BLOCKCHAIN LAB (EXPERIMENT 04)

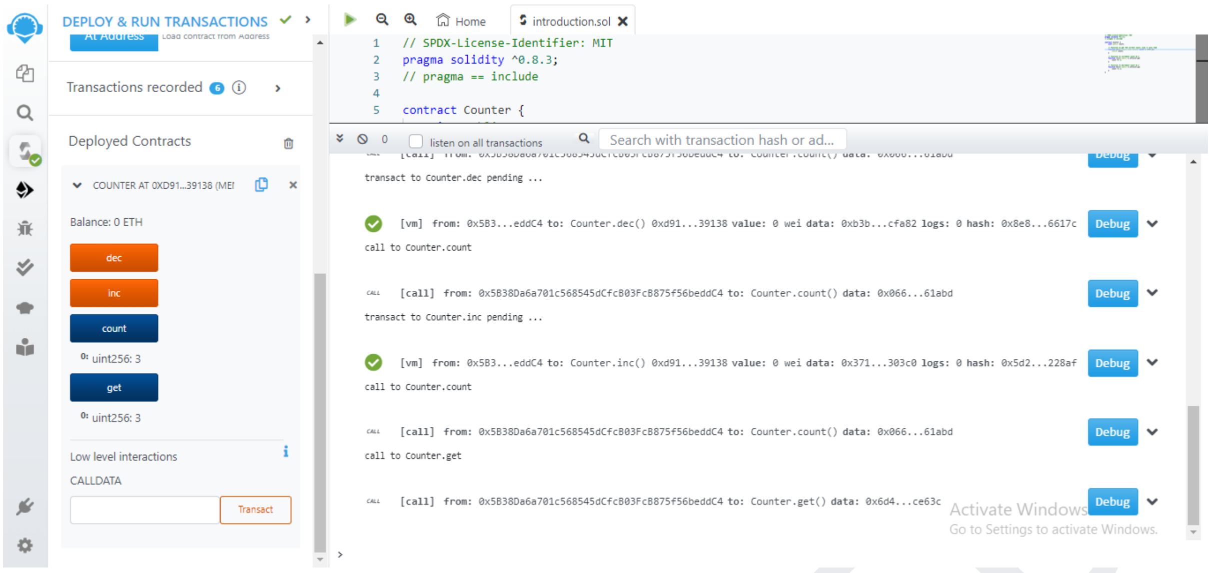
AAYUSH MANISH TALREJA (D17C / 56)

AIM

Hands on Solidity Programming Assignments for creating Smart Contracts

PROGRAM 1

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;
// pragma == include

contract Counter {
   uint public count;

   // Function to get the current count; view == only read
   function get() public view returns (uint) {
      return count;
   }

   // Function to increment count by 1
   function inc() public {
      count += 1;
   }

