

PRACTICAL NO 3:

Implement and demonstrate the use of the following in Solidity :

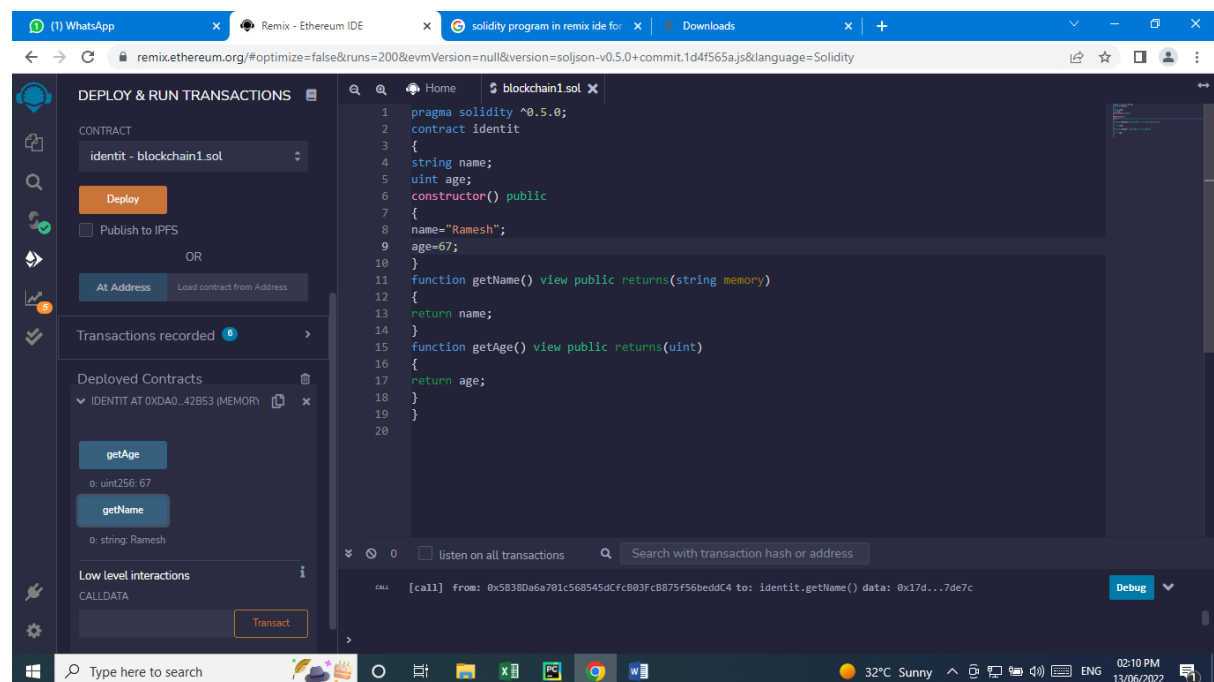
A] Variables, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs, Mappings.

1. Variables :

CODE :

```
pragma solidity ^0.5.0;
contract identity
{
    string name;
    uint age;
    constructor() public
    {
        name="Salman";
        age=57;
    }
    function getName() view public returns(string memory)
    {
        return name;
    }
    function getAge() view public returns(uint)
    {
        return age;
    }
}
```

OUTPUT :



2. Operators :

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract OperatorDemo
{
    // Initializing variables
    uint16 public a = 20;
    uint16 public b = 10;

    // Initializing a variable
    // with sum
    uint public sum = a + b;

    // with the difference
    uint public diff = a - b;

    // with product
    uint public mul = a * b;

    // with quotient
    uint public div = a / b;

    // with modulus
    uint public mod = a % b;

    // decrement value
    uint public dec = --b;

