A. Conceptual Knowledge

1. **What is a smart contract? How are they deployed? You should be able to describe how a smart contract is deployed and the necessary steps.**

My Answer:
A smart contract is just like an application running on the mobile phone.The difference is that a smart contract is running on the blockchain(evm) without any user interface. We have to use it via some other tools rather than just tap it on the phone.
Deployed steps:
(1) Code the script on IDE (remix and so on).
(2) Compile the sol source file on remix.
(3) Select the network, wallet address and smart contract you want to deploy and input the parameters on remix, then click the deploy button.(make sure you have enough gas fee)
Certainly, you can also use truffle or solc to compile the smart contract and sign it into a raw transaction via ethereum sdk,then send it to the blockchain, which is a good way for development.

2. **What is gas? Why is gas optimization such a big focus when building smart contracts?**

My Answer:
Gas refers to the fee, being used to allocate resources of the evm. The more gas a transaction uses, the more fee you have to pay for executing it. So, gas optimization is important to save ethers(money).

3. **What is a hash? Why do people use hashing to hide information?**

My Answer:
Hash is a fixed-length output from which a hash function returns with the input of arbitrary length data. People use that to hide information because we can only encrypt the input data but can not decrypt the output data into what you input. And blockchain makes good use of this feature.

4. **How would you prove to a colorblind person that two different colored objects are actually of different colors? You could check out Avi Wigderson talk about a similar problem here.**

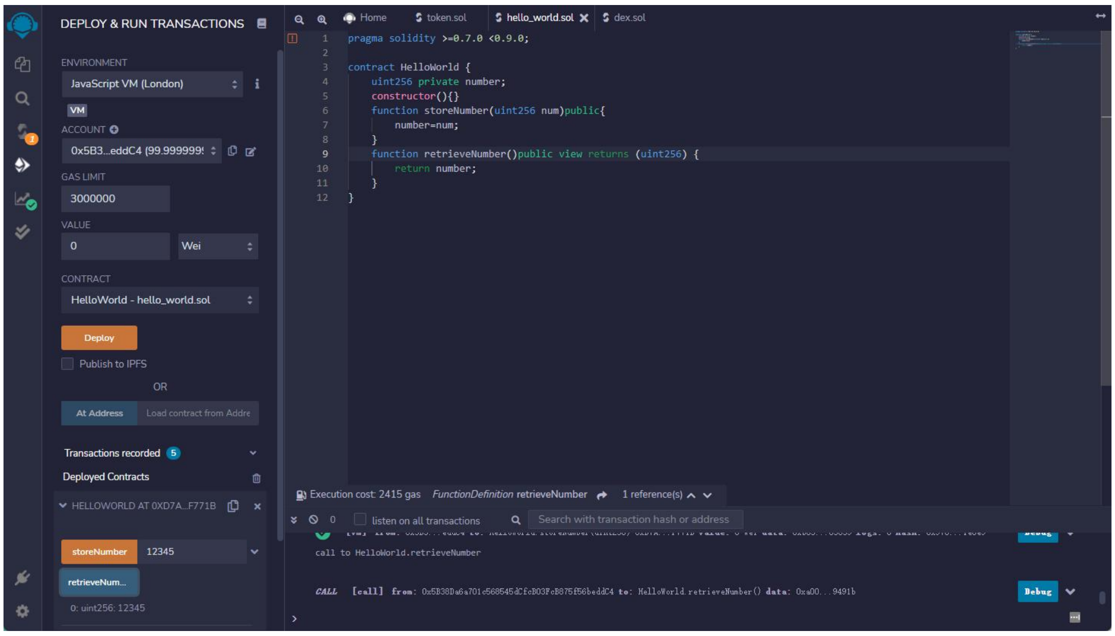
My Answer:
Steps:
(1) The colorblind person takes some photos of both objects, and these photos should not show any features of both objects, which can confuse him or her. The

colorblind person have to make sure that he or she knows these photos were taken of which object. For example, he or she can mark them but do not let me know that.
(2) The colorblind person gives me the photos and I will divide them into two parts according to their color.
(3) If both parts are from the same object which verified by the marks the colorblind person makes, I classify the photos successfully.
(4) The colorblind person can repeat the process for several times till he thinks it is enough. If I can classify the photos every time, then I prove that the two objects have different colors.

