PRACTICAL NO 4

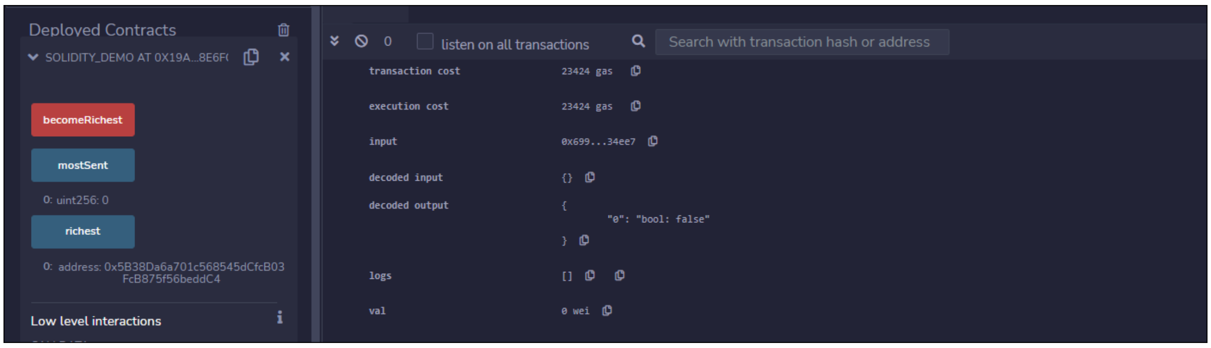
Implement and demonstrate the use of the following in solidity :

A. Withdrawal Pattern, Restricted Access.

<u>Withdrawal Pattern :</u>

```
pragma solidity >=0.5.0 <0.9.0;
contract solidity_demo
{
    address public richest;
    uint public mostSent;

   mapping (address => uint) pendingWithdrawals;

   constructor() public payable
   {
      richest = msg.sender;
      mostSent = msg.value;
   }
   function becomeRichest() public payable returns (bool)
    {
      if (msg.value > mostSent)
      {
         pendingWithdrawals[richest] += msg.value;
         richest = msg.sender;
         mostSent = msg.value;
         return true;
      }
      else
      {
         return false;
      }
    }
}
```

OUTPUT :



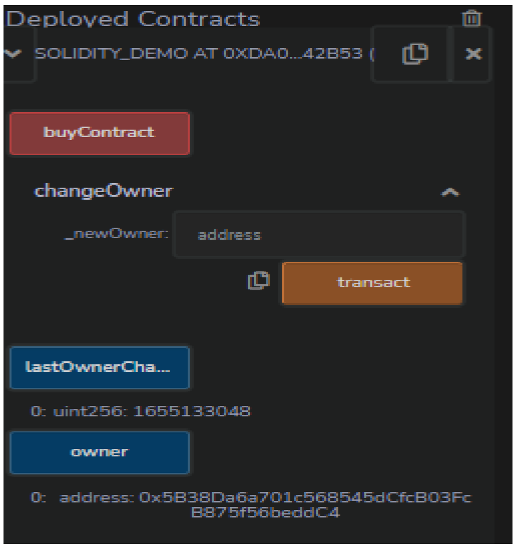
```
Deployed Contracts
SOLIDITY_DEMO AT 0X19A...8E6F(
becomeRichest
mostSent
0: uint256: 0
richest
0: address: 0x5B38Da6a701c568545dCfcB03
            FcB875f56beddC4
Low level interactions

0   listen on all transactions   Search with transaction hash or address
transaction cost        23424 gas
execution cost          23424 gas
input                   0x699...34ee7
decoded input           {}
decoded output          {
                              "0": "bool: false"
                        }
logs                    []
val                     0 wei
```

Restricted Access :

```
pragma solidity >=0.5.0 <0.9.0;
contract solidity_demo
{
    address public owner = msg.sender;
    uint public lastOwnerChange = now;

    modifier onlyBy(address _account) {
        require(msg.sender == _account);
        _;
    }

    modifier onlyAfter(uint _time) {
        require(now >= _time);
        _;
    }

    modifier costs(uint _amount) {
        require(msg.value >= _amount);
        _;
        if (msg.value > _amount) {
            msg.sender.transfer(msg.value - _amount);
        }
    }

    function changeOwner(address _newOwner) public onlyBy(owner) {
        owner = _newOwner;
    }
    function buyContract() public payable onlyAfter(lastOwnerChange + 4 weeks)
costs(1 ether) {
        owner = msg.sender;
        lastOwnerChange = now;
    }
}
```

