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

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

1. Aim

   To understand and implement advanced inheritance patterns in Solidity, including multi-level, multiple, and hierarchical inheritance, and observe how function overriding and super keyword works in smart contracts.

2. Theory

   Inheritance in Solidity:

   Inheritance allows one contract to inherit properties and methods of another. It promotes code reusability and modular design.

   -Types of Inheritance:

   - Single Inheritance – One contract inherits from one parent.
   - Multiple Inheritance – One contract inherits from multiple parents.
   - Multi-level Inheritance – A chain of inheritance where one contract inherits another which in turn inherits another.
   - Hierarchical Inheritance – Multiple contracts inherit from the same parent contract.

3. Key Concepts:
   - Overriding Functions: Child contracts can override parent functions using the override keyword.
   - Virtual Functions: Functions in the parent contract must be declared virtual to allow overriding.
   - super Keyword: Calls the parent contract's function in the hierarchy.
   - Linearization (C3 Linearization): Solidity uses C3 linearization to resolve the order of execution in multiple inheritance.
4. Benefits:
   - Reusability of code
   - Cleaner, organized contracts
   - Flexibility to extend functionality without modifying parent contracts

* Softwares used

1. Brave browser
2. MetaMask Wallet
3. Remix IDE
4. Sepolia Testnet

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

* Implementation Phase: Final Output (no error)

Step 1: Create Base Contract

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

contract Vehicle {
    string public vehicleType;

    // Virtual function allows child contracts to override
    function start() public virtual returns(string memory) {
        return "Vehicle is starting";
    }
}
```

Step 2: Single Inheritance

```
contract Car is Vehicle {
    string public model;

    function start() public override returns(string memory) {
        return "Car is starting";
    }
}
```

Step 3: Multiple Inheritance

```
contract Electric {
    function charge() public pure returns(string memory) {
        return "Charging Electric Vehicle";
    }
}

contract ElectricCar is Car, Electric {
    function start() public override returns(string memory) {
        return "Electric Car is starting silently";
    }
}
```

Step 4: Multi-Level Inheritance

```
contract SportsCar is Car {
    function start() public override returns(string memory) {
        return string(abi.encodePacked("Sports ", super.start()));
    }
}
```

Step 5: Hierarchical Inheritance

```
contract Bike is Vehicle {
    function start() public override returns(string memory) {
        return "Bike is starting";
    }
}
```


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

* Observations:

Contract	Function Call	Output
Vehicle	start()	Vehicle is starting
Car	start()	Car is starting
ElectricCar	start()	Electric Car is starting silently
ElectricCar	charge()	Charging Electric Vehicle
SportsCar	start()	Sports Car is starting
Bike	start()	Bike is starting

- override and virtual keywords control function overriding.
- super.start() in SportsCar calls parent (Car) function.
- Multiple inheritance requires **careful ordering** to avoid conflicts.

*Conclusion

- Advanced inheritance in Solidity allows **flexible and reusable contract design**.
- **Function overriding** and super keyword allow child contracts to extend or modify parent behavior.
- **Multiple and multi-level inheritance** must follow **linearization rules** to prevent ambiguity.
- Using these patterns improves **readability, modularity, and maintainability** of smart contracts.

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. :

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