Centurion University
Shaping Lives... Empowering Communities...

School: ..... Campus: .....

Academic Year: ..... Subject Name: ..... Subject Code: .....

Semester: ..... Program: ..... Branch: ..... Specialization: .....

Date: .....

Applied and Action Learning

(Learning by Doing and Discovery)

Name of the Experiement :

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

ERC-20 Token Standard (In Other Words)

In Ethereum, tokens can represent almost anything — such as:

- Reputation points on a website or app
- Game character abilities or scores
- Shares or financial assets of a company
- Stablecoins like USD or other fiat currencies
- Commodities such as gold, and many more

To make all these token types work smoothly and consistently, Ethereum uses a rule set called the ERC-20 standard.

This standard ensures that all tokens built using it follow the same structure and can easily interact with other wallets, smart contracts, and decentralized applications (DApps). It also helps developers add extra features beyond what Ether (ETH) provides.

About ERC-20

- ERC-20 stands for Ethereum Request for Comments 20, proposed by Fabian Vogelsteller in November 2015.
- It defines a standard interface for fungible tokens, meaning all tokens are identical in type and value — just like how every 1 ETH equals another 1 ETH.
- An ERC-20 token behaves like digital money where each unit is interchangeable with another.

Prerequisites

Before working with ERC-20, you should understand:

- Accounts – for holding tokens
- Smart Contracts – for programming token logic
- Token Standards – to know how tokens interact with the blockchain

Main Functions of ERC-20 Tokens

ERC-20 defines a common set of methods and events that all compliant tokens must have.

These include:

- Transferring tokens between accounts
- Checking the balance of any account
- Viewing the total supply of tokens in existence
- Approving and allowing another account (like a DApp or exchange) to spend tokens on your behalf

When a smart contract includes these ERC-20 functions and events, it becomes an ERC-20 Token Contract.

Once deployed, the contract manages and records all token transactions across the Ethereum network.

*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

Coding Phase: Pseudo Code / Flow Chart / Algorithm

Function	Purpose (Simplified)
name()	Returns the token's name (for example, "MyToken").
symbol()	Returns the token's short symbol (like "MTK").
decimals()	Tells how divisible the token is — for example, 18 decimals means you can send 0.000000000000000001 tokens.
totalSupply()	Shows the total number of tokens that exist.
balanceOf(address _owner)	Returns how many tokens a specific wallet address owns.
transfer(address _to, uint256 _value)	Sends a chosen amount of tokens from your account to another address.
transferFrom(address _from, address _to, uint256 _value)	Lets a third party (like a DApp) transfer tokens from one address to another, if they have permission.
approve(address _spender, uint256 _value)	Gives permission to another address (like an exchange or contract) to spend tokens on your behalf.
allowance(address _owner, address _spender)	Shows how many tokens an approved address is still allowed to spend from your account.

🔔 ERC-20 Events (In Other Words)

Events help track token movements and approvals on the blockchain.

Event	Meaning (Simplified)
Transfer(address indexed _from, address indexed _to, uint256 _value)	Triggered whenever tokens move from one account to another.
Approval(address indexed _owner, address indexed _spender, uint256 _value)	Triggered when an owner allows another address to spend a certain number of tokens.

* Softwares used

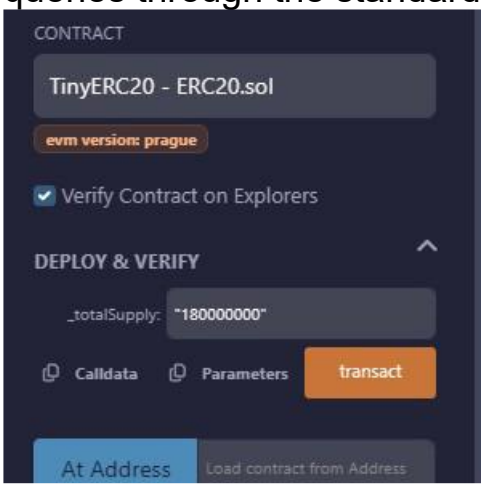
- Solidity
- Solc compiler
- JSON
- ERC20 Token Standard
- Remix IDE
- Brave Browser

* Implementation Phase: Final Output (no error)

This contract implements the core ERC-20 methods and events necessary for the token to operate correctly.

