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

Debugging plays a crucial role in the development of smart contracts, ensuring their accuracy, security, and reliability before deployment. During this phase, developers utilize **Hardhat Console** and **console.log** statements to detect and resolve logical, runtime, or arithmetic errors within Solidity code. Hardhat provides a powerful **built- in debugging environment** that supports step by-step transaction tracing, variable inspection, and gas usage analysis. This allows developers to closely monitor contract behavior, identify issues early, and optimize performance prior to deployment on the Ethereum mainnet.

Algorithm

1. **Step 1:** Install and configure the Hardhat environment by running the command:
   `npm install --save-dev hardhat`
2. **Step 2:** Initialize a new Hardhat project and create the smart contract file (e.g., `Storage.sol`).
3. **Step 3:** Integrate debugging logs into the Solidity code using: `import "hardhat/console.sol";`
   `console.log("Debug Value:", variableName);`
4. **Step 4:** Develop deployment and testing scripts inside the **scripts/** and **test/** directories to automate contract execution and validation.
5. **Step 5:** Start the Hardhat network and deploy the contract using: `npx hardhat run scripts/deploy.js`
6. **Step 6:** Monitor the console output to analyze the execution flow, verify variable values, and detect logical or computational inconsistencies.
7. **Step 7:** Resolve identified issues, refine the code, and re-run the tests iteratively until the contract executes correctly without errors.
8. **Step 8:** Once verified, prepare the final contract for deployment to the testnet or mainnet.

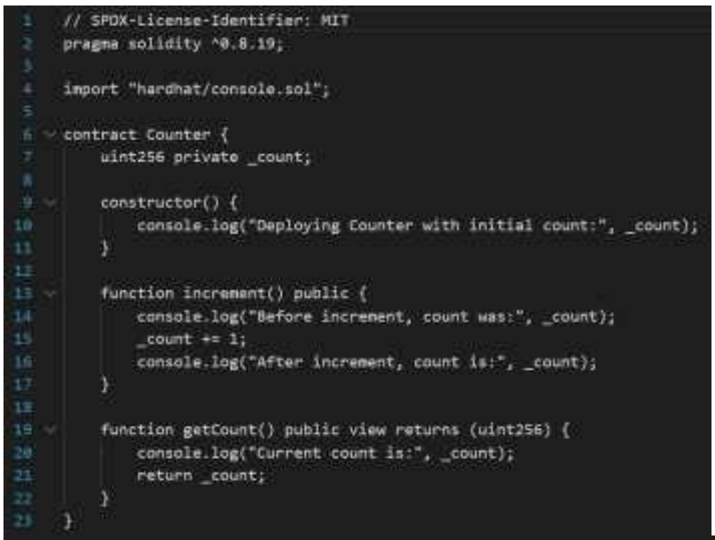
* Softwares used

- **Node.js** – To run JavaScript and manage dependencies.
- **Hardhat** – For compiling, deploying, and debugging smart contracts.
- **Solidity Compiler (solc)** – To compile the Solidity code.
- **Visual Studio Code (VS Code)** – For writing and editing the smart contracts.
- **Ethereum Local Network (Hardhat Network)** – To simulate blockchain and test the contracts.

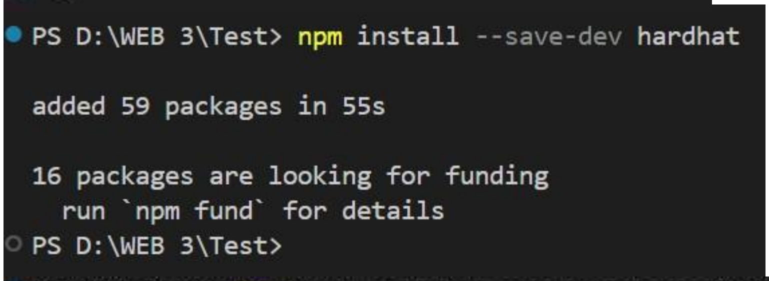
*****As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.*

* Implementation Phase: Final Output (no error)

