Solidity Exam

Instructions

- In this exam, you will add code to a smart contract and test file to satisfy a series of assert checks within the `test_exam.py` file.
- The smart contract is contained in `ExtractReward.sol`, while the test file is contained in `test_exam.py`. Both of these files should be submitted, along with a screenshot of your console output when you run: `brownie test tests/test_exam.py`..
- In the github you will find 4 contracts: `ExtractReward.sol, GoodToken.sol, reward.sol and truster.sol`. You are not allowed to change the 3 later ones.
- Full marks in each exercise will be rewarded for code satisfying the assertions and the instructions of each exercise.
- All code should be commented.
- If an assertion is not satisfied, a percentage of points will be awarded for attempts, comments, and any pseudocode provided in comments.

Contracts overview

- `ExtractReward.sol`: In this contract, you will write the solutions of the problems. This contract contains the necessary imports for interacting with the other ones.
- `GoodToken.sol`: Is just a normal ERC20 contract
- `reward.sol`: The reward contract consists of a contract that will contain some ETH. This contract contains a function that allows Holders of the ERC20 contract are allowed to extract an Ether.
- `truster.sol`: This is a Flashloan pool that provides Tokens to players.

Exercises

In these exercises, you should only use the account `you` and deploy contracts on its behalf, unless stated otherwise (exercise 1).

1. (2 points) In the `test_exam.py` file, make a transaction that transfers $100 \cdot 10^{18}$ tokens from `deployer` account to `you` account. The `you` account will use these later.
2. (2 points) Write a function `thankYou()` in the `ExtractReward` contract. `thankYou()` should take as input an address `receiver`. The function sends to `receiver` the balance of ether in the smart contract. Your function `thankYou()` should require:

- The address calling the function is the person who deployed the contract. This should be done with a modifier.
- The ether balance of the contract is greater than 0.

3. (1 point) In `test.py`, deploy the `ExtractReward` contract. Send 1 ETH to the deployed `ExtractReward` contract.

4. (1 points) Write code in the test file with `you` account calling the `thankYou()` function, inputting the address of the `deployer` account.

5. (1 points) Now that the `you` account holds some amount of tokens, claim one ether from the `Rewarder` contract in the `reward.sol` file.

6. (2 points) Assuming that the `you` account holds some amount of tokens, write a function `extractAllEth()` in the `ExtractReward` smart contract that extracts all the ether from the `Rewarder` contract in one transaction. Your function should try to call the `reward()` function in the `Rewarder` contract at least once.

7. (2 points) **[Bonus]** Write a function `extractAllEth2()` in the `ExtractReward` contract to extract all the ethers from the deployed `Rewarder` contract. In this exercise you should:
   (a) Deploy the `ExtractReward` contract from the `you2` account.
   (b) **Not** call the `extractAllEth2()` function directly from the test file.
   (c) **Not** send any tokens to the deployed `ExtractReward` contract.