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 gain knowledge of advanced inheritance mechanisms in Solidity.
- To understand how multiple parent contracts interact in inheritance.
- To examine the working of override, virtual, and super keywords in Solidity.
- To study how Solidity resolves ambiguity using C3 linearization in multiple inheritance

## Apparatus/Software Used:

- **Programming Language: Solidity**
- **IDE/Compiler: Remix IDE or VS Code with Hardhat**
- **Blockchain Client: Local Ethereum Test Network (Ganache / Hardhat Network)**
- **Wallet: MetaMask for contract deployment and interaction**

## Theory concept:

Inheritance is a key **object-oriented programming** concept in Solidity that allows one contract to derive properties and functions from another. It helps reduce code repetition and enhances modular design in smart contracts.

There are several advanced aspects of inheritance in Solidity:

- **Single Inheritance:** A contract inherits features from one parent contract only.
- **Multiple Inheritance:** A contract can inherit from more than one parent contract, which may lead to ambiguity if the same function name exists in multiple parents.
- **Virtual and Override Keywords:**
  - Functions in parent contracts that can be overridden must be declared as `virtual`.
  - Functions in child contracts that override parent behavior must use the `override` keyword.
- **Super Keyword:** The `super` keyword is used to call the immediate parent's function implementation in the hierarchy.
- **C3 Linearization:** Solidity uses this method to determine the order in which parent contracts are called, ensuring consistent function resolution even with multiple inheritance.

These features together make Solidity's inheritance system both **powerful and predictable**, ensuring clarity and reusability in complex contract hierarchies.

# Procedure:



1: Launch **Remix IDE** or set up a **Hardhat project**.
2: Create multiple parent contracts (e.g., A and B) with functions marked as virtual.
3: Create a child contract (e.g., C) that inherits from both A and B.
4: Use the override keyword in the child contract to redefine functions from parents.
5: Deploy the contracts on a **local Ethereum test network** using MetaMask.
6: Call inherited and overridden functions to observe how Solidity decides which parent function executes first.
7: Analyze results to understand the order of execution and role of super.

# Observation:

The contracts compiled without errors and were deployed successfully. When functions with the same name were called from the derived contract, Solidity followed the **C3 linearization rule** to determine the order of execution. The results confirmed that the `super` keyword invokes parent functions sequentially according to this hierarchy. This demonstrated how Solidity handles multiple inheritance and avoids ambiguity through a clear and structured inheritance path.

## 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.*