    // with increment value
    uint public inc = ++a;
}
```

OUTPUT :

The screenshot displays the Remix Ethereum IDE interface. The top section shows the 'DEPLOY & RUN TRANSACTIONS' panel with a list of variables and their values:

- a: uint8: 5
- b: uint8: 8
- dec: uint256: 7
- diff: uint256: 3
- div: uint256: 1
- inc: uint256: 6
- mod: uint256: 5
- mul: uint256: 40
- sum: uint256: 13

The middle section shows the Solidity code in the editor, which is a modified version of the code provided in the first block. It uses `uint8` for `a` and `b`, and `uint` for the other variables. The code defines a contract `OperatorDemo` with public variables for each operation.

The bottom section shows the 'CALL' log, indicating a successful transaction from `0x58380a6a701c568545dcfc803fc8875f56bedc4` to `OperatorDemo.sum()` with data `0x853...255cc`. The status bar at the bottom indicates the system is at 32°C Sunny, and the date is 13/06/2022.

3. Loops :

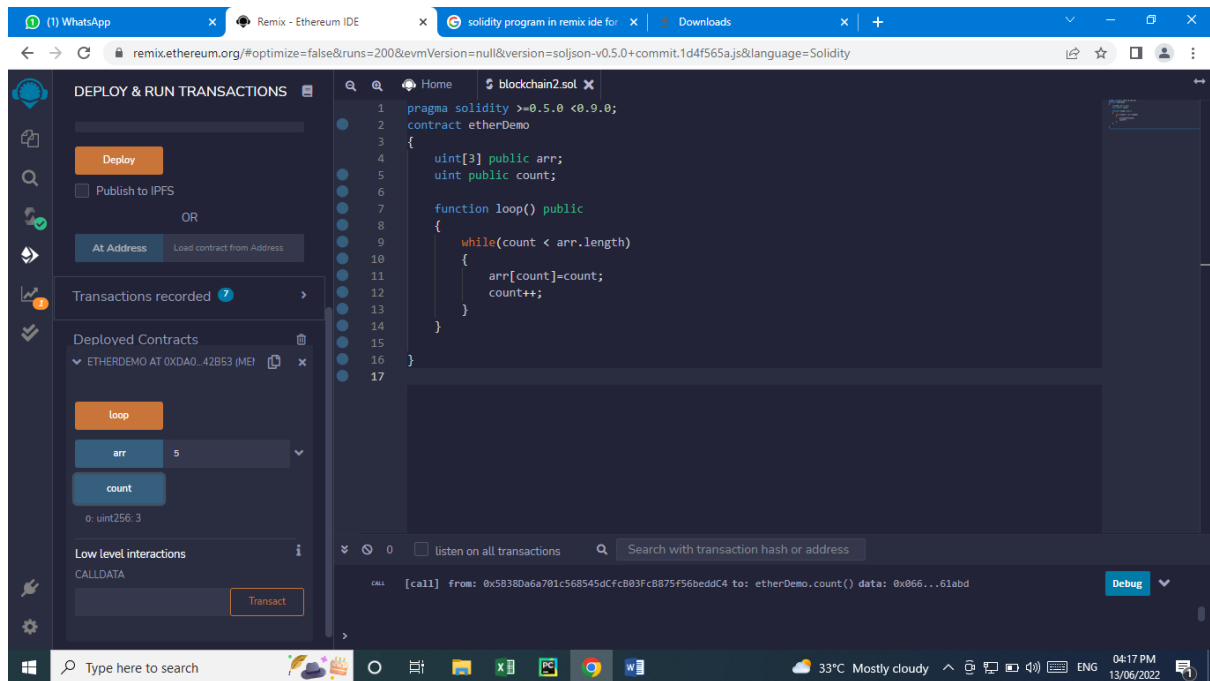
While loop :-

CODE : `pragma solidity >=0.5.0 <0.9.0;`

```
contract etherDemo
{
    uint[3] public arr;
    uint public count;