   // Function to decrement count by 1
   function dec() public {
      count -= 1;
   }
}
```

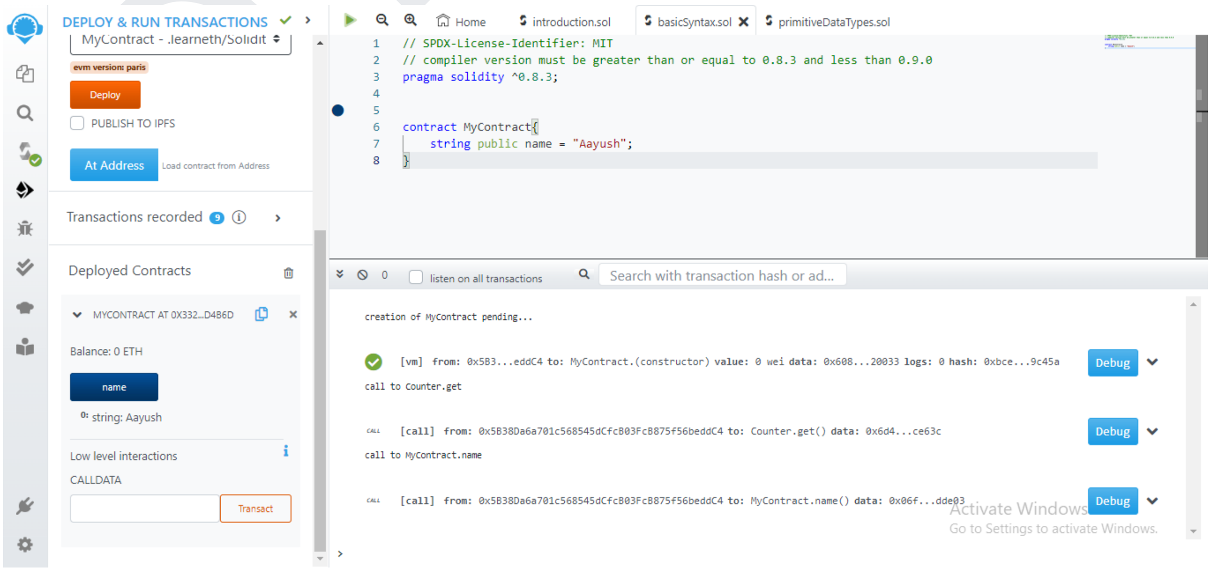


```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;
// pragma == include

contract Counter {
```

```
transact to Counter.dec pending ...

[vm] from: 0x5B3...eddC4 to: Counter.dec() 0xd91...39138 value: 0 wei data: 0xb3b...cfa82 logs: 0 hash: 0x8e8...6617c
call to Counter.count

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Counter.count() data: 0x066...61abd
transact to Counter.inc pending ...

[vm] from: 0x5B3...eddC4 to: Counter.inc() 0xd91...39138 value: 0 wei data: 0x371...303c0 logs: 0 hash: 0x5d2...228af
call to Counter.count

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Counter.count() data: 0x066...61abd
call to Counter.get

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Counter.get() data: 0x6d4...ce63c
```

PROGRAM 2

```
// SPDX-License-Identifier: MIT
// compiler version must be greater than or equal to 0.8.3 and less than 0.9.0
pragma solidity ^0.8.3;

contract MyContract{
    string public name = "Aayush";
}
```



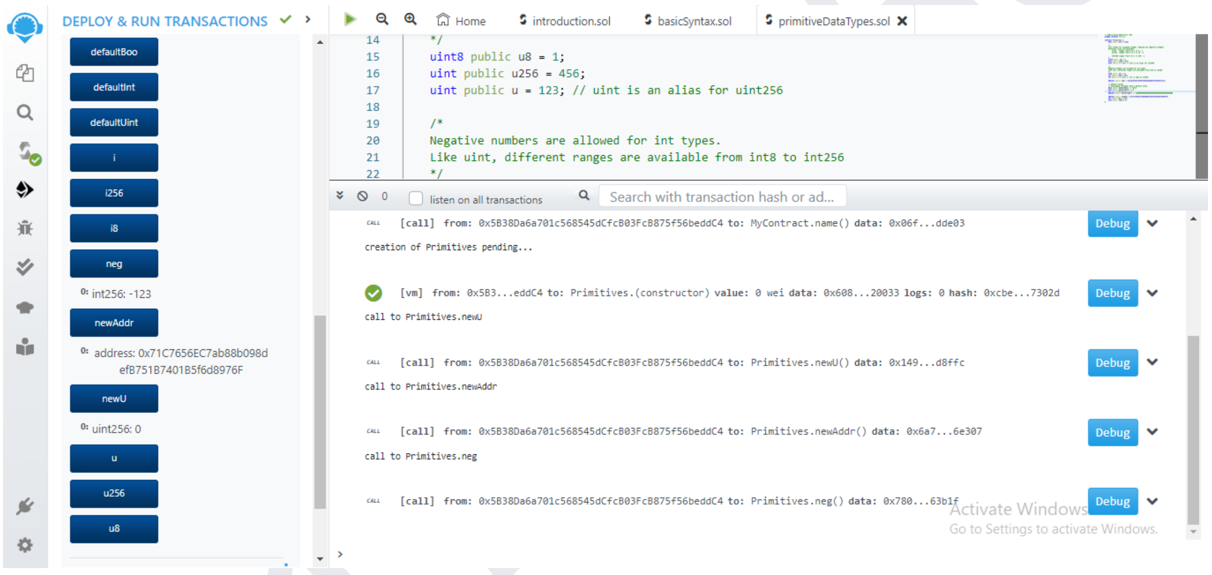
```
// SPDX-License-Identifier: MIT
// compiler version must be greater than or equal to 0.8.3 and less than 0.9.0
pragma solidity ^0.8.3;

