



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 Experiment :

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

Pseudo Code:

```
START Token Creation
IMPORT ERC20 standards from OpenZeppelin
DEFINE contract MyToken inheriting from ERC20
CONSTRUCTOR(initialSupply):
    SET token name: "MyToken"
    SET token symbol: "MTK"
    MINT initialSupply to contract deployer
END CONSTRUCTOR
END CONTRACT
```

Algorithm:

1. Import necessary ERC20 template
2. Initialize token with name and symbol
3. Mint initial supply to deployer
4. Deploy to blockchain network
5. Verify deployment and functions

Flow Chart:

```
[Start] → [Import OpenZeppelin ERC20] → [Create MyToken Contract]
→ [Define Constructor] → [Set Name/Symbol] → [Mint Tokens]
→ [Deploy to Network] → [Test Functions] → [End]
```

Coding Phase: Pseudo Code / Flow Chart / Algorithm

* Softwares used

- Remix IDE: Online Solidity development environment
- MetaMask: Crypto wallet for network interaction
- OpenZeppelin: Library for secure smart contract templates
- Sepolia Testnet: Ethereum test network for deployment
- Web Browser: Brave browser for accessing Remix

* **Testing Phase: Compilation of Code (error detection)**

Code Testing Steps:

solidity

Test 1: Compilation Check

No syntax errors

OpenZeppelin imports correctly

Test 2: Contract Deployment

Constructor accepts initialSupply parameter

Token deploys successfully to Remix VM

Test 3: Function Testing

name() returns "MyToken"

symbol() returns "MTK"

balanceOf() shows correct initial supply

decimals() returns 18 (standard)

Errors Detected & Fixed:

1. Compiler Version Mismatch - Fixed by matching pragma version
2. Import Path Errors - Corrected OpenZeppelin import syntax
3. Constructor Parameters - Ensured initialSupply is passed correctly
4. Network Connection - Resolved MetaMask-Sepolia connection issues

* Implementation Phase: Final Output (no error)

The screenshot displays a Solidity IDE interface. At the top, a 'Compile' button is visible. Below it, the code for 'Indiana.sol' is shown, featuring a constructor and a mint function. The deployment status is confirmed as 'Contract deployment -0 SepoliaETH -0 SepoliaETH' with a 'Confirmed' label. The Etherscan interface shows the contract address '0x8ABCe56534cAEDea4C3F6469a73E0C03b7560101' and the contract creator '0x12C7A6Ef...5274a25F8'.

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
5
6 contract Indiana is ERC20 {
7     constructor() ERC20("Indiana", "INDI") {
8         _mint(msg.sender, 1000000*10 ** decimals());
9     }
10 }

```

Account 1
0x12C7A...a25F8

Contract deployment
Confirmed

-0 SepoliaETH
-0 SepoliaETH

Sepolia Testnet

Etherscan

Contract 0x8ABCe56534cAEDea4C3F6469a73E0C03b7560101

Overview

ETH BALANCE
0 ETH

More Info

CONTRACT CREATOR
0x12C7A6Ef...5274a25F8 | 1 min ago

TOKEN TRACKER
Indiana (INDI)

The deployed Token is added to the MetaMask:

+ Import tokens

↻ Refresh list

Implementation Phase: Final Output (no error)

Import tokens

scams and security risks.

Select a network >

Token contract address

Next

Import tokens

Sepolia >

Token contract address

:AEDea4C3F6469a73E0C03b75601

Token symbol

INDI

Next

< Import tokens

Back

Import

✓ Token imported

You've successfully imported INDI.

CONTRACT DEPLOYED SUCCESSFULLY

- Contract Address: 0x8ABC...60101
- Network: Sepolia Testnet
- Token Name: Indiana
- Token Symbol: INDI
- Initial Supply: 100 INDI
- Status: VERIFIED - No Errors

Sepolia		
S	SepoliaETH	No conversion rate available 0.57038 SepoliaETH
B	BETK	No conversion rate available 999,959.99999 BETK
A	ABHI	No conversion rate available 200 ABHI
L	LUIS	No conversion rate available 999,550 LUIS
L	LUIS	No conversion rate available 999,960 LUIS
I	INDI	No conversion rate available 1,000,000 INDI

* **Observations**

Technical Observations:

- Gas Efficiency: ERC20 standard optimized gas usage
- Security: OpenZeppelin implementation prevents common vulnerabilities
- Interoperability: Standard ERC20 functions work with all wallets/exchanges
- Scalability: Contract can handle large number of token transfers

User Experience Observations:

- Remix IDE: Easy for beginners, instant compilation feedback
- MetaMask Integration: Smooth for testnet deployments
- Sepolia Network: Reliable testnet with good transaction speed
- Cost: Zero real money spent (test ETH only)

ASSESSMENT

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:

Page No.....

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