    function loop() public
    {
        while(count < arr.length)
        {
            arr[count]=count;
            count++;
        }
    }
}
```

OUTPUT :



For loop :-

CODE :

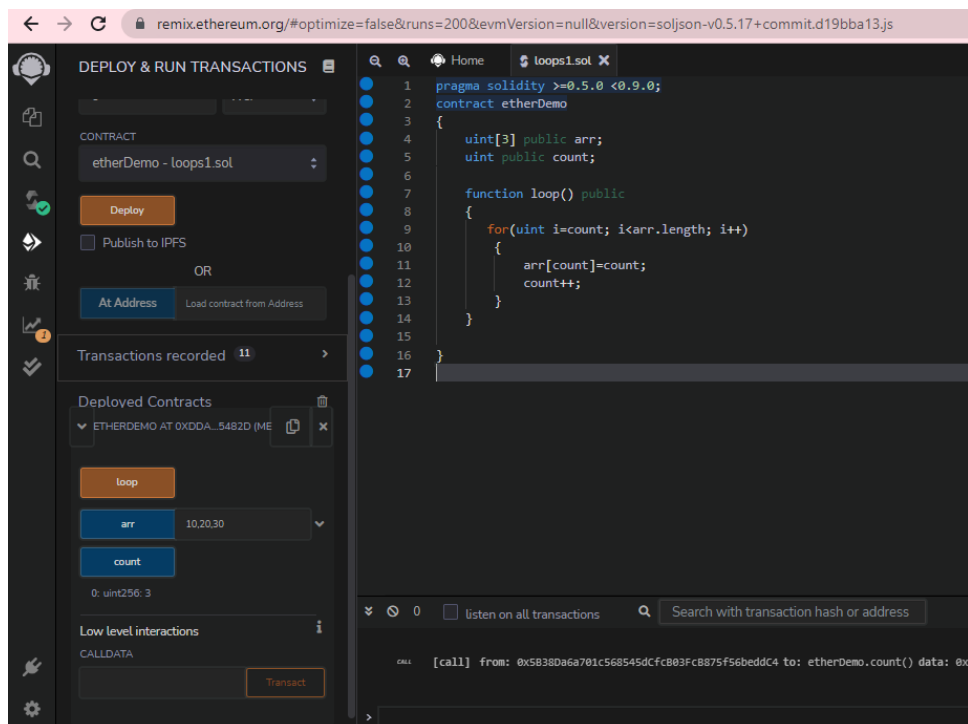
```
pragma solidity >=0.5.0 <0.9.0;
contract etherDemo
{
    uint[3] public arr;
    uint public count;
```

```

function loop() public
{
    for(uint i=count; i<arr.length; i++)
    {
        arr[count]=count;
        count++;
    }
}
}

```

OUTPUT :



Do-while loop :-

CODE :

```

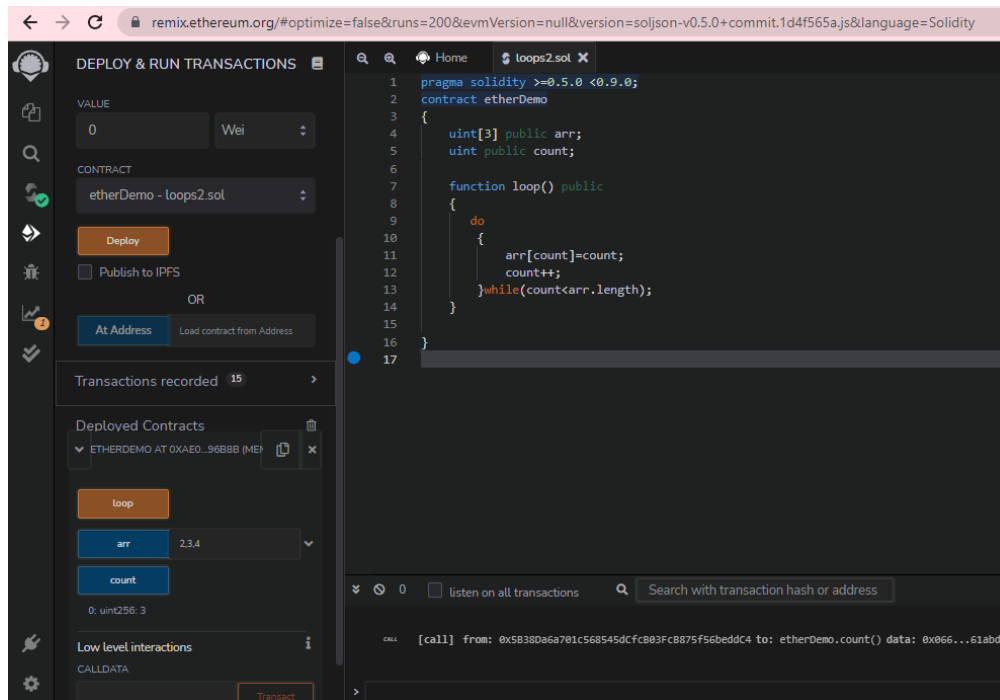
pragma solidity >=0.5.0 <0.9.0;
contract etherDemo
{
    uint[3] public arr;
    uint public count;

    function loop() public
    {
        do
        {
            arr[count]=count;
            count++;
        } while(count<arr.length);
    }
}

```