contract MyContract{
    string public name = "Aayush";
}
```

```
creation of MyContract pending...

[vm] from: 0x5B3...eddC4 to: MyContract.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0xbce...9c45a
call to Counter.get

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Counter.get() data: 0x6d4...ce63c
call to MyContract.name

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: MyContract.name() data: 0x06f...dde03
```

PROGRAM 3

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract Primitives {
  bool public boo = true;

  /*
  uint stands for unsigned integer, meaning non negative integers
  different sizes are available
    uint8   ranges from 0 to 2 ** 8 - 1
    uint16  ranges from 0 to 2 ** 16 - 1
    ...
    uint256 ranges from 0 to 2 ** 256 - 1
  */
  uint8 public u8 = 1;
  uint public u256 = 456;
  uint public u = 123; // uint is an alias for uint256

  /*
  Negative numbers are allowed for int types.
  Like uint, different ranges are available from int8 to int256
  */
  int8 public i8 = -1;
  int public i256 = 456;
  int public i = -123; // int is same as int256

  address public addr = 0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c;

  // Default values
  // Unassigned variables have a default value
```

```
    bool public defaultBoo; // false
    uint public defaultUint; // 0
    int public defaultInt; // 0
    address public defaultAddr; // 0x0000000000000000000000000000000000000000

    address public newAddr = 0x71C7656EC7ab88b098defB751B7401B5f6d8976F;
    int public neg = -123;
    uint public newU = 0;
}
```



```
0: int256: -123
0: address: 0x71C7656EC7ab88b098d
   efB751B7401B5f6d8976F
0: uint256: 0
```

