Smart Contracts Exercise 03: ERC-20 CTU Token – Solution

The example implementation of the CTU Token contract can be found in this GitLab repository. The required implementation is in the file `contracts/CTUToken.sol`. Even this implementation cannot prevent a frontrunning attack using only the `approve()` function. One possible solution is to use the `increaseAllowance()` and `decreaseAllowance()` functions instead of the `approve()` function. It's important to note, that even this solution is not completely foolproof, as the `decreaseAllowance()` function can still be frontrun before the allowance value is reduced. Frontrunning is part of the complex topic of MEV (Maximal Extractable Value), which is beyond the scope of this exercise and will be covered in future exercises.

```
function increaseAllowance(
    address spender,
    uint256 addedValue
) public returns (bool success) {
    // Check if the spender is not the zero address
    require(spender != address(0),  IncreaseAllowanceForZeroAddress());
    // Increase the allowance
    allowances[msg.sender][spender] += addedValue;
    // Emit Approval event
    emit Approval(msg.sender, spender, allowances[msg.sender][spender]);
    // Return true if the operation is successful
    return true;
}
```

```
function decreaseAllowance(
    address spender,
    uint256 subtractedValue
) public returns (bool success) {
    // Check if the spender is not the zero address
    require(spender != address(0), DecreaseAllowanceForZeroAddress());
    // Check if the current allowance is sufficient
    require(
        allowances[msg.sender][spender] >= subtractedValue,
        "Decreased allowance below zero"
    );
    // Decrease the allowance
    allowances[msg.sender][spender] -= subtractedValue;
    // Emit Approval event
    emit Approval(msg.sender, spender, allowances[msg.sender][spender]);
    // Return true if the operation is successful
    return true;
}
```

We recommend using the OpenZeppelin ERC20 token implementation, which you can find in the file `contracts/CTUTokenOpenZeppelin.sol`. Take time to explore and

understand the implementation in the OpenZeppelin GitHub repository. Note: OpenZeppelin has removed the increaseAllowance and decreaseAllowance functions from their implementation. You can find their explanation for this decision here.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

// Import OpenZeppelin's ERC20 implementation
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

/**
 * @title CTUToken
 * @dev A custom implementation of an ERC-20 Token using OpenZeppelin's library.
 */
contract CTUToken is ERC20 {
    // Define the initial supply: 1,000,000 tokens with 18 decimal places
    uint256 private constant INITIAL_SUPPLY = 1_000_000 * 10 ** 18;

    /**
     * @dev Constructor that initializes the ERC-20 token with a name and symbol,
     * and mints the total supply to the deployer's address.
     */
    constructor() ERC20("CTU Token", "CTU") {
        // Mint the initial supply to the deployer of the contract
        _mint(msg.sender, INITIAL_SUPPLY);
    }
}
```