```
}
```

OUTPUT :



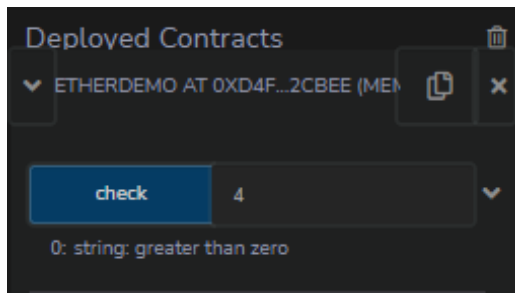
Decision Making :

CODE :

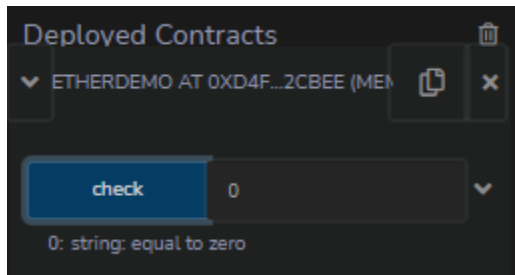
```
pragma solidity >=0.5.0 <0.9.0;
contract etherDemo
{
    function check(int a) public pure returns(string memory)
    {
        string memory value;
        if(a>0)
        {
            value="greater than zero";
        }
        else if(a==0)
        {
            value="equal to zero";
        }
        return value;
    }
}
```

OUTPUT :

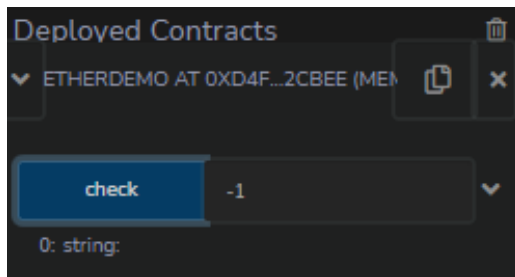
Greater :



Equal :



Less :



Strings :

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract etherDemo
{
    // using double quotes
    string str1 = "Edpresso";

    // using single quotes
    string str2 = 'Educative';

    // using string constructor
    string str3 = new string(2022);

    // getter function to retrieve value of str1
    function getString1() public view returns(string memory) {
        return str1;
    }

    // getter function to retrieve value of str2
    function getString2() public view returns(string memory) {
        return str2;
    }
}
```

```
// getter function to retrieve value of str3
function getString3() public view returns(string memory) {
    return str3;
}
}
```

OUTPUT :

The screenshot displays the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar shows the 'CONTRACT' dropdown set to 'etherDemo - strings.sol'. The 'Deploy' button is highlighted. Below it, the 'Published to IPFS' checkbox is unchecked. The 'At Address' section allows loading a contract from an address. The 'Transactions recorded' section shows 17 transactions. The 'Deployed Contracts' section lists 'ETHERDEMO AT 0x5FD...9D88D (ME)'. Under this contract, three functions are listed: 'getString1', 'getString2', and 'getString3'. Each function has a corresponding button and a preview of its output: '0: string: Edpresso' for getString1, '0: string: Educative' for getString2, and '0: string:' for getString3. The 'Low level interactions' section shows the 'CALLDATA' field. On the right, the main editor displays the Solidity code for 'strings.sol'. The code defines a contract 'etherDemo' with three public view functions: 'getString1', 'getString2', and 'getString3', each returning a string value. The bottom status bar shows a call from '0x5B38Da6a701c568545dCfcB03FcB875f56beddC4' to 'etherDemo.getString3()'. The bottom right corner shows a search bar and a checkbox for 'listen on all transactions'.

Arrays :

Fixed length:-

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract etherDemo
{
    uint[] public arr=[10,20,30,40];

    function setter(uint index, uint value) public
    {
        arr[index]=value;
    }
}
```

OUTPUT :

The screenshot displays the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is active. It shows the 'VALUE' set to 0 Wei, the 'CONTRACT' as 'etherDemo - array.sol', and a 'Deploy' button. Below this, there's a section for 'Transactions recorded' (19) and 'Deployed Contracts'. Under 'Deployed Contracts', the 'setter' function is highlighted with a gas cost of 2,800 and the 'arr' variable is shown with a value of 2. The 'Low level interactions' section at the bottom has a 'Transact' button. The main editor on the right shows the Solidity code for 'array.sol':

