BBS: Workshop #2

Building a Lottery smart contract:

Introduction: A lottery is a game of chance, where participants can enter the lottery by sending a specific amount of cryptocurrency to the contract address, and the winner is chosen based on a random number generated by the contract. Once the winner is determined, the contract automatically distributes the prize to the winner's account.

In this worksheet you will be:

- Setting up metamask
- Adding an Ethereum test network and getting some funds using a faucet
- Setting up remix IDE with the skeleton code provided
- Writing out the lottery smart contract
- Testing the smart contract using remix and MetaMask

Part 1: Setting up MetaMask (If you already have a MetaMask wallet skip to part 2)

1. Visit https://metamask.io/
2. Hit "Download" in the menu bar
3. Click "Install MetaMask for Chrome or Firefox". You will be directed to the Chrome Web Store.
4. Click "Add to Chrome"
5. On the pop up, click "Add extension"
6. Create a new wallet
7. Write down a good password
8. Secure your wallet and write down the **Secret Recovery Phrase**
9. Congrats! Now you have a MetaMask wallet.

Part 2: Adding Sepolia test network to MetaMask

We will be using a testnet with some fake funds to test our smart contract

10. Open MetaMask and go to settings
11. Enable **Show test networks**
12. Then under the networks menu on the top right select **Sepolia test network**
13. Copy your wallet address and paste it in this faucet to get some funds

Part 3: Remix IDE

Now that you have some funds in your wallet to test your smart contracts, we can start setting up the development environment.

14. Open Remix IDE at https://remix.ethereum.org/
15. Upload the skeleton code that we have provided (https://github.com/Bristol-Blockchain-Society/lottery-workshop/blob/main/Lottery.sol) to Remix
16. Take a few minutes to familiarise yourself with solidity by looking at the documentation and solidity by example. Throughout this worksheet, refer back to these resources if you don't know how to implement something.

Part 4: Development

Now let us start building our contract! Complete, according to the comments in the skeleton code, the: (*Note: Check the references above for the syntax and examples).*

17. necessary variables and data structures (above the constructor).
18. `constructor` function. Takes durationHours as parameter
19. `enterLottery` function. *Note: Make sure to complete all the* require *statements as well.*
20. `openLottery` function.
21. `closeLottery` function.
22. `selectWinner` function.
23. `winningProposal` function.
24. `GetPlayers` function.
25. `GetWinner` function.

Part 5: Deployment and Testing

Now it is time to test our smart contract on the TestNet!

26. Compile the contract on Remix.
27. Deploy the contract on Remix selecting **Injected Provider – MetaMask** in environment.
28. Select a duration in hours and deploy the contract.
29. In MetaMask you can create multiple wallets under the same account and transfer some funds to the other wallets created.
30. Run the openLottery function from remix to open the lottery.
31. Using the wallets created, send atleady the minimum bet of eth to the contract's address which will call the enterLottery function.
32. Make sure the transaction was successful and call the getPlayers function to check the players in the lottery.
33. Do the same thing for 1 or 2 more wallets.
34. Other people can enter your lottery as well by sending eth to your contract's address. Pair up with someone and send some eth to their contract's address
35. Finally, call the selectWinner function and watch as the total balance of the contract goes to 0
36. Call the getWinner function to see which address has won the lottery.