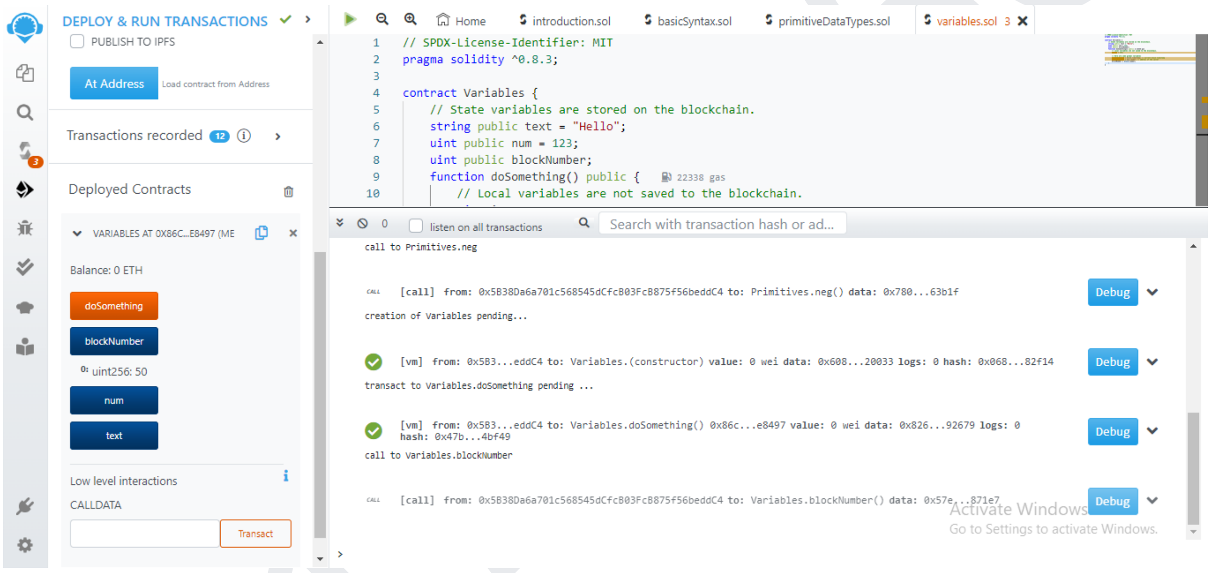
```
    */
    uint8 public u8 = 1;
    uint public u256 = 456;
    uint public u = 123; // uint is an alias for uint256

    /*
    Negative numbers are allowed for int types.
    Like uint, different ranges are available from int8 to int256
    */
```

```
CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: MyContract.name() data: 0x06f...dde03
creation of Primitives pending...

[vm] from: 0x5B3...eddC4 to: Primitives.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0xcbe...7302d
call to Primitives.newU

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Primitives.newU() data: 0x149...d8ffc
call to Primitives.newAddr

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Primitives.newAddr() data: 0x6a7...6e307
call to Primitives.neg

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Primitives.neg() data: 0x780...63b1f
```

Activate Windows
Go to Settings to activate Windows.

PROGRAM 4

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract Variables {
    // State variables are stored on the blockchain.
    string public text = "Hello";
    uint public num = 123;
    uint public blockNumber;
    function doSomething() public {
        // Local variables are not saved to the blockchain.
        uint i = 456;

        // Here are some global variables
        uint timestamp = block.timestamp; // Current block timestamp
        address sender = msg.sender; // address of the caller
        blockNumber = block.number;
    }
}
```

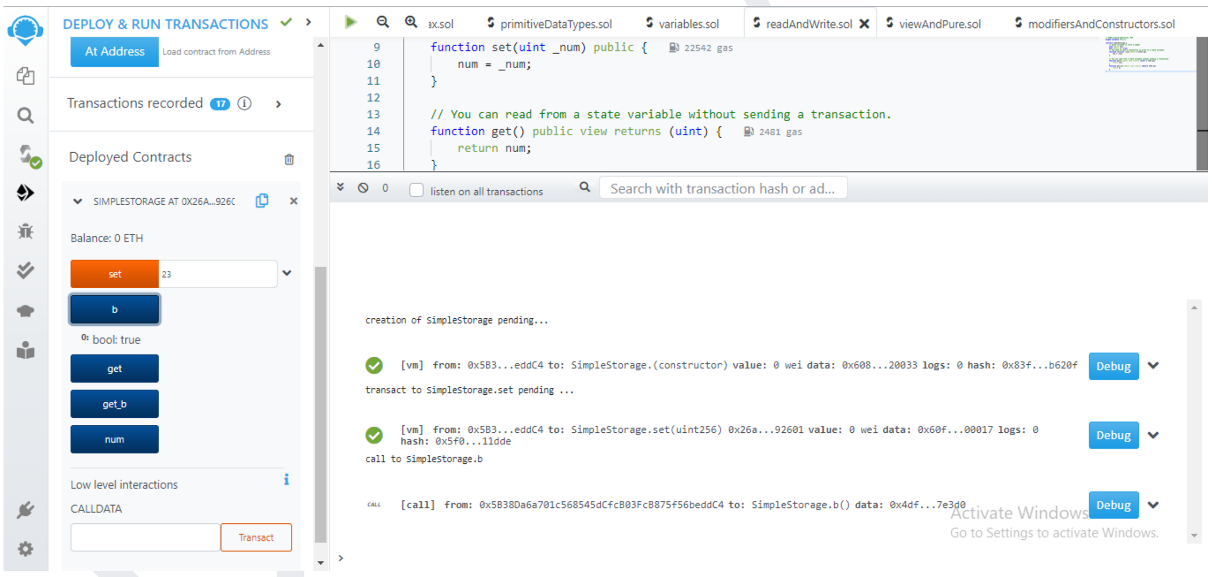