```
1 pragma solidity >=0.5.0 <0.9.0;
2 contract etherDemo
3 {
4     uint[] public arr=[10,20,30,40];
5
6     function setter(uint index, uint value) public
7     {
8         arr[index]=value;
9     }
10 }
11
12
```

At the bottom of the interface, a transaction log shows a call: '[call] from: 0x5B38Da6a701c568545dCfcB03Fc8875f56beddC4 to: etherDemo.arr(uint256)'.

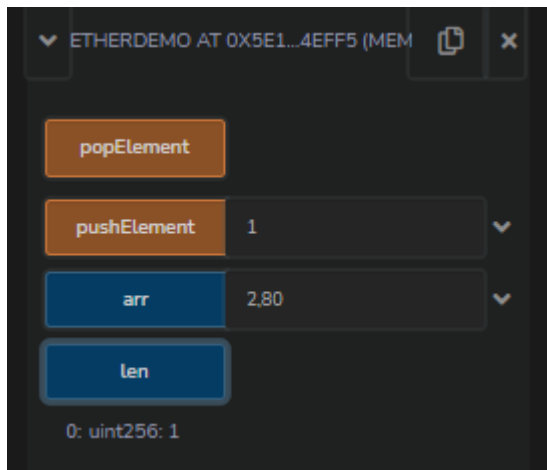
Dynamic length :-

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract etherDemo
{
    uint[] public arr;
    function pushElement(uint item) public
    {
        arr.push(item);
    }
    function len()public view returns(uint)
    {
        return arr.length;
    }
    function popElement()public
    {
        arr.pop();
    }
}
```

OUTPUT :

Push element :-



Pop element :-



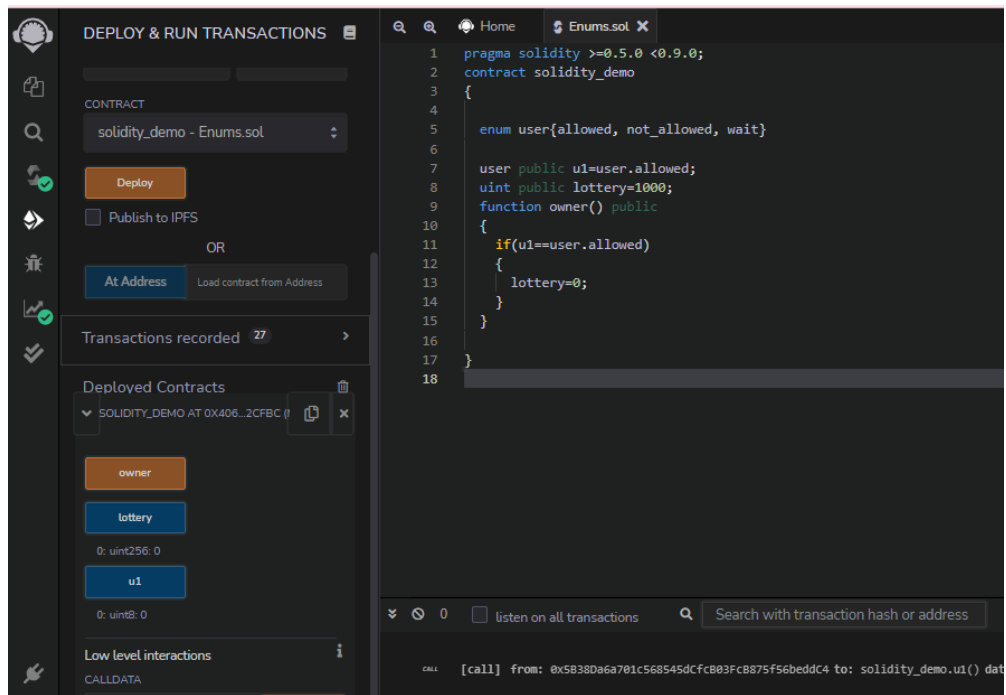
Enums :

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract solidity_demo
{
    enum user{allowed, not_allowed, wait}

    user public u1=user.allowed;
    uint public lottery=1000;
    function owner() public
    {
        if(u1==user.allowed)
        {
            lottery=0;
        }
    }
}
```

OUTPUT :



Structs :

CODE :

