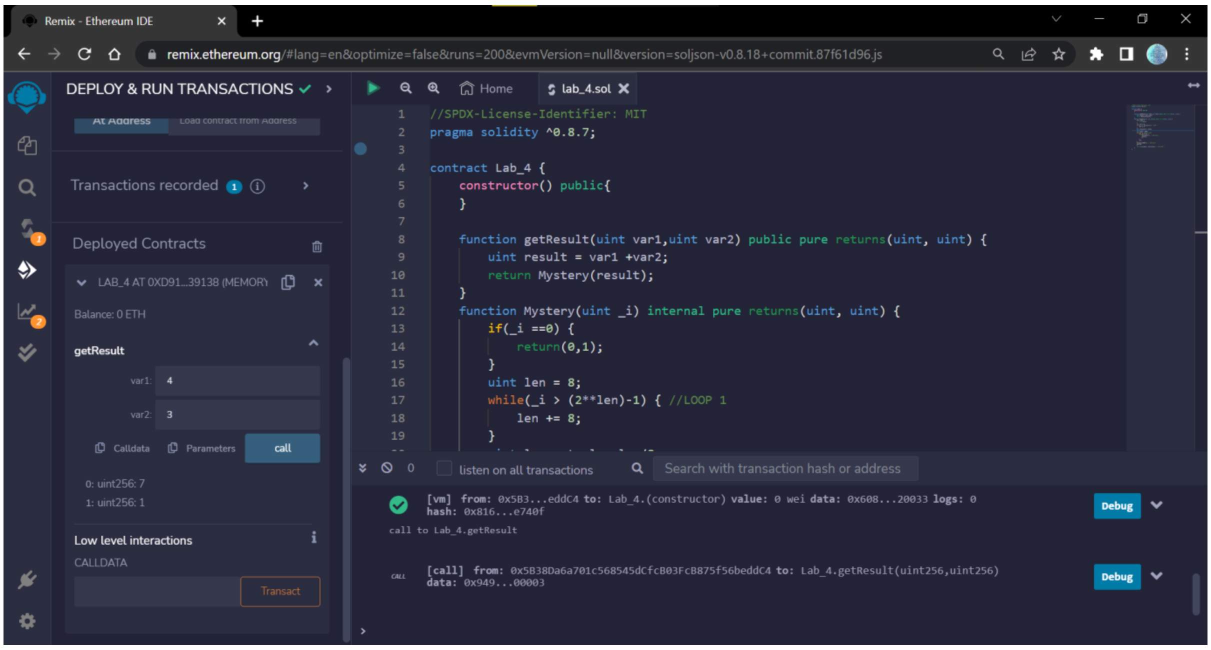
UE20CS335 – Blockchain
Hands-On

Name: Vishwa Mehul Mehta

SRN: PES2UG20CS389

Date: 30-03-2023

a) Screenshots:

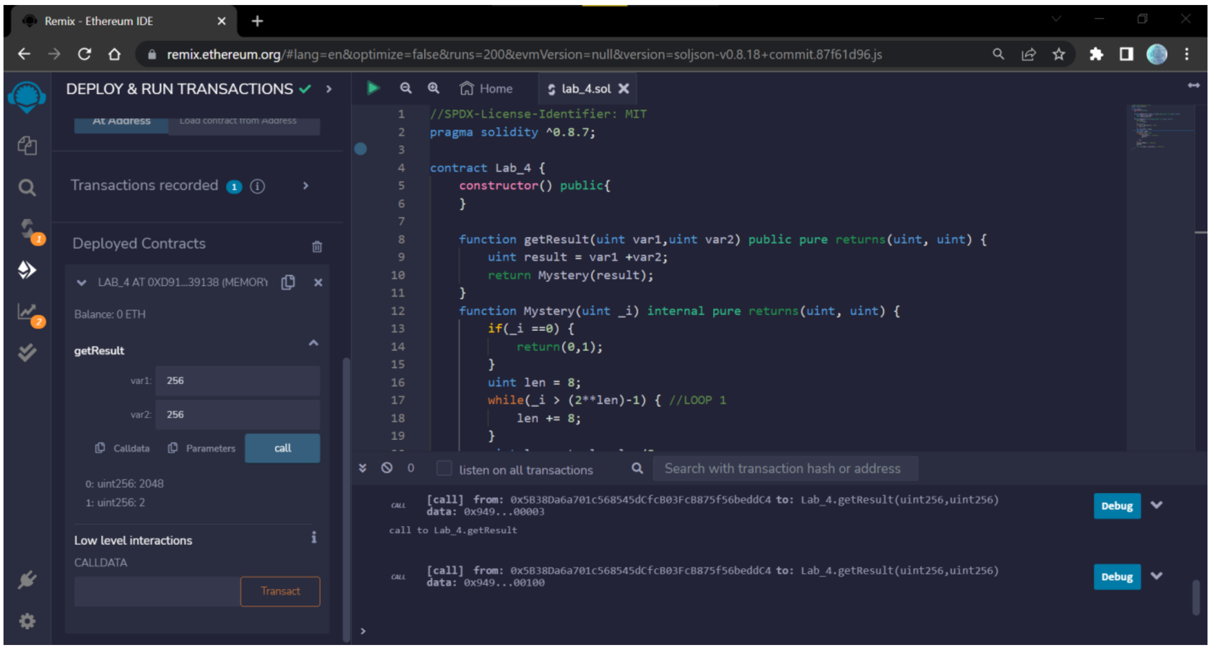


```
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;

contract Lab_4 {
    constructor() public{
    }

    function getResult(uint var1,uint var2) public pure returns(uint, uint) {
        uint result = var1 +var2;
        return Mystery(result);
    }
    function Mystery(uint _i) internal pure returns(uint, uint) {
        if(_i ==0) {
            return(0,1);
        }
        uint len = 8;
        while(_i > (2**len)-1) { //LOOP 1
            len += 8;
        }
```

getResult — var1: 4, var2: 3

```
0: uint256: 7
1: uint256: 1
```

```
[vm] from: 0x5B3...eddC4 to: Lab_4.(constructor) value: 0 wei data: 0x608...20033 logs: 0
hash: 0x816...e740f
call to Lab_4.getResult

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Lab_4.getResult(uint256,uint256)
data: 0x949...00003
```

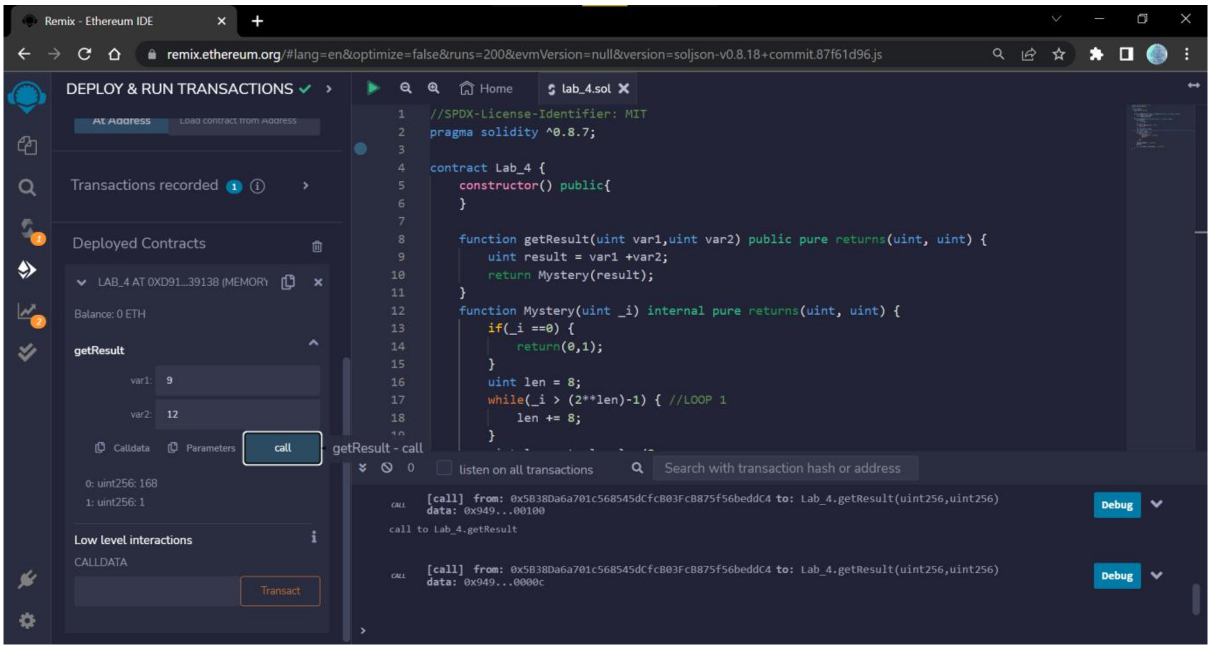


```
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;

contract Lab_4 {
    constructor() public{
    }

    function getResult(uint var1,uint var2) public pure returns(uint, uint) {
        uint result = var1 +var2;
        return Mystery(result);
    }
    function Mystery(uint _i) internal pure returns(uint, uint) {
        if(_i ==0) {
            return(0,1);
        }
        uint len = 8;
        while(_i > (2**len)-1) { //LOOP 1
            len += 8;
        }
```

