ERC20 Exercise 2

Intro

A receipt token is an ERC20 token that is issued to the depositor upon depositing assets in a crypto lending and borrowing platform.

For example, if 1 ETH is deposited in Aave, 1 aWETH (wrapped ETH) receipt token is returned. Aave uses the "a" prefix while Compound receipt tokens use a "c." One USDT sent to Compound returns a cUSDT token.

Your goal in this exercise is to create a smart contract which allow deposits of the following ERC20 tokens: AAVE UNI WETH

And mints to the depositor a receipt token with the "r" prefix for the tokens that was deposited. For example: if the user ddeposits `AAVE`, he will get back `rAave`.

When the depositor want to claim back it's deposited tokens, he can send the receipt token back to the contract, the receipt tokens will be burned and the deposited tokens will be returned to the depositor.

Note: This exercise is executed on an Ethereum mainnet Fork block number `15969633`. Everything is already configured in the `hardhat.config.js` file

Tasks

Create your contract under the `./contracts/erc20-2` folder:

Task 1

1. Complete the ERC20 receipt tokens contract (`rToken.sol`) (inherit from OpenZepplin `ERC20.sol`)
2. The contracts should support minting and burning of tokens
3. Only the contract owner can mint and burn tokens (onlyOwner protected)
4. The contract should receive in the constructor `address _underlyingToken` and store it as a public storage variable

Task2

Complete the `TokensDepository.sol` Smart contract, which allows users to:

1. Deposit supported assets and receive receipt tokens
2. Withdraw deposited assets by providing the receipt tokens

- Upon contract deployment, the contract should deploy the receipt tokens contracts as well
- The contract should be the owner of the receipt tokens contracts
- The contract should burn the receipt token upon withdrawal

Task3

Complete the tests in the `tests.js` file.