```
pragma solidity >=0.5.0 <0.9.0;

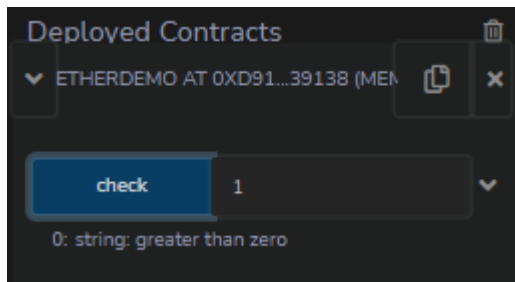
struct Student
{
    uint roll;
    string name;
}

contract solidity_demo
{
    Student public s1;

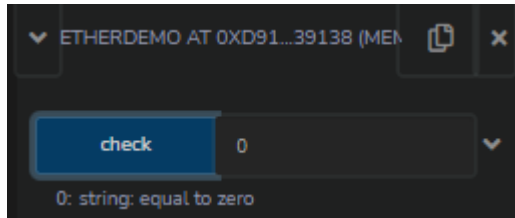
    constructor(uint _roll, string memory _name)
    {
        s1.roll=_roll;
        s1.name=_name;
    }
}
```

OUTPUT :

A greater than zero :



A equal to zero :



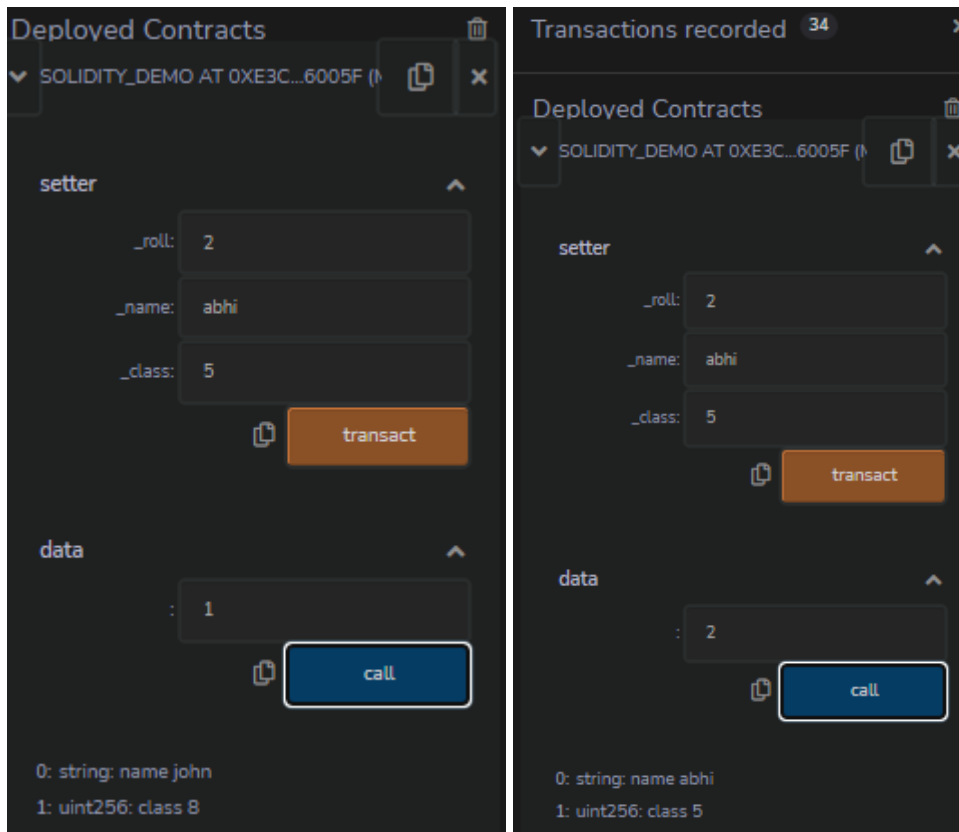
Mappings :

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract solidity_demo
{
    struct Student
    {
        string name;
        uint class;
    }
    mapping (uint => Student) public data;

    function setter(uint _roll, string memory _name, uint _class) public
    {
        data[_roll]= Student(_name, _class);
    }
}
```

OUTPUT :



