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 : Solidity Patterns – Advanced Inheritance

Objective/Aim:

- To understand the concept of **inheritance** in Solidity.
- To study **advanced inheritance patterns** used in Ethereum smart contracts.
- To explore how multiple contracts can share and override functionalities safely.
- To implement **multi-level** and **multiple inheritance** without conflicts or errors.

Apparatus/Software Used:

- **Programming Language**: Solidity
- **IDE/Compiler**: Remix IDE / Hardhat / Foundry
- **Client**: Local Ethereum testnet (Ganache, Hardhat, or Geth)
- **Wallet**: MetaMask .

Theory concept:

- Inheritance in Solidity allows a contract to **derive** features (state variables, functions, modifiers) from another contract.

- It promotes **code reusability**, **modularity**, and **organization**.

- Solidity supports **single**, **multi-level**, and **multiple inheritance**.

- **Function Overriding:** Child contracts can override parent functions using the override keyword.
- **Virtual Functions:** Marked with virtual, allowing them to be overridden in child contracts.
- **Super Keyword:** Calls the immediate parent function in the inheritance hierarchy.
- **C3 Linearization:** Solidity resolves multiple inheritance using a linearized order to avoid ambiguity.

Procedure:

1. Create a **Base contract (A)** with virtual functions.
2. Create another **Base contract (B)** with similar or related functions.
3. Define a **Derived contract (C)** inheriting from both (A, B).
4. Use `override` to specify which base function implementation is used.
5. Deploy the derived contract and execute inherited and overridden functions.

.

.

Observation:

- Functions from multiple parent contracts are accessible in the derived contract.
- Function overriding works correctly using virtual and override.
- The compiler follows a **deterministic linear order** for resolving multiple inheritance.

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 :

Signature of the Faculty:

Regn. No. :

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