**UE20CS335 – Blockchain**

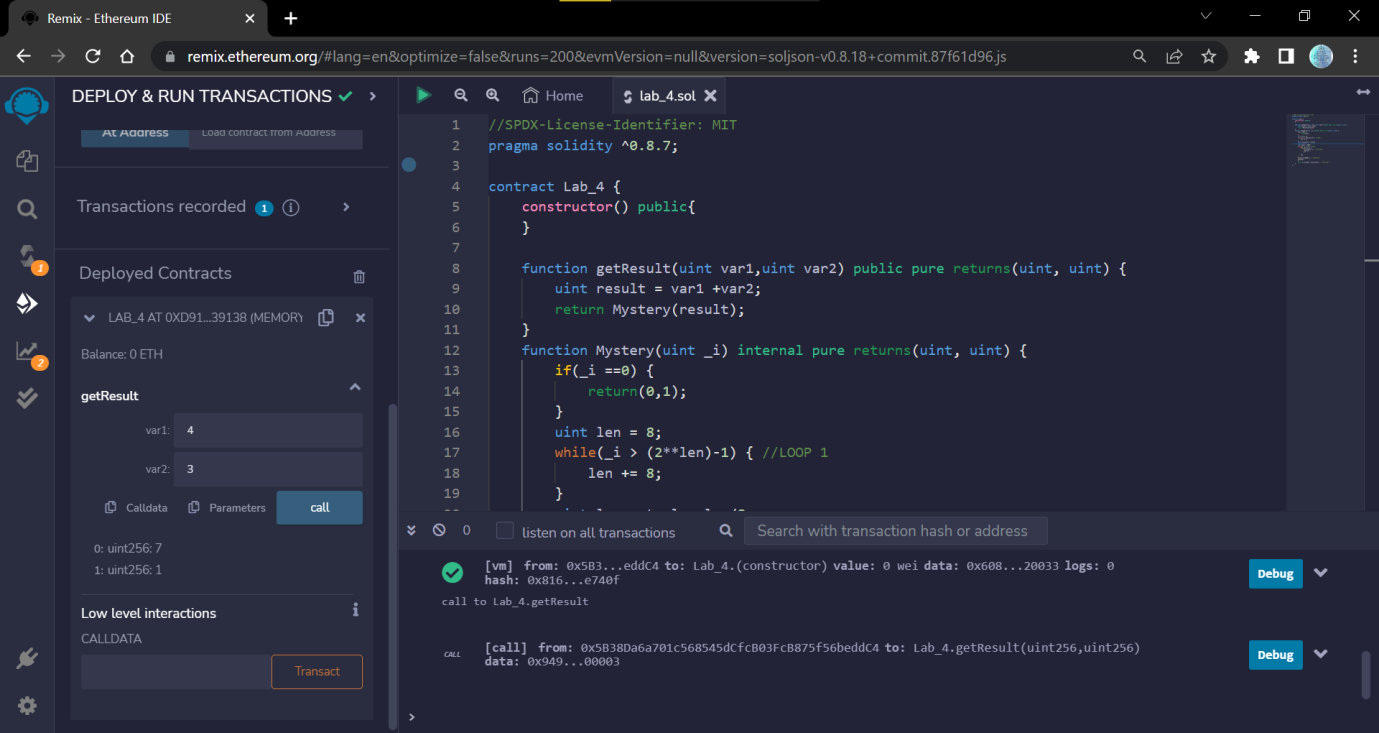
**Hands-On**

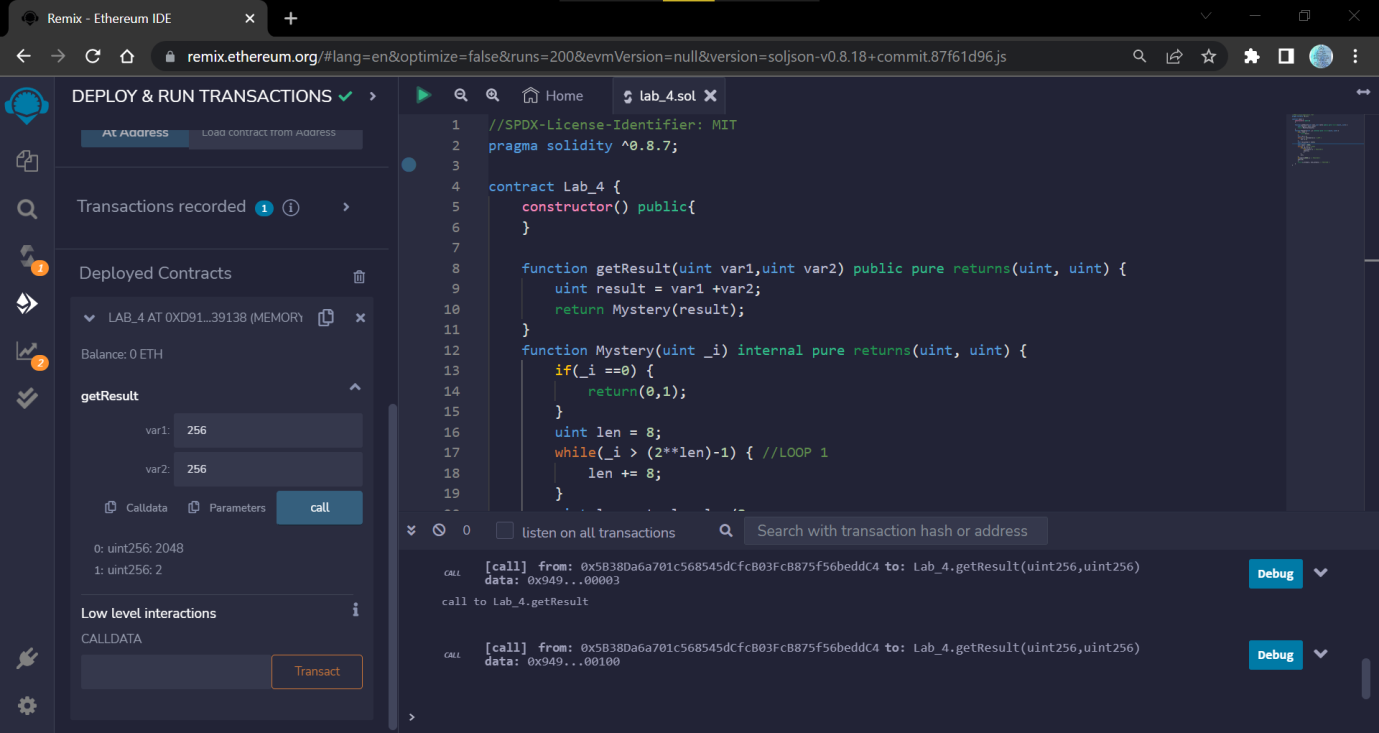
**Name: Vishwa Mehul Mehta**

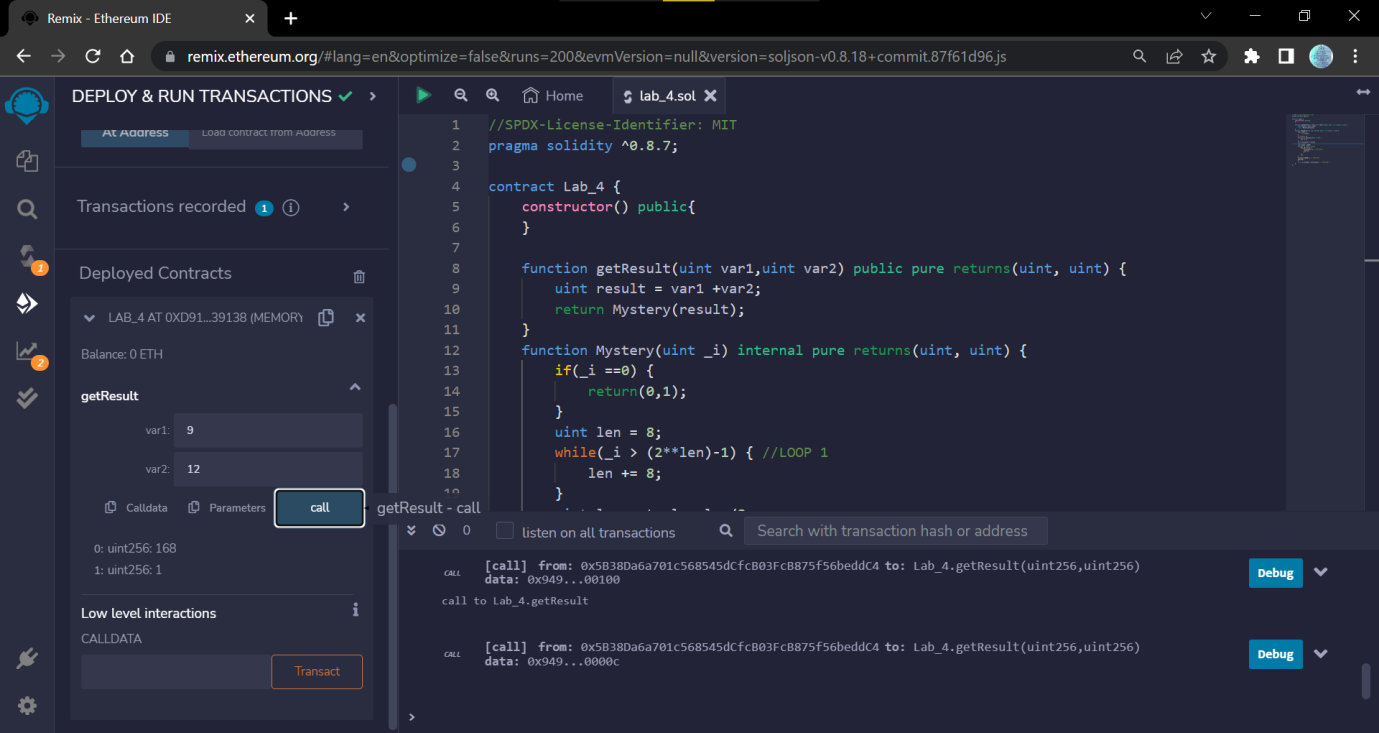
**SRN: PES2UG20CS389**

**Date: 30-03-2023**

**a) Screenshots:**







**b) Output Explanation:**

**1. 4 and 3:**

This is a Solidity smart contract that defines a Lab\_4 contract with a getResult function that takes two unsigned integer parameters 4 and 3 and returns a tuple of two unsigned integers.

The getResult function computes the sum of 4 and 3 and passes the result to a Mystery function. The Mystery function performs a series of operations on the input value 7, which is the sum of 4 and 3. The output of Mystery is then returned as the output of getResult.

The Mystery function first checks if 7 = zero which is false so it enters a loop (LOOP1) that computes the number of bytes needed to represent 7. This is done by repeatedly multiplying the length of the byte representation by 2 until the byte representation can represent the value 7.

After computing the number of bytes needed to represent 7, the function enters another loop (LOOP2) that iterates through each number from 2 to 7 and checks if it is a divisor of 7. If it is, it stores it as the smallest divisor found so far (in POSITION 3).

After the loop is finished, none was found, it sets small to zero. Finally, the function returns a tuple of two unsigned integers. The first integer is the result of left-shifting 7 by the value of 0 (at POSITION 5), and the second integer is the actual number of bytes needed to represent the shifted value of 7.