```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract Variables {
    // State variables are stored on the blockchain.
    string public text = "Hello";
    uint public num = 123;
    uint public blockNumber;
    function doSomething() public {    22338 gas
        // Local variables are not saved to the blockchain.
```

```
call to Primitives.neg

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Primitives.neg() data: 0x780...63b1f
creation of Variables pending...

[vm] from: 0x5B3...eddC4 to: Variables.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0x068...82f14
transact to Variables.doSomething pending ...

[vm] from: 0x5B3...eddC4 to: Variables.doSomething() 0x86c...e8497 value: 0 wei data: 0x826...92679 logs: 0
hash: 0x47b...4bf49
call to Variables.blockNumber

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Variables.blockNumber() data: 0x57e...871e7
```

PROGRAM 5.1

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract SimpleStorage {
    // State variable to store a number
    uint public num;
    bool public b = true;
    // You need to send a transaction to write to a state variable.
    function set(uint _num) public {
        num = _num;
    }

    // You can read from a state variable without sending a transaction.
    function get() public view returns (uint) {
        return num;
    }
    function get_b() public view returns (bool){
        return b;
    }
}
```

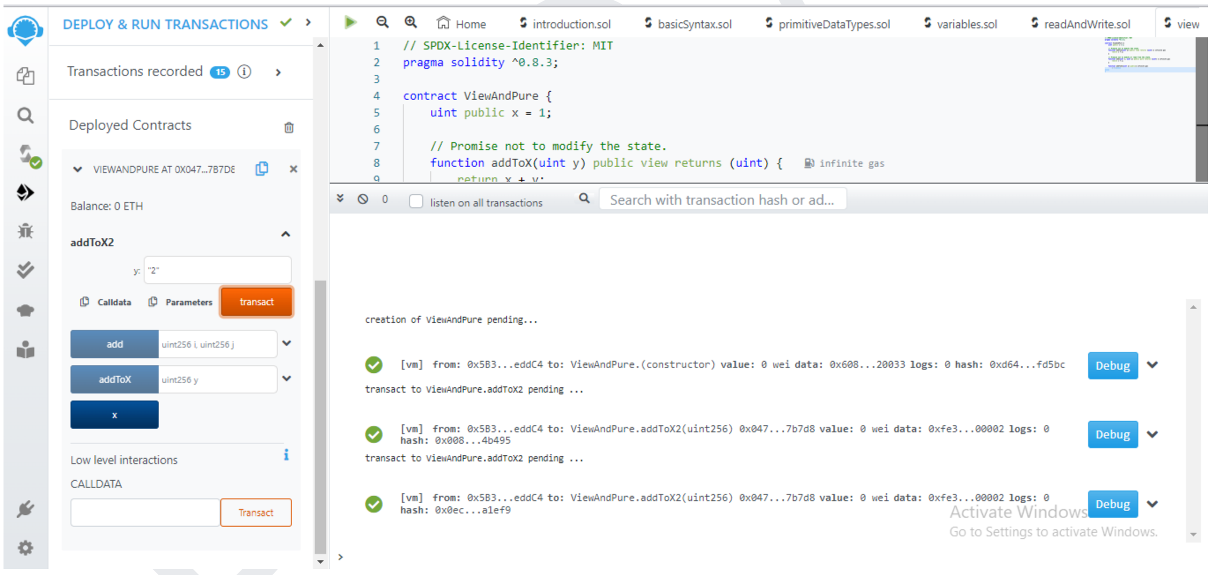


```
    function set(uint _num) public {    22542 gas
        num = _num;
    }

    // You can read from a state variable without sending a transaction.
    function get() public view returns (uint) {    2481 gas
        return num;
    }
```

```
creation of SimpleStorage pending...

[vm]  from: 0x5B3...eddC4 to: SimpleStorage.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0x83f...b620f
transact to SimpleStorage.set pending ...

[vm]  from: 0x5B3...eddC4 to: SimpleStorage.set(uint256) 0x26a...92601 value: 0 wei data: 0x60f...00017 logs: 0
hash: 0x5f0...11dde
call to SimpleStorage.b

CALL  [call]  from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: SimpleStorage.b() data: 0x4df...7e3d0
```

