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 : Gas Race – Optimizing Smart Contract Efficiency

Objective/Aim:

To analyze and implement **gas optimization techniques** in Ethereum smart contracts, reducing transaction costs and improving execution efficiency while maintaining the same logic and functionality.

Apparatus/Software Used:

- Laptop/PC
- PowerPoint/Word for documentation
- Internet for research

Theory/Concept:

Gas in Ethereum

Gas is the unit of computational cost for executing operations on the Ethereum Virtual Machine (EVM).
Every operation (storage, function call, loop, etc.) consumes a specific gas amount.
Users pay for gas in ETH, depending on gas used × gas price.

Gas Optimization

The goal is to reduce unnecessary gas usage in smart contracts through efficient design and coding.
Optimized contracts save users money and make the blockchain more scalable.

Procedure:

- Define two versions of a smart contract – unoptimized and optimized.
- Each contract will perform a similar operation (e.g., summing numbers or storing data).
- Deploy both contracts on Remix IDE.
- Use Remix Gas Reporter to record gas used for each function call.
- Compare results and identify which version is more efficient.
- Document and analyze the optimization impact.

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

contract GasOptimized {
    uint[] public numbers;

    function addNumbers(uint[] memory _nums) public {    infinite gas
        uint length = _nums.length;
        uint[] memory temp = new uint[](length);

        for (uint i = 0; i < length; i++) {
            temp[i] = _nums[i]; // Works in memory (cheaper)
        }

        // Write once to storage
        for (uint i = 0; i < length; i++) {
            numbers.push(temp[i]);
        }
    }
}
```

Observation:

- Writing directly to **storage** inside loops significantly increases gas cost.
- Using **memory** variables and minimizing state changes reduces gas usage.
- Each `SSTORE` (storage write) operation is expensive, costing ~20,000 gas.
- Optimized contracts perform the same logic with lower gas consumption and faster execution.

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**		