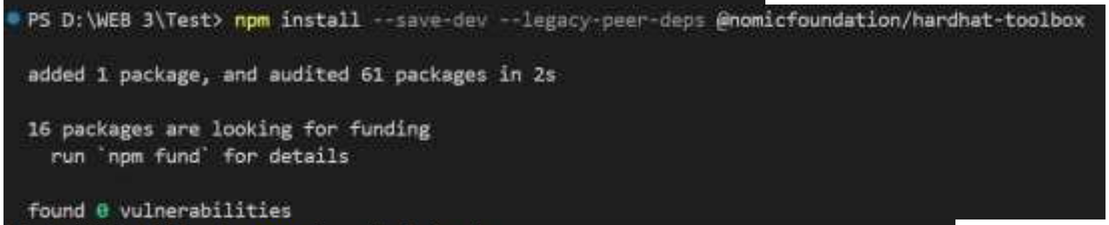
Steps :



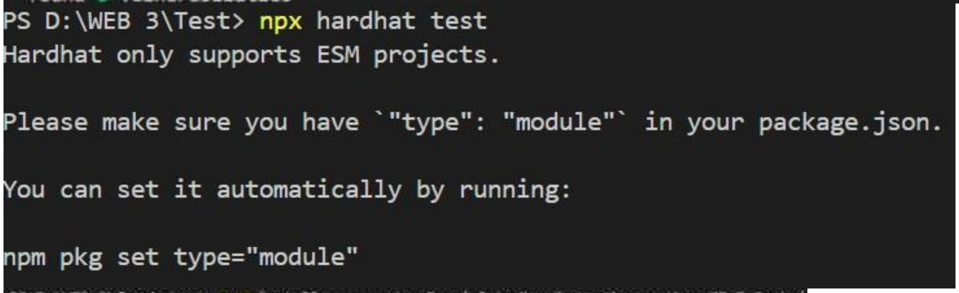
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

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;
    }
}
```



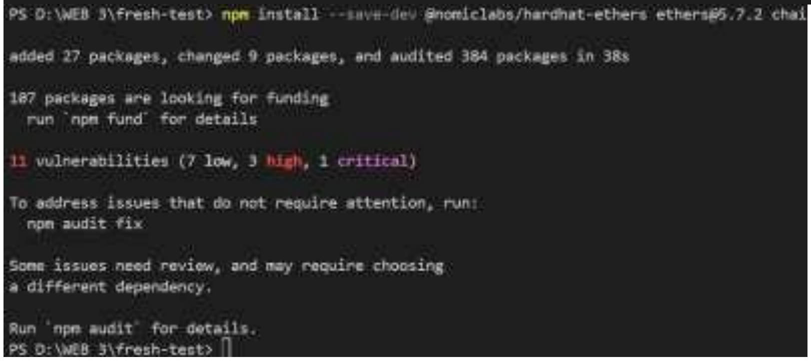
```
PS D:\WEB 3\Test> npm install --save-dev hardhat

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 and Log Analysis:**
Utilized structured debugging techniques with **Hardhat console logs** to efficiently detect, trace, and fix compilation, deployment, and runtime errors in Solidity smart contracts. This systematic approach ensured quick issue identification and improved overall code stability.

• **Performance and Execution Tracking:**
Improved debugging precision by closely monitoring **transaction flow**, **gas consumption**, and **function outputs** using Hardhat's built-in console and **network tracing tools**. This enabled better performance optimization and contract behavior analysis.

• **Code Validation and Testing Alignment:**
Verified the correctness and reliability of smart contracts by matching **console log outputs** with expected results during test execution. This process maintained consistency, transparency, and alignment between debugging and testing phases.

• **Scalability and Continuous Improvement:**
Adopted a **modular debugging framework** that allowed for rapid iteration and reusability across multiple contract versions. This approach boosted developer efficiency, supported scalable project workflows, and encouraged continuous refinement of the development process.

* Observations

1. Identified and resolved deployment and runtime issues effectively using Hardhat console logs.
2. Observed detailed transaction flow and gas usage for better performance analysis.
3. Verified contract behavior and improved debugging accuracy through structured log outputs.

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.