PROGRAM 5.2

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract ViewAndPure {
    uint public x = 1;
    // Promise not to modify the state.
    function addToX(uint y) public view returns (uint) {
        return x + y;
    }

    // Promise not to modify or read from the state.
    function add(uint i, uint j) public pure returns (uint) {
        return i + j;
    }

    function addToX2(uint y) public{
        x = x + y;
    }
}
```



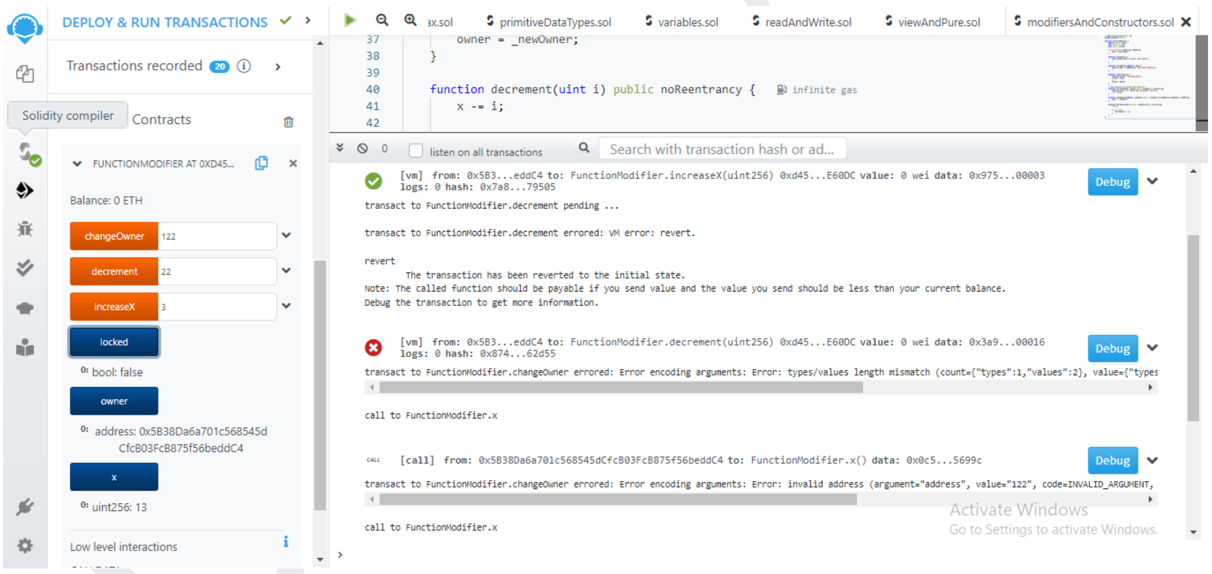
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract ViewAndPure {
    uint public x = 1;

    // Promise not to modify the state.
    function addToX(uint y) public view returns (uint) {    infinite gas
        return x + y;
```

```
creation of ViewAndPure pending...

[vm]  from: 0x5B3...eddC4 to: ViewAndPure.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0xd64...fd5bc
transact to ViewAndPure.addToX2 pending ...

[vm]  from: 0x5B3...eddC4 to: ViewAndPure.addToX2(uint256) 0x047...7b7d8 value: 0 wei data: 0xfe3...00002 logs: 0
hash: 0x008...4b495
transact to ViewAndPure.addToX2 pending ...

[vm]  from: 0x5B3...eddC4 to: ViewAndPure.addToX2(uint256) 0x047...7b7d8 value: 0 wei data: 0xfe3...00002 logs: 0
hash: 0x0ec...a1ef9
```