**c) Loop1 and Loop2:**

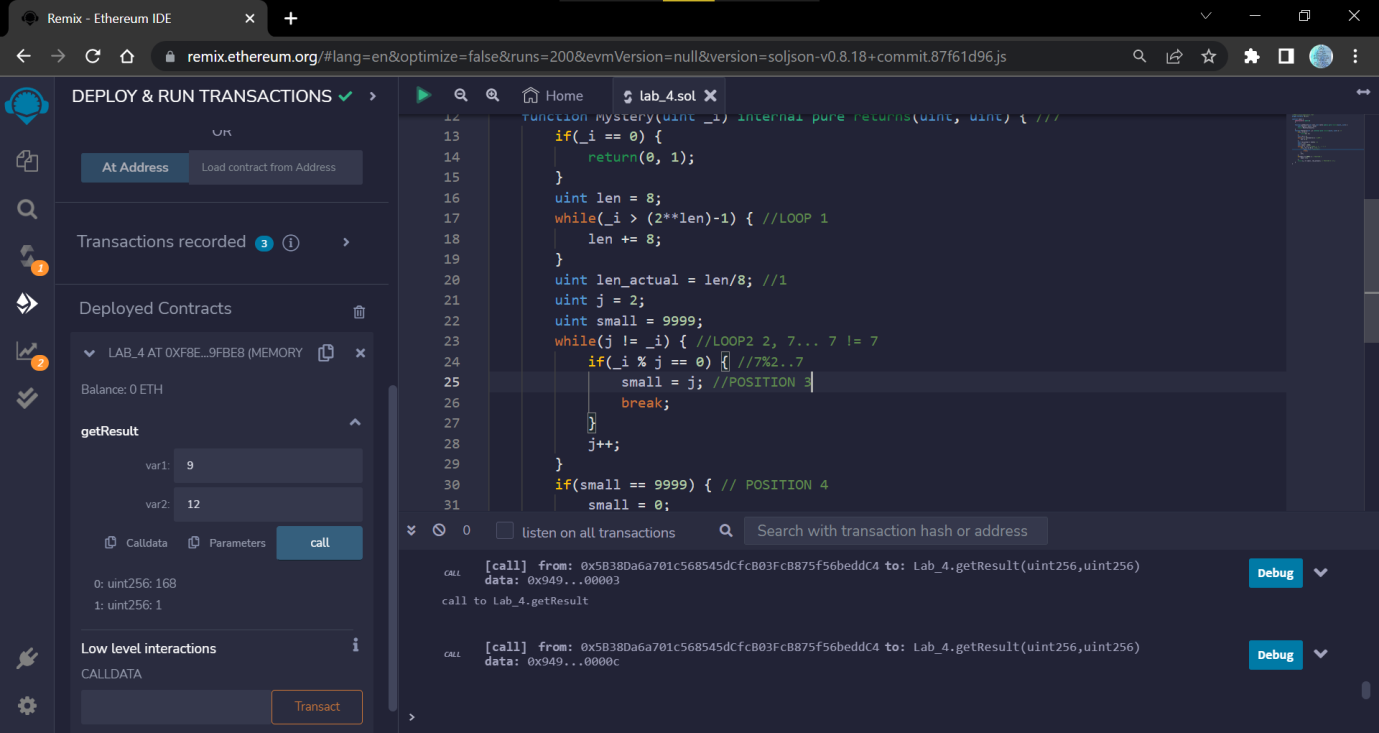
Loop 1:

Computes the number of bytes needed to represent \_i.

Loop 2:

Iterates through each number from j to \_i and checks if it is a divisor of \_i.

**d) Replacing line at POSITION 3:**



**f) Modified Mystery function:**

Code:

//SPDX-License-Identifier: MIT

pragma solidity ^0.8.7;

contract Lab\_4 {

    constructor() public{

    }

    uint public x = 0;

    uint public y = 0;

    function getResult(uint var1,uint var2) public {

        uint result = var1 + var2;

        Mystery(result);

    }

    function Mystery(uint \_i) private { //7

        if(\_i == 0) {

            x = 0;

            y = 1;

        }

        uint len = 8;

        while(\_i > (2\*\*len)-1) { //LOOP 1

            len += 8;

        }

        uint len\_actual = len/8; //1

        uint j = 2;

        uint small = 9999;

        while(j != \_i) { //LOOP2 2, 7... 7 != 7

            if(\_i % j == 0) { //7%2..7

                small = j; //POSITION 3

                break;

            }

            j++;

        }

        if(small == 9999) { // POSITION 4

            small = 0;

        }

        x = \_i << small;

        y = len\_actual; // POSITION 5 //?,1

    }

}

**Output SS:**