getResult — var1: 256, var2: 256

```
0: uint256: 2048
1: uint256: 2
```

```
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Lab_4.getResult(uint256,uint256)
data: 0x949...00003
call to Lab_4.getResult

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Lab_4.getResult(uint256,uint256)
data: 0x949...00100
```



```
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;

contract Lab_4 {
    constructor() public{
    }

    function getResult(uint var1,uint var2) public pure returns(uint, uint) {
        uint result = var1 +var2;
        return Mystery(result);
    }
    function Mystery(uint _i) internal pure returns(uint, uint) {
        if(_i ==0) {
            return(0,1);
        }
        uint len = 8;
        while(_i > (2**len)-1) { //LOOP 1
            len += 8;
        }
```

getResult
var1: 9
var2: 12

```
0: uint256: 168
1: uint256: 1
```

```
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Lab_4.getResult(uint256,uint256)
data: 0x949...00100
call to Lab_4.getResult

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Lab_4.getResult(uint256,uint256)
data: 0x949...0000c
```

Remix - Ethereum IDE
remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.18+commit.87f61d96.js
DEPLOY & RUN TRANSACTIONS
At Address
Load contract from Address
Transactions recorded 1
Deployed Contracts
LAB_4 AT 0XD91...39138 (MEMORY)
Balance: 0 ETH
Calldata
Parameters
call
getResult - call
Low level interactions
CALLDATA
Transact
Home
lab_4.sol
listen on all transactions
Search with transaction hash or address
Debug

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:

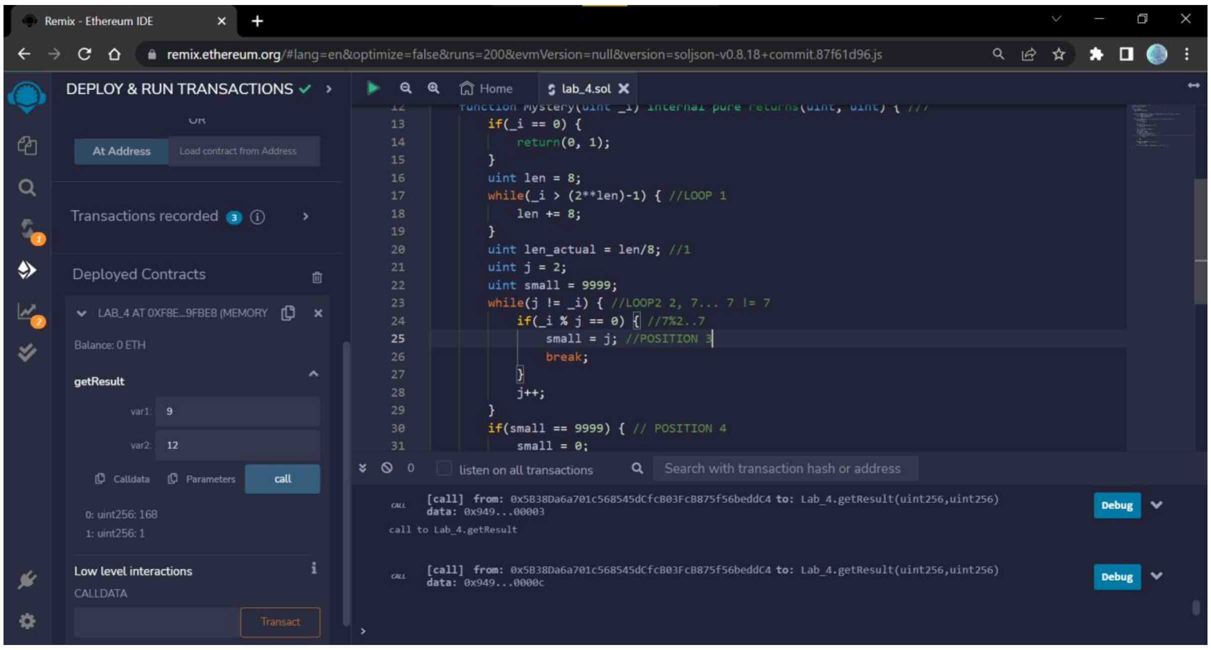
Loop 1:

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:



```
    function Mystery(uint _i) internal pure returns(uint, uint) { ///
        if(_i == 0) {
            return(0, 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;
```

getResult
var1: 9
var2: 12

```
0: uint256: 168
1: uint256: 1
```

```
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Lab_4.getResult(uint256,uint256)
data: 0x949...00003
call to Lab_4.getResult

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Lab_4.getResult(uint256,uint256)
data: 0x949...0000c
```

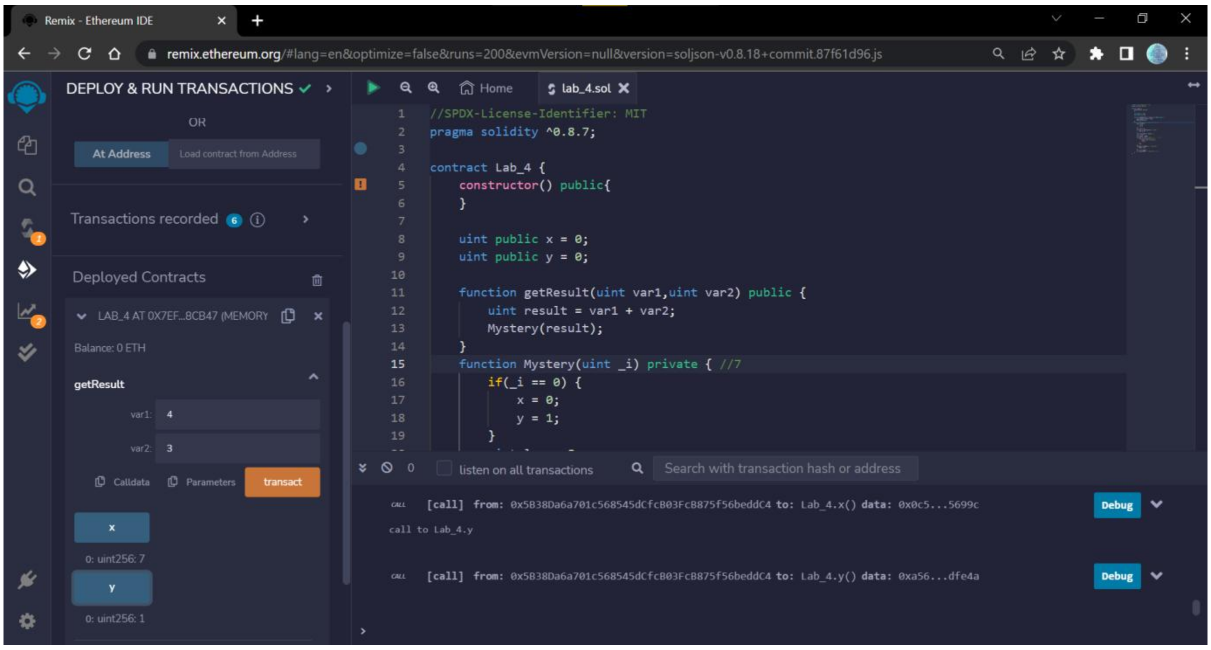
f) Modified Mystery function:

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 {
```

```solidity
        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:



Remix - Ethereum IDE
remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.18+commit.87f61d96.js
DEPLOY & RUN TRANSACTIONS
OR
At Address
Load contract from Address
Transactions recorded 6
Deployed Contracts
LAB_4 AT 0X7EF...8CB47 (MEMORY)
Balance: 0 ETH
getResult
var1: 4
var2: 3
Calldata
Parameters
transact
x
0: uint256: 7
y
0: uint256: 1
Home
lab_4.sol
listen on all transactions
Search with transaction hash or address
Debug

```solidity
//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;
        }
```

```
CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Lab_4.x() data: 0x0c5...5699c
call to Lab_4.y

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Lab_4.y() data: 0xa56...dfe4a
```