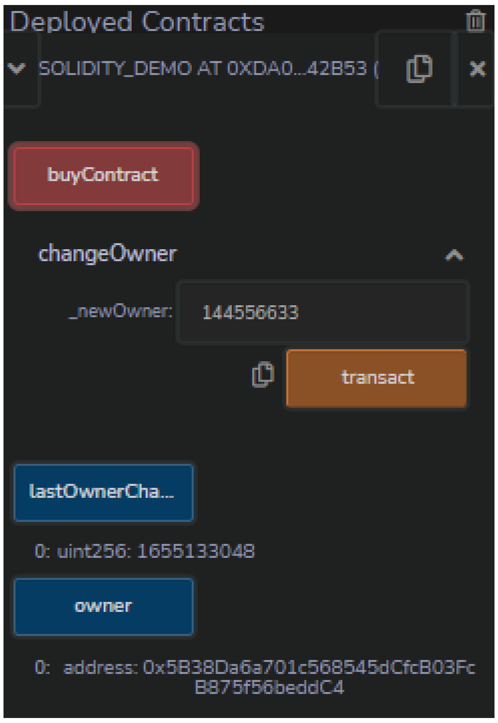
OUTPUT :

Last Owner :-



```
0: uint256: 1655133048
0:  address: 0x5B38Da6a701c568545dCfcB03Fc
             B875f56beddC4
```

After clicking Buy contract :-



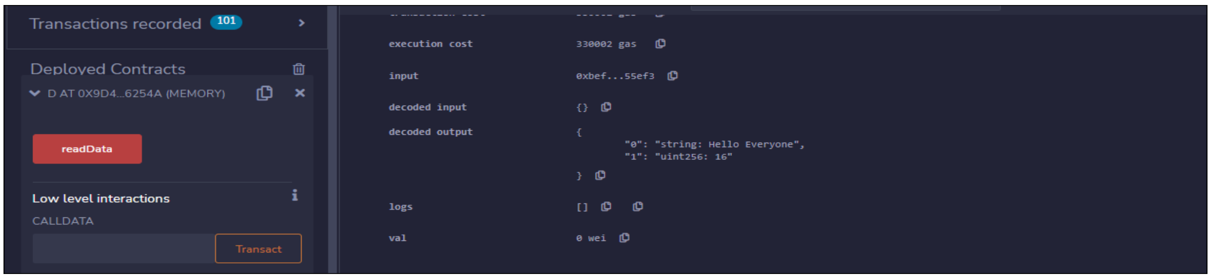
```
144556633
0: uint256: 1655133048
0: address: 0x5B38Da6a701c568545dCfcB03Fc
B875f56beddC4
```

B. Contracts, Inheritance, Constructors, Abstract Contracts, Interfaces.

Contracts :

```
// Solidity program to demonstrate
// visibility modifiers
pragma solidity ^0.5.0;

// Creating a contract
contract contract_example {

// Declaring private
// state variable
uint private num1;

// Declaring public
// state variable
uint public num2;

// Declaring Internal
// state variable
string internal str;

// Defining a constructor
constructor() public {
    num2 = 10;
}

// Defining a private function
function increment( uint data1) private pure returns(uint)
{ return data1 + 1; }

// Defining public functions
function updateValue(uint data1) public { num1 = data1; }
function getValue() public view returns(uint)
{
    return num1;
}

// Declaring public functions
function setStr(
    string memory _str) public;
function getStr(
) public returns (string memory);
}

// Child contract inheriting
// from the parent contract
// 'contract_example'
contract derived_contract is contract_example{

// Defining public function of
// parent contract
function setStr(
    string memory _str) public{
str = _str;
}

// Defining public function
// of parent contract
function getStr(
) public returns (
    string memory){ return str; }
}

//External Contract
contract D {

// Defining a public function to create
// an object of child contract access the
// functions from child and parent contract
function readData() public payable returns(string memory, uint)
{
    contract_example c = new derived_contract();
    c.setStr("Hello Everyone");
    c.updateValue(16);
    return (c.getStr(), c.getValue());
}
}
```

OUTPUT :