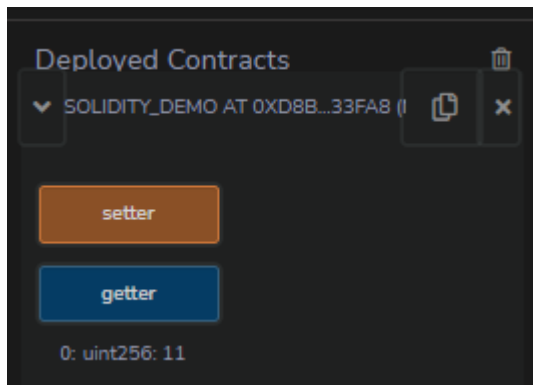
B] Functions, Function Modifiers, View functions, Pure functions, Fallback

Functions :

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract solidity_demo
{
    uint age=10;
    function getter() public view returns(uint)
    {
        return age;
    }
    function setter() public
    {
        age=age+1;
    }
}
```

OUTPUT :



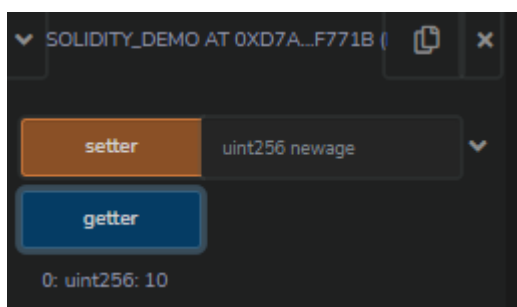
View functions:

CODE :

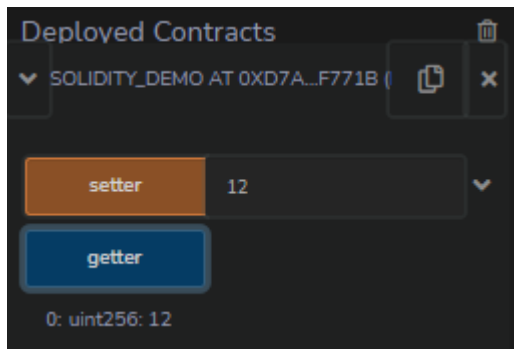
```
pragma solidity >=0.5.0 <0.9.0;
contract solidity_demo
{
    uint age=10;
    function getter() public view returns(uint)
    {
        return age;
    }
    function setter(uint newage) public
    {
        age=newage;
    }
}
```

OUTPUT :

Defined age = 10:



Age after set :

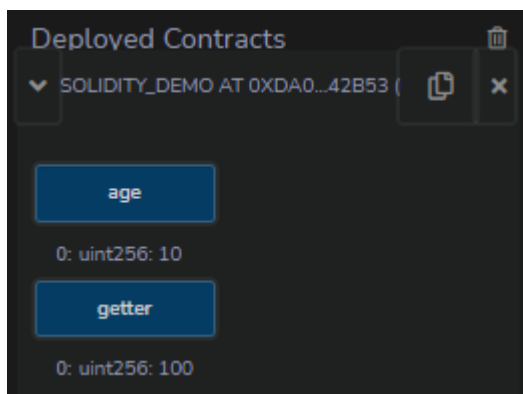


Pure Functions:

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract solidity_demo
{
    uint public age=10;
    function getter() public pure returns(uint)
    {
        uint roll=100;
        return roll;
    }
}
```

OUTPUT :



Function overloading :

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract solidity_demo
{
    function getSum(uint a, uint b) public pure returns(uint)
    {
        return a + b;
    }
}
```

```

    }
    function getSum(uint a, uint b, uint c) public pure returns(uint)
    {
        return a + b + c;
    }
}

```

OUTPUT :



Mathematical functions :

CODE :

```

pragma solidity >=0.5.0 <0.9.0;
contract solidity_demo
{
    function callAddMod() public pure returns(uint){
        return addmod(4, 5, 3);
    }
    function callMulMod() public pure returns(uint){
        return mulmod(4, 5, 3);
    }
}

```

OUTPUT :



Cryptographic functions :

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract solidity_demo
{
    function callKeccak256() public pure returns(bytes32 result)
    {
        return keccak256("ABC");
    }
    function callripemd160() public pure returns(bytes20 result)
    {
        return ripemd160("ABC");
    }
    function callsha256() public pure returns(bytes32 result)
    {
        return sha256("ABC");
    }
}
```

OUTPUT :

