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.

```
function increaseAllowance(
   address spender,
   uint256 addedValue
) public returns (bool success) {
   // Check if the spender is not the zero address
   require(spender != address(0), "Increase allowance for zero address");
   // 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), "Decrease allowance for zero address");
   // 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;
}
```

The recommended approach is to use the following ERC20 token implementation from OpenZeppelin. You can find it in the file `contracts/CTUTokenOpenZeppelin.sol`. Go through OpenZeppelin GitHub repository and understand it.

```
// 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);
    }
}
```