FT5004 Lab 2

Preparing the environment:

1. Open the sample Dice.sol using remix: https://bit.ly/Dice_sol

2. Open 639….32e > Dice.sol

3. Enable the "Solidity" environment

Alternatively, check out links to the code in the appendix.

Create a file called 'Dice.sol' and copy over the other Dice contract's skeleton. We'll be using this for the problems in the lab.

Problem 1: (Revise Coding)

- Extend the ***Dice*** contract to add a new property: ***luckyTimes*** which is incremented *every time the max number is rolled.*
- Add a getter function to read this property (Basically return luckyTimes value)
- Add an event called **luckytimesEvent** for the times max number is rolled
- Add a function called **destroyDice** to destroy dice and return ether

Problem 2: (OOP with Smart Contracts)

- Implement a ***DiceMarket*** contract. It receives ownership of the dice, and enable the functions. A commission fee is set by the owner during creation of ***DiceMarket*** Contract. Implement the following methods:
- ***list(uint256 id, uint256 price)*** – list a dice for sale. Price needs to be >= value + comission fee
    - o *First, transfer the dice to the DiceMarket contract's address.*
    - o *Then, you should be able to list the dice in this market*
- ***unlist(uint256 id)*** – unlist dice from the market
    - o *Upon unlisting do not transfer the dice back to their owners.*
    - o *Simply delist them from the market, ie nobody should be able to buy the die.*
- ***checkPrice(uint256 id)*** – get price of dice
- ***buy(uint256 id)*** – Buy the dice at the requested price

    - If you want to implement an airtight solution, you should return any extra money to the msg.sender.

- Note: please set appropriate modifier to check for condition before allowing the execution of certain functions.

Problem 3: (ERC20 Standard)

- Issue a ERC-20 token, **DT** (DiceToken), such that
- It complies with ERC-20 Interface
- The total supply is 10,000 token
- Anyone can top up DT, with the price of 0.01 Eth per DT
- When the supply is not enough (e.g., someone wants to top up 200DT, but there is only 100DT left in supply), return with error message “DT supply is not enough”.
- Hint: We’ll be using the ERC20 contract accessible in the appendix

Lab exercises:

Exercise 1:

- Extend the **Dice smart contract** and implement another contract called **DiceBattle**. **DiceBattle** allows the uses to roll 2 dice by supplying the diceId. The ownership of the **Dice** is transferred to the winner of the **DiceBattle.**
    - **See the skeleton for a detailed breakdown of the contract’s purpose.**

Exercise 2:

- Modify problem 2 to use DT instead.
    - Perform the same functionalities as problem 2 but instead of using ether, use DT as payment method (for both commission and trade).
    - HINT: We created DT in the lab

Submission: Please submit a zip that contains 5 subfolders, corresponding to the 5 questions. For each folder, put all the necessary .sol files for that question in it.

Appendix

ERC20 Contract Link:

https://github.com/VibhuKrovvidi/IS4302_Instruction/blob/main/ERC20.sol

HW Skeleton Link:
https://github.com/VibhuKrovvidi/IS4302_Instruction/blob/main/DiceBattleSkeleton.sol