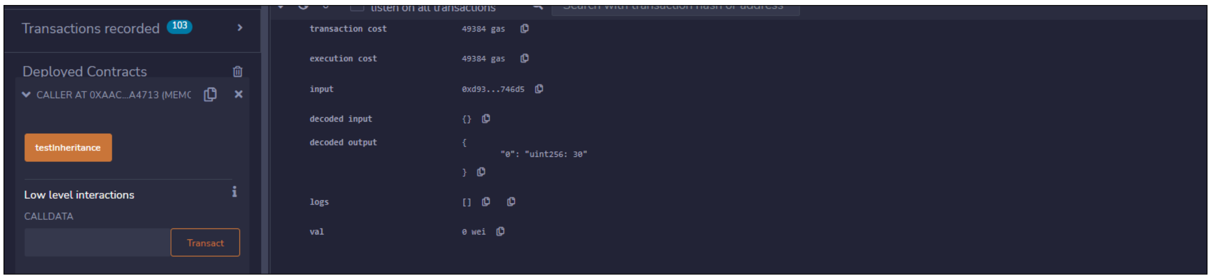


```
execution cost      330002 gas
input               0xbef...55ef3
decoded input       {}
decoded output      {
                        "0": "string: Hello Everyone",
                        "1": "uint256: 16"
                    }
logs                []
val                 0 wei
```

Inheritance :

```
// Solidity program to
// demonstrate
// Single Inheritance
pragma solidity >=0.4.22 <0.6.0;

// Defining contract
contract parent{

    // Declaring internal
    // state variable
    uint internal sum;

    // Defining external function
    // to set value of internal
    // state variable sum
    function setValue() external {
        uint a = 10;
        uint b = 20;
        sum = a + b;
    }
}

// Defining child contract
contract child is parent{

    // Defining external function
    // to return value of
    // internal state variable sum
    function getValue(
    ) external view returns(uint) {
        return sum;
    }
}

// Defining calling contract
contract caller {

    // Creating child contract object
    child cc = new child();

    // Defining function to call
    // setValue and getValue functions
    function testInheritance(
    ) public returns (uint) {
        cc.setValue();
        return cc.getValue();
    }
}
```

OUTPUT :

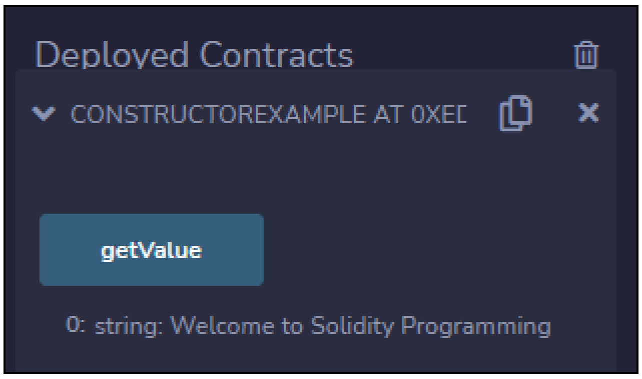


```
transaction cost          49384 gas
execution cost            49384 gas
input                     0xd93...746d5
decoded input             {}
decoded output            {
                                "0": "uint256: 30"
                          }
logs                      []
val                       0 wei
```

Constructors:

```
// Solidity program to demonstrate
// creating a constructor
pragma solidity ^0.5.0;

// Creating a contract
contract constructorExample {

    // Declaring state variable
    string str;

    // Creating a constructor
    // to set value of 'str'
    constructor() public {
        str = "Welcome to Solidity Programming";
    }

    // Defining function to
    // return the value of 'str'
    function getValue(
    ) public view returns (
    string memory) {
        return str;
    }
}
```

OUTPUT :

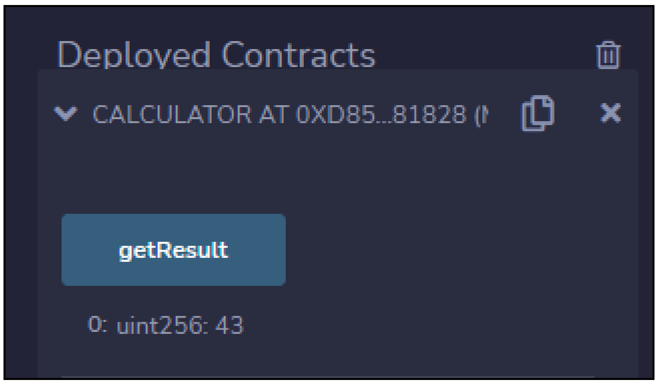


0: string: Welcome to Solidity Programming

Abstract Contracts :

```
pragma solidity ^0.5.0;

contract AbstractClass {
   function getResult() public view returns(uint);
}
contract Calculator is AbstractClass {
   function getResult() public view returns(uint) {
      uint a = 18;
      uint b = 25;
      uint result = a + b;
      return result;
   }
}
```

