes32[](3);
        proposals[0] = ("p1");
        proposals[1] = ("p2");
        proposals[2] = ("p3");
        ballot = new Ballot(proposals);
    }

    function testFailVoteAfterPeriod() public
    {
        //go t+5 minutes
        vm.warp(block.timestamp + 6 minutes);
        //try to vote, it will revert.
        ballot.vote(0);
    }

    function testVoteBeforePeriod() public
    {
        //go t+2 minutes. we still have 3 minutes to vote.
        vm.warp(block.timestamp + 2 minutes);
        ballot.vote(0);
    }
}
```

-zsh

```
denizsurmeli@Denizs-MacBook-Pro Ballot % forge test -vvvv
[⠊] Compiling...
No files changed, compilation skipped

Running 2 tests for src/test/BallotTest.t.sol:BallotTest
[PASS] testFailVoteAfterPeriod() (gas: 10645)
Traces:
  [10645] BallotTest::testFailVoteAfterPeriod()
    ├─ [0] VM::warp(360)
    │   └─ ← ()
    ├─ [2492] Ballot::vote(0)
    │   └─ ← ""
    └─ ← ""

[PASS] testVoteBeforePeriod() (gas: 61729)
Traces:
  [61729] BallotTest::testVoteBeforePeriod()
    ├─ [0] VM::warp(120)
    │   └─ ← ()
    ├─ [53584] Ballot::vote(0)
    │   └─ ← ()
    └─ ← ()

Test result: ok. 2 passed; 0 failed; finished in 3.11ms
denizsurmeli@Denizs-MacBook-Pro Ballot %
```

You can clone the repo and run the tests if you want to !