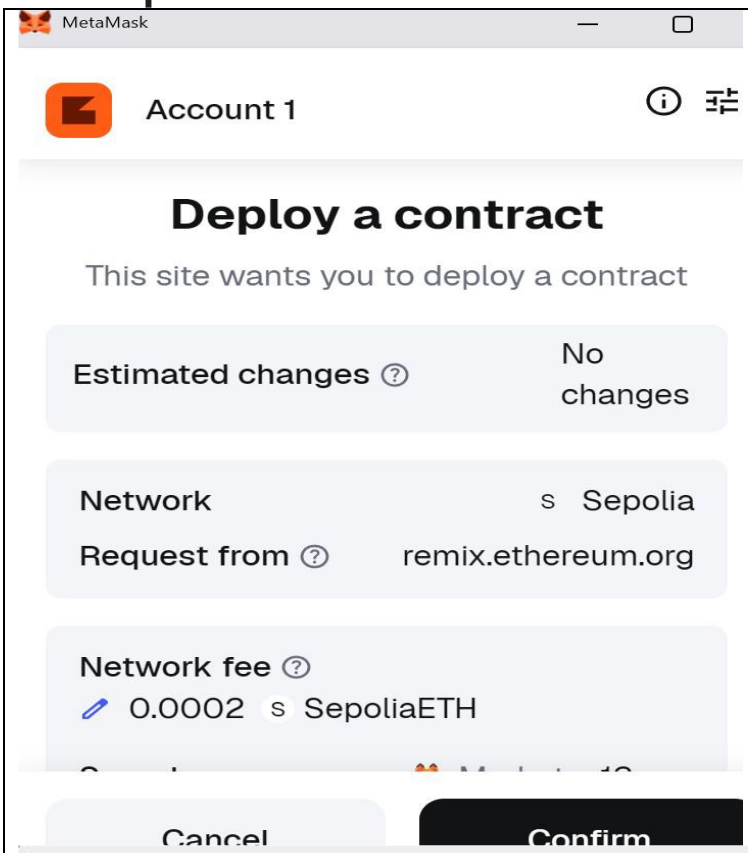
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract TinyERC20 {
    string public constant name = "Tiny Token";
    string public constant symbol = "TINY";
    uint8 public constant decimals = 18;

    uint256 public totalSupply;
    mapping(address => uint256) public balanceOf;
    mapping(address => mapping(address => uint256)) public allowance;

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

    constructor(uint256 _totalSupply) {   infinite gas 500400 gas
        totalSupply = _totalSupply;
        balanceOf[msg.sender] = _totalSupply;
        emit Transfer(address(0), msg.sender, _totalSupply);
    }

    function transfer(address to, uint256 value) external returns (bool) {   infinite gas
        balanceOf[msg.sender] -= value;
        balanceOf[to] += value;
        emit Transfer(msg.sender, to, value);
        return true;
    }

    function approve(address spender, uint256 value) external returns (bool) {   infinite gas
        allowance[msg.sender][spender] = value;
        emit Approval(msg.sender, spender, value);
        return true;
    }

