Maxim Vezenov
Background Assignment

Github: https://github.com/vezenovm

A. Conceptual Knowledge

1. A smart-contract is a self-executing contract where the interactions between entities and the conditions for execution are dictated by code. Smart contracts exist on blockchain systems that automatically execute code once the specified conditions from the contract are met. In order to actually interact with these contracts they must first be deployed onto a blockchain. Deploying a contract requires sending a transaction on the network containing the compiled code, but not specifying a recipient. In order to deploy a contract there are several tools such as Remix and Truffle that can be used. These tools allow developers to easily compile contracts to their bytecode and scripting tools for deployment. Upon getting access to an Ethereum client a developer can send the necessary transaction and deploy their contract to Ethereum or other EVM based chains.

2. Gas is the unit of measurement Ethereum and other blockchains for the amount of computational effort required by an operation. The more gas a contract uses to execute, the more Ether will be required to call that contract. Smart contract developers focus on gas optimization so much due to this reason. A contract may be more complex to read for a programmer, but actually uses less gas than all other versions of the same functionality.

3. A hash is a cryptographic function that takes input of arbitrary size and maps it to a set bit size encrypted value. A hash is a one-way operation that cannot be reversed. This is very useful for safe storage and verification of information. For example, rather than storing a password in plain text, we would store a hash of that password. When a user wants to login again we would check that the hash of the newly provided password matches what has already been stored. If it does match, we can log the user in.

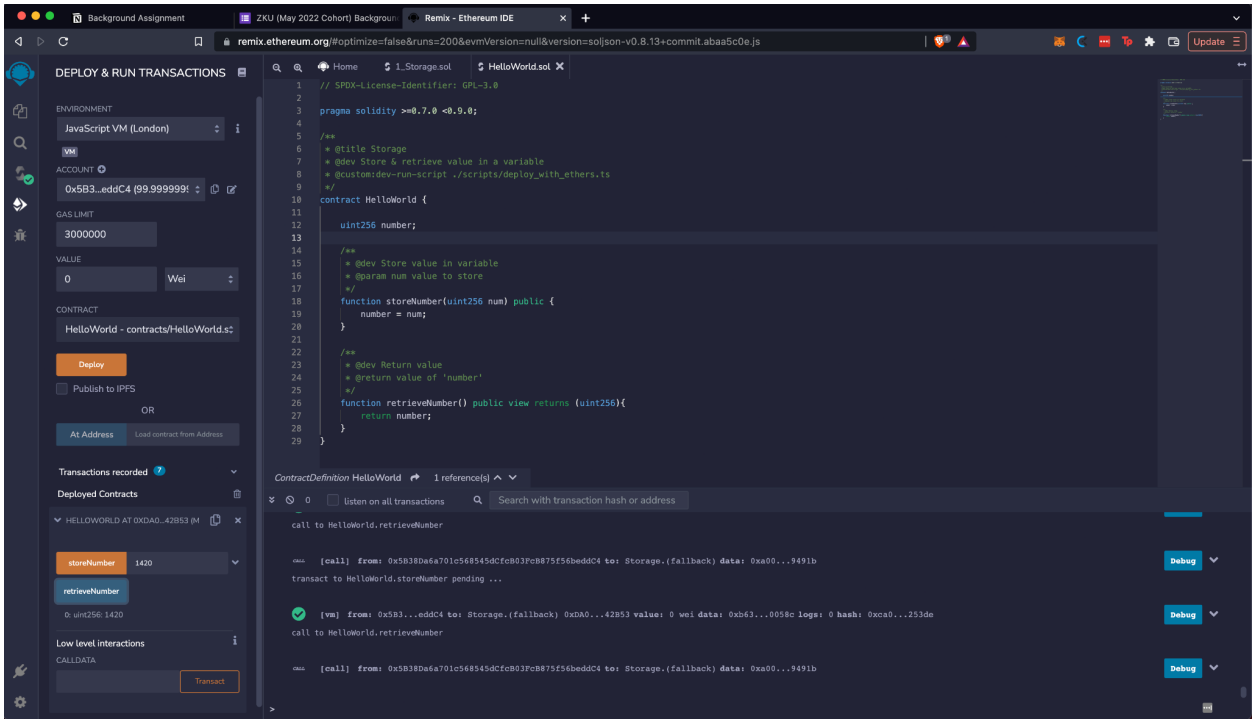
4. You could have the colorblind person put the two objects behind their back. The person can then show either object to you who is not colorblind and note that the object is the color you say. This could be a lucky guess, so the colorblind person can repeat this experiment over and over until they are convinced, switching between the objects. An object must be either one color or the other. After enough trials it becomes mathematically improbable you were able to keep guessing the same color for the same object if the objects were actually the same color.

