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 : Debugging Deep – Using Hardhat Console & Logs

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

ALGORITHM:

1.Start
    Set up the Hardhat development environment for local Ethereum testing.
2.Create a Smart Contract
    Write a Solidity contract that includes functions for testing and debugging purposes.
3.Import Hardhat Console Library
    Use import "hardhat/console.sol"; to enable console logging within Solidity code.
4.Add Console Logs
    Insert console.log() statements inside functions to track variable values, control flow, and logic execution.
5.Compile the Contract
    Run the compile command to ensure there are no syntax errors.
6.Deploy the Contract Locally
    Deploy it on a Hardhat local network using a script or console.
7.Use Hardhat Console for Debugging
    Launch Hardhat's interactive console to call functions and observe printed logs in real-time.
8.Check Console Outputs
    Analyze the logs displayed in the terminal to understand contract behavior and locate bugs.
9.Modify and Re-test
    Fix errors, recompile, redeploy, and re-test until the contract executes without issues.
10.End

* Softwares used

1.Node.js
2.Hardhat
3.Solidity Compiler
4.Visual Studio Code (VS Code)
5.Ethereum Local Network

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

* Implementation Phase: Final Output (no error)

Steps :

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

import "hardhat/console.sol";

contract Counter {
    uint256 private _count;

    constructor() {
        console.log("Deploying Counter with initial count:", _count);
    }

    function increment() public {
        console.log("Before increment, count was:", _count);
        _count += 1;
        console.log("After increment, count is:", _count);
    }

    function getCount() public view returns (uint256) {
        console.log("Current count is:", _count);
        return _count;
    }
}
```

```
PS D:\WEB 3\Test> npm install --save-dev hardhat

added 59 packages in 55s

16 packages are looking for funding
  run `npm fund` for details
PS D:\WEB 3\Test>
```

```
PS D:\WEB 3\Test> npm install --save-dev --legacy-peer-deps @nomicfoundation/hardhat-toolbox

added 1 package, and audited 61 packages in 2s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

```
PS D:\WEB 3\Test> npx hardhat test
Hardhat only supports ESM projects.

Please make sure you have `"type": "module"` in your package.json.

You can set it automatically by running:

npm pkg set type="module"
```

```
PS D:\WEB 3\fresh-test> npm install --save-dev @nomiclabs/hardhat-ethers ethers@5.7.2 chai

added 27 packages, changed 9 packages, and audited 384 packages in 38s

107 packages are looking for funding
  run `npm fund` for details

11 vulnerabilities (7 low, 3 high, 1 critical)

To address issues that do not require attention, run:
  npm audit fix

Some issues need review, and may require choosing
a different dependency.

Run `npm audit` for details.
PS D:\WEB 3\fresh-test>
```


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

* Implementation Phase: Final Output (no error)

• **Error Detection & Log Analysis:** Implemented structured debugging using Hardhat console logs to identify and resolve compilation, deployment, and runtime errors in smart contracts efficiently.

• **Performance & Execution Tracking:** Enhanced debugging accuracy by monitoring transaction flow, gas usage, and function outputs through Hardhat's console and network tracing tools.

• **Code Validation & Testing Alignment:** Ensured smart contract reliability by cross-verifying console outputs with expected behaviors during test execution, maintaining consistency and transparency in results.

• **Scalability & Continuous Improvement:** Developed a modular debugging approach allowing quick iteration, improving developer efficiency, and supporting scalable project development across multiple contract versions.

* Observations

1. Hardhat console and console.log() help in tracking smart contract behavior during development for quick error detection.
2. Debugging through local logs ensures smooth and error-free deployment before moving to the live blockchain.

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:

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