    function transferFrom(address from, address to, uint256 value) external returns (bool) {   infinite gas
        allowance[from][msg.sender] -= value;
        balanceOf[from] -= value;
        balanceOf[to] += value;
        emit Transfer(from, to, value);
        return true;
    }
}
```

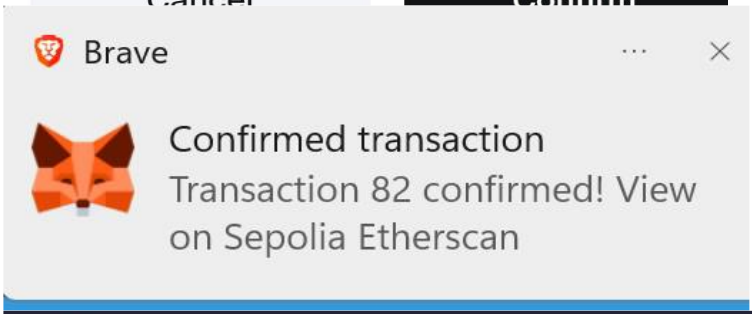
With a deployment supply of 1,000,000 * 10^{18} units, the token is automatically accessible to ERC-20-compliant platforms, allowing transactions and balance queries through the standard ABI using its contract address.



*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

Implementation Phase: Final Output (no error)

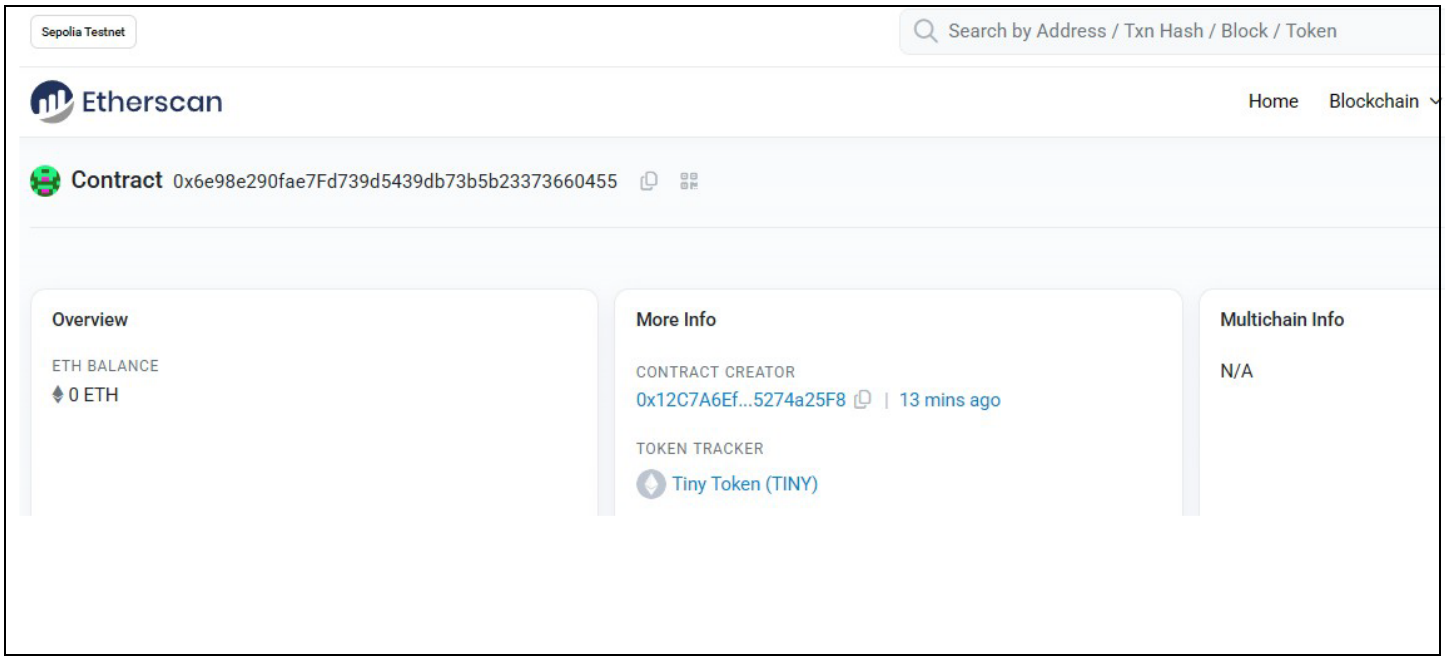





```
[block:9541358 txIndex:6] from: 0x12c...a25f8 to: TinyERC20.(constructor) value: 0 wei data: 0x608...a9500 logs: 1
hash: 0x6c1...c8f96
Verification process started...
Verifying with Sourcify...
Verifying with Routescan...
Etherscan verification skipped: API key not found in global Settings.
Sourcify verification successful.
https://repo.sourcify.dev/11155111/0x6e98e290fae7Fd739d5439db73b5b23373660455/
Routescan verification successful.
https://testnet.routescan.io/address/0x6e98e290fae7Fd739d5439db73b5b23373660455/contract/11155111/code
```


*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Implementation Phase: Final Output (no error)



* Observations

* A fixed total supply is set — a common approach used by most tokens.
* The token uses 18 decimal places, which is the standard for ERC-20 tokens.
* MetaMask can be used to view token balances and make transfers.
* Uniswap enables trading and liquidity provision for the token.
* Etherscan allows users to track token transactions and contract details.
* Supported on all decentralized and centralized exchanges (DEXs & CEXs) that handle ERC-20 tokens.
* Compatible with all major crypto wallets like Trust Wallet, Coinbase Wallet, and others.

ASSESMENT

Rubrics	Full Mark	Marks Obtained	Remarks
Concept	10		
Planning and Execution/ Practical Simulation/ Programming	10		
Result and Interpretation	10		
Record of Applied and Action Learning	10		
Viva	10		
Total	**50**		

Signature of the Student:

Name :

Regn. No. :

Signature of the Faculty:

**As applicable according to the experiment.*
Two sheets per experiment (10-20) to be used.