B. Solidity Test

1. Hello World smart contract:

Solidity file link:
https://github.com/MuserQuantity/zero-knowledge-proof-courses/blob/main/background-test/hello_world.sol
Git pull-request link:
https://github.com/MuserQuantity/zero-knowledge-proof-courses.git



ENVIRONMENT: JavaScript VM (London)
ACCOUNT: 0x5B3...eddC4 (99.999999
GAS LIMIT: 3000000
VALUE: 0 Wei
CONTRACT: HelloWorld - hello_world.sol
HELLOWORLD AT 0XD7A...F771B
storeNumber 12345
0: uint256: 12345

```
pragma solidity >=0.7.0 <0.9.0;

contract HelloWorld {
    uint256 private number;
    constructor(){}
    function storeNumber(uint256 num)public{
        number=num;
    }
    function retrieveNumber()public view returns (uint256) {
        return number;
    }
}
```

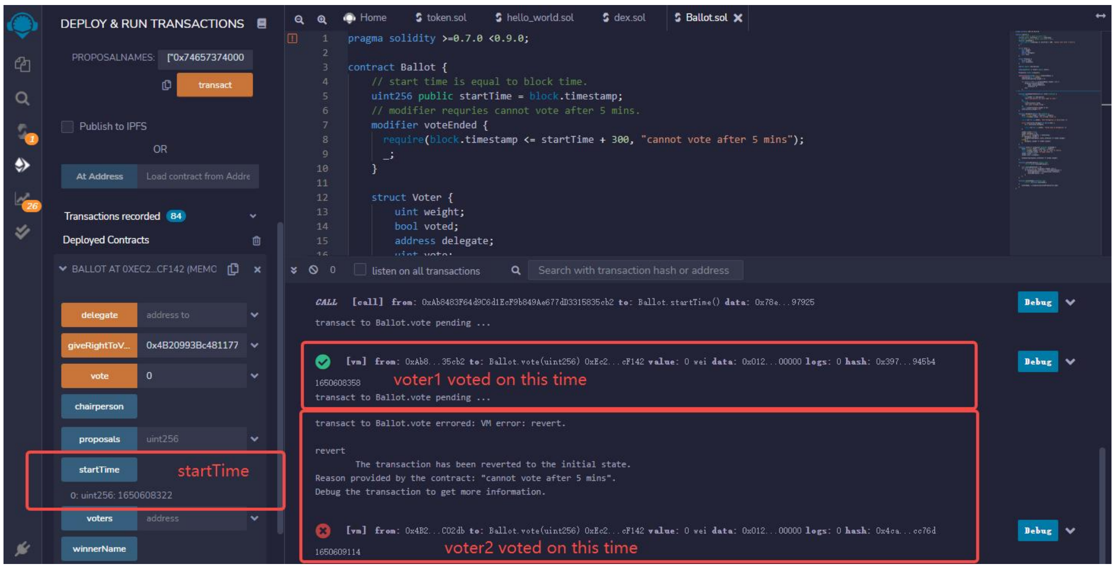
```
Execution cost: 2415 gas   FunctionDefinition retrieveNumber   1 reference(s)

call to HelloWorld.retrieveNumber

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

2. Ballot.sol

Solidity file link:
https://github.com/MuserQuantity/zero-knowledge-proof-courses/blob/main/background-test/Ballot.sol
Git pull-request link:(the same as above)
https://github.com/MuserQuantity/zero-knowledge-proof-courses.git



```
pragma solidity >=0.7.0 <0.9.0;

contract Ballot {
    // start time is equal to block time.
    uint256 public startTime = block.timestamp;
    // modifier requries cannot vote after 5 mins.
    modifier voteEnded {
      require(block.timestamp <= startTime + 300, "cannot vote after 5 mins");
      _;
    }

    struct Voter {
        uint weight;
        bool voted;
        address delegate;
```

```
CALL  [call] from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 to: Ballot.startTime() data: 0x78e...97925
transact to Ballot.vote pending ...

[vm] from: 0xAb8...35cb2 to: Ballot.vote(uint256) 0xEc2...cF142 value: 0 wei data: 0x012...00000 logs: 0 hash: 0x397...945b4
1650608358
transact to Ballot.vote pending ...
transact to Ballot.vote errored: VM error: revert.

revert
        The transaction has been reverted to the initial state.
Reason provided by the contract: "cannot vote after 5 mins".
Debug the transaction to get more information.

[vm] from: 0x4B2...C02db to: Ballot.vote(uint256) 0xEc2...cF142 value: 0 wei data: 0x012...00000 logs: 0 hash: 0x4ca...cc76d
1650609114
```