B. Solidity Tutorials

HelloWorld.sol

Code can be found on Github here:
https://github.com/vezenovm/ZK-University/blob/main/background/HelloWorld.sol

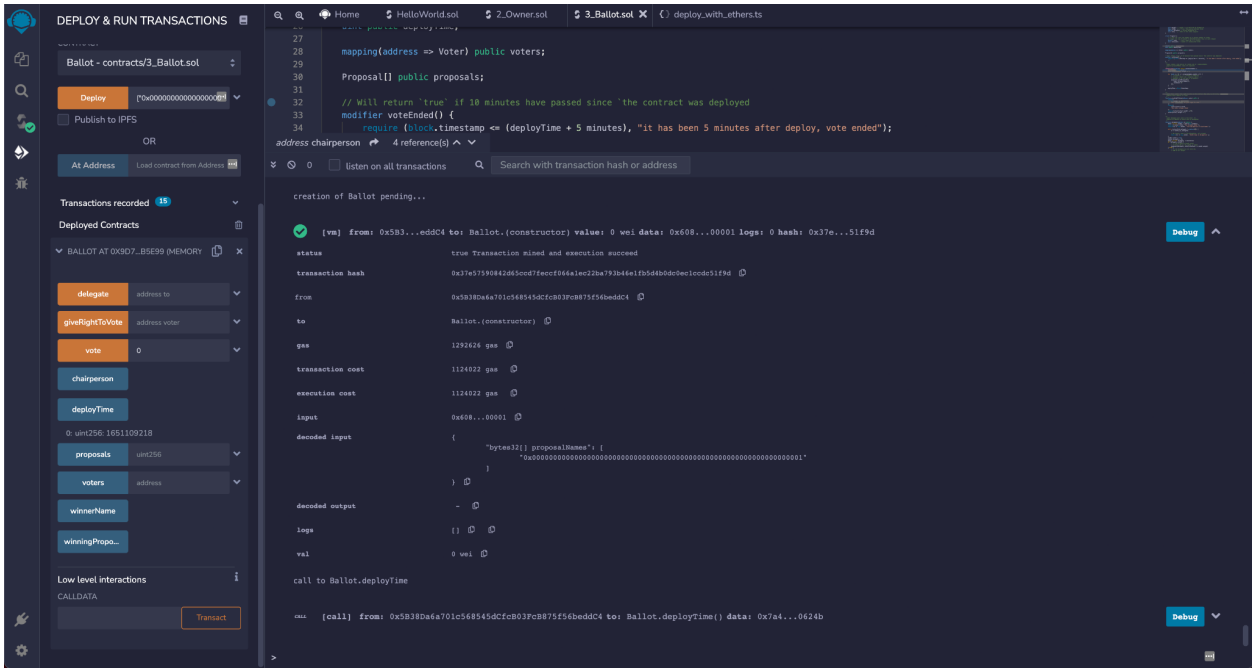
This is a screenshot of the deployed contract with a number being stored and received.



Ballot.sol with voteEnded modifier

Code can be found on Github here:
https://github.com/vezenovm/ZK-University/blob/main/background/Ballot.sol

Below is the deployment process of *Ballot.sol*. There is a deployTime variable set to block.timestamp inside the constructor of the contract. It is a public variable so you can see the value in the screenshot below.



Below is a revert showing that the `voteEnded` modifier was hit. You can see the reason for the revert in the screenshot as below: “it has been 5 minutes after deploy, vote ended”.