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

Objective/Aim:

To learn how to efficiently debug Ethereum smart contracts using the Hardhat framework's console logs, stack traces, and error tracking tools.

Apparatus/Software Used:

- **Programming Language:** Solidity
- **IDE:** Visual Studio Code
- **Runtime Environment:** Node.js
- **Framework:** Hardhat

Theory/Concept:

Debugging in smart contract development is extremely important because once a contract is deployed on the blockchain, its code cannot be changed. Any small error or overlooked bug can lead to major problems such as:

- Financial losses or frozen assets
- Failed transactions
- Incorrect logic execution
- Contract re-deployment costs

To prevent such issues, **Hardhat** provides powerful debugging utilities that allow developers to identify and fix problems **before deployment**.

Key Debugging Features in Hardhat:

- **Console Logs:** Helps print variable values and messages directly during execution.
- **Stack Traces:** Shows the exact function or line where an error occurs.
- **Event Logs and Gas Reports:** Useful for analyzing function behavior and optimizing gas usage.

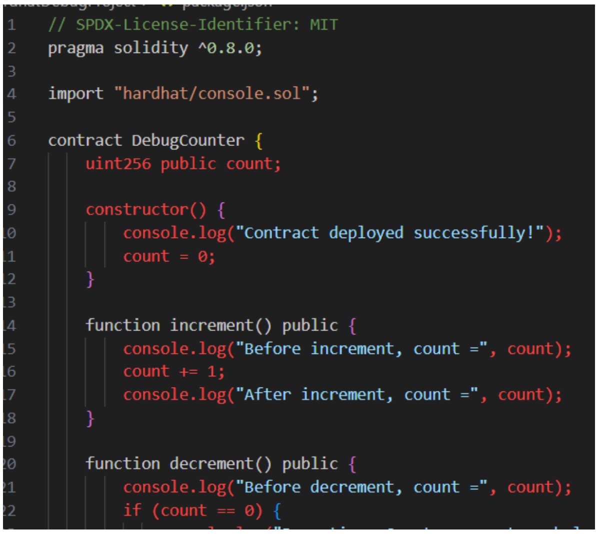
Using `console.log` in Solidity:
Hardhat provides a special debugging library that works only in its local environment. It allows developers to print data types like `uint`, `string`, `address`, and `bool` to track contract behavior.

```
import "hardhat/console.sol";
```

For example, developers can use `console.log()` statements inside functions to display the internal state of a contract while testing. This feature simplifies the debugging process and ensures that the logic works correctly before deploying to a live network.

Procedure:

Step 1:Create Hardhat project
npx hardhat
Step 2: Add DebugCounter.sol in contracts/
Step 3: Add test file in test/debug.js
Step 4: Compile
npx hardhat compile
Step 5: Run test
npx hardhat test
Step 6: Observe log output printed in terminal



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

import "hardhat/console.sol";

contract DebugCounter {
    uint256 public count;

    constructor() {
        console.log("Contract deployed successfully!");
        count = 0;
    }

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

    function decrement() public {
        console.log("Before decrement, count =", count);
        if (count == 0) {
```

Smart contract

Observation Table:

Test Case	Input / Action	Output (Console Log)
Increment counter	increment()	Before increment: 0 → After increment: 1
Increment again	increment()	Before increment: 1 → After increment: 2
Decrement counter	decrement()	Before decrement: 2 → After decrement: 1
Decrement at zero	decrement() when count = 0	Revert: Counter cannot go negative

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.

Signature of the Faculty: