Experiment No-5

Title: Creation and Deployment of an ERC-20 Token on Ethereum Test net

Aim: To create and deploy an ERC-20 token smart contract on the Ethereum Test net using Remix IDE and Meta Mask.

Theory:

- **Ethereum** is a blockchain that supports smart contracts.
- **Smart Contract** is a program that runs on the blockchain without manual control.
- **ERC-20 Token** is a standard for making fungible tokens (all tokens are equal in value, like currency).
- **Testnet (Sepolia/Goerli)** is a practice version of Ethereum used for testing without spending real money.
- **Remix IDE** is an online tool to write and deploy Solidity smart contracts.
- **MetaMask** is a crypto wallet that connects to Ethereum networks.

Key Characteristics of ERC-20 Tokens:

- **Fungible** – Every token has the same value.
- **Transferable** – Tokens can be sent between wallets.
- **Interoperable** – Works in all Ethereum wallets and apps.
- **Supply** – Total number of tokens is fixed at deployment (or can be minted later).
- **Standard Functions** –
  - `totalSupply()` → shows total tokens
  - `balanceOf(address)` → shows balance of a wallet
  - `transfer(address, amount)` → send tokens
  - `approve(address, amount)` → allow someone else to spend tokens
  - `transferFrom(from, to, amount)` → send tokens on behalf of another user
  - `allowance(owner, spender)` → check approved tokens

Steps of Execution:

1. Install and set up **MetaMask** wallet.
2. Connect wallet to **Ethereum Testnet (Sepolia/Goerli)**.
3. Get free **test ETH** from a faucet.
4. Open **Remix IDE** and create a new Solidity file.
5. Write the ERC-20 smart contract code.
6. Compile the smart contract.
7. Deploy it using MetaMask on Testnet.

8. Check contract on **Etherscan (Testnet)**.
9. Test token functions (transfer, balance check, etc.).

Stepwise Procedure :

1. **Set up Wallet**
    - Install MetaMask extension.
    - Create a wallet and save the recovery phrase.
    - Switch to Sepolia/Goerli test network.
    - Get test ETH from a faucet.
2. **Open Remix IDE**
    - Go to https://remix.ethereum.org.
    - Create a new file `MyToken.sol`.
3. **Write Smart Contract**
    - Copy the ERC-20 token code into the file.
4. **Compile Contract**
    - Open **Solidity Compiler** tab.
    - Select version (e.g., 0.8.20).
    - Click **Compile**.
5. **Deploy Contract**
    - Open **Deploy & Run Transactions**.
    - Select **Injected Web3** → Connect MetaMask.
    - Deploy the contract with an initial token supply.
    - Confirm in MetaMask.
6. **Verify Contract**
    - Copy contract address from Remix.
    - Search it on **Etherscan Testnet**.
7. **Interact with Token**
    - Use deployed functions like `balanceOf()` and `transfer()`.
    - Add token in MetaMask by pasting the contract address.

Program (ERC-20 Token Code in Solidity) :

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;
interface IERC20 {
    function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns (uint256);
    function transfer(address to, uint256 value) external returns (bool);
    function allowance(address owner, address spender) external view returns (uint256);
    function approve(address spender, uint256 value) external returns (bool);
    function transferFrom(address from, address to, uint256 value) external returns (bool);
    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);
}
contract MyToken is IERC20 {
    string public name = "MyToken";
    string public symbol = "MTK";
    uint8 public decimals = 18;
    uint256 private _totalSupply;
    mapping(address => uint256) private balances;
    mapping(address => mapping(address => uint256)) private allowed;
    constructor(uint256 initialSupply) {
        _totalSupply = initialSupply * 10 ** uint256(decimals);
        balances[msg.sender] = _totalSupply;
        emit Transfer(address(0), msg.sender, _totalSupply);
    }
    function totalSupply() public view override returns (uint256) {
        return _totalSupply;
    }
    function balanceOf(address account) public view override returns (uint256) {
        return balances[account];
    }
    function transfer(address to, uint256 value) public override returns (bool) {
        require(balances[msg.sender] >= value, "Not enough balance");
        balances[msg.sender] -= value;
        balances[to] += value;
        emit Transfer(msg.sender, to, value);
        return true;
    }
    function approve(address spender, uint256 value) public override returns (bool) {
        allowed[msg.sender][spender] = value;
        emit Approval(msg.sender, spender, value);
        return true;
    }
    function allowance(address owner, address spender) public view override returns (uint256)
{
        return allowed[owner][spender];
    }
    function transferFrom(address from, address to, uint256 value) public override returns
(bool) {
        require(balances[from] >= value, "Not enough balance");
        require(allowed[from][msg.sender] >= value, "Allowance exceeded");
        balances[from] -= value;
        balances[to] += value;
        allowed[from][msg.sender] -= value;
        emit Transfer(from, to, value);
        return true;
    }
}
```

Key Points :

- ERC-20 is the most common token standard on Ethereum.
- Always test on **Testnet** before deploying on Mainnet.
- MetaMask + Remix makes it easy to deploy contracts.
- Tokens can be viewed and transferred in MetaMask after deployment.

Conclusion :

In this experiment, we created and deployed an ERC-20 token smart contract on the Ethereum Testnet. We used Solidity to write the contract, Remix IDE to deploy it, and MetaMask for transactions. This shows how cryptocurrencies and blockchain tokens are built and managed.

Viva Questions:

1. What is an ERC-20 token?
2. Why do we use Ethereum Testnet?
3. What is the role of MetaMask in this experiment?
4. Explain the function of transfer() in ERC-20.
5. What is the use of the approve() function?
6. How can you check your token balance?
7. What is the difference between Mainnet and Testnet?