OUTPUT :

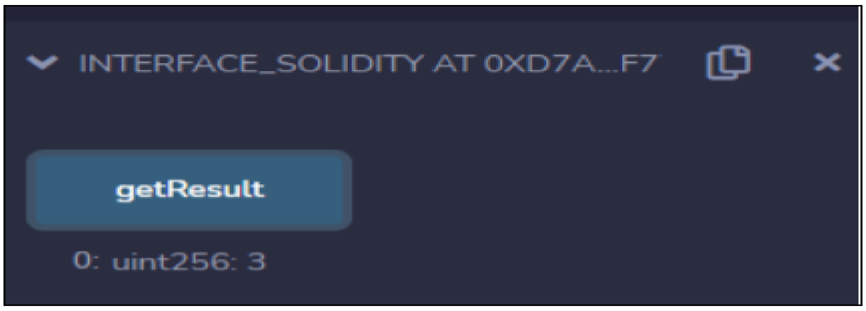


```
0: uint256: 43
```

Interfaces :

```
pragma solidity ^0.5.0;

interface Calculator {
   function getResult() external view returns(uint);
}
contract Interface_Solidity is Calculator {
   function getResult() public view returns(uint){
      uint a = 1;
      uint b = 2;
      uint result = a + b;
      return result;
   }
}
```

OUTPUT:

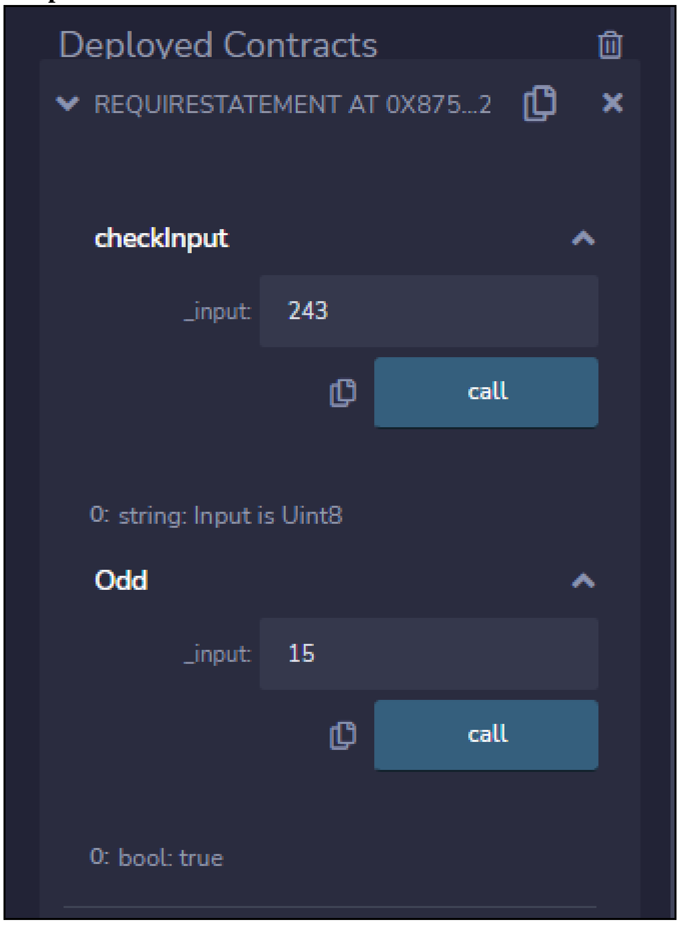


```
0: uint256: 3
```

***Error* handling**

```
// Solidity program to demonstrate require statement
pragma solidity ^0.5.0;

// Creating a contract
contract requireStatement {

    // Defining function to check input
    function checkInput(uint _input) public view returns(string memory){
        require(_input >= 0, "invalid uint8");
        require(_input <= 255, "invalid uint8");

        return "Input is Uint8";
    }

    // Defining function to use require statement
    function Odd(uint _input) public view returns(bool){
        require(_input % 2 != 0);
        return true;
    }
}
```

Output:



```
243
0: string: Input is Uint8
15
0: bool: true
```