1. **Token Expiry Implementation:** How would you implement an expiry date feature for ERC-20 tokens within a smart contract?

Ans:

// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.20;

interface IERC20 {

function totalSupply() external view returns (uint256);

function balanceOf(address account) external view returns (uint256);

function transfer(address recipient, uint256 amount) external returns (bool);

function allowance(address owner, address spender) external view returns (uint256);

function approve(address spender, uint256 amount) external returns (bool);

function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);

event Transfer(address indexed from, address indexed to, uint256 value);

event Approval(address indexed owner, address indexed spender, uint256 value);

}

contract Block is IERC20{

string public name="Shihab"; //name of the token

string public symbol="SSU"; //symbol of the token

uint public decimals=0;

address public founder;//initially this will have the total supply

uint256 public expiryDate;//Implementing expiery time feature

mapping(address=>uint) public balances; //information of balance of each address

uint public override totalSupply;

mapping(address=>mapping(address=>uint)) allowed;

constructor(uint256 \_expiryDate){

totalSupply=1000;

founder=msg.sender;

balances[founder]=totalSupply;

expiryDate = \_expiryDate;

}

//balance of token of an account

function balanceOf(address account) external view returns (uint256){

return balances[account];

}

function transfer(address recipient, uint256 amount) external returns (bool){

require(amount>0,"amount must be greater than zero");

require(balances[msg.sender]>=amount,"Balance must be greater than zero");

balances[msg.sender]-=amount;//balances[msg.sender]=balances[msg.sender]-amount

balances[recipient]+=amount;

emit Transfer(msg.sender, recipient, amount);

return true;

}

//this function determines how many tokens the the owner has allowed to spend this address

//spender = 0xabc owner= 0xdef 10Tokens //passbook - who has given whom and how many?

function allowance(address owner, address spender) external view returns (uint256){

return allowed[owner][spender];

}

//here we are giving the approval of the "allowed 100" tokens

//msg.sender has given the spender this amount

//this is like signing check

//10:am - 100 tokens

function approve(address spender, uint256 amount) external returns (bool){

require(block.timestamp < expiryDate, "Token has expired");

require(amount>0,"amount must be greater than zero");

require(balances[msg.sender]>=amount,"Balance must be greater than zero");

allowed[msg.sender][spender]=amount;

emit Approval(msg.sender, spender, amount);

return true;

}

//in order to cash out the check

//12:00pm

function transferFrom(address sender, address recipient, uint256 amount) external override returns (bool){

//I want to cash out the token. so receipient is me

//checking if the sender who has allowd me 100 tokens to spend of his is it really allowed or not

//i.e is the check really signed for this amount or not

require(allowed[sender][recipient]>=amount,"Recipient don't have authority to spend sender's token");

require(block.timestamp < expiryDate, "Token has expired");

//the tkens have allowed at 10 pm. but at 12 pm does he have the sufficient tokens or not

require(balances[sender]>=amount,"Insufficient balance");

balances[sender]-=amount;

balances[recipient]+=amount;

emit Transfer(msg.sender, recipient, amount);

return true;

}

// Function to update the expiry date (only callable by the owner)

function updateExpiryDate(uint256 newExpiryDate) external {

expiryDate = newExpiryDate;

}

}