PROGRAM 5.3

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.3;

contract FunctionModifier {
    address public owner;
```

```
uint public x = 10;
bool public locked;

constructor() {
   owner = msg.sender;
}

modifier onlyOwner() {
   require(msg.sender == owner, "Not owner");
   _;
}

modifier validAddress(address _addr) {
   require(_addr != address(0), "Not valid address");
   _;
}

modifier noReentrancy() {
   require(!locked, "No reentrancy");
   locked = true;
   _;
   locked = false;
}

// New function to increase the value of x
function increaseX(uint _value) public onlyOwner {
   require(_value > 0, "Value must be greater than 0");
   x += _value;
}

function changeOwner(address _newOwner) public onlyOwner validAddress(_newOwner) {
        owner = _newOwner;
    }

    function decrement(uint i) public noReentrancy {
        x -= i;

        if (i > 1) {
            decrement(i - 1);
        }
    }
}
```

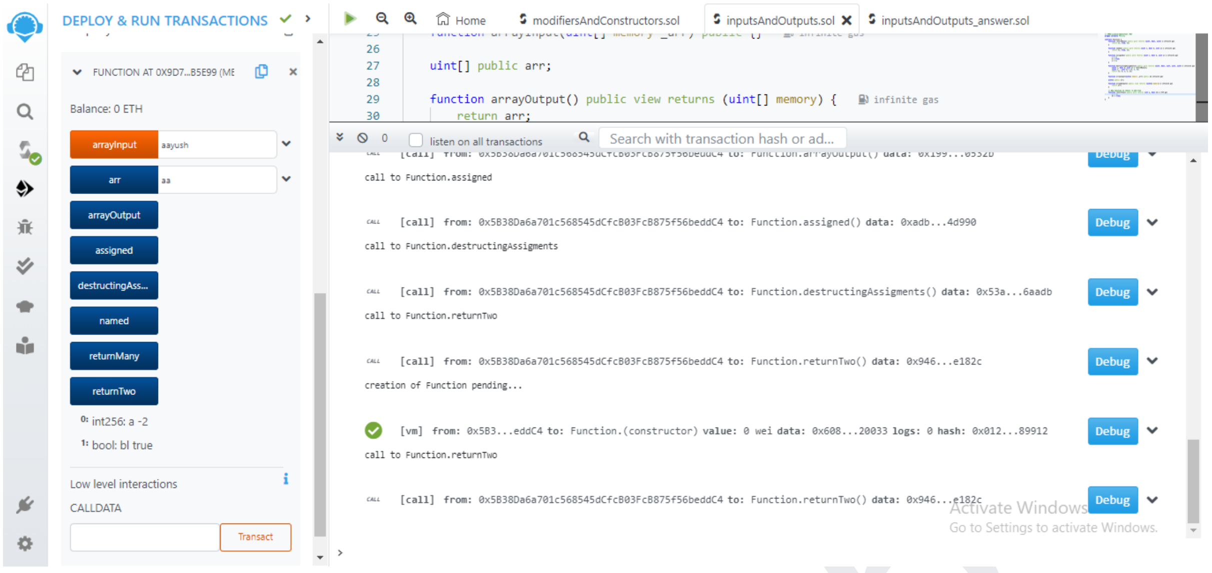


```
        owner = _newOwner;
    }

    function decrement(uint i) public noReentrancy {    infinite gas
        x -= i;
```

```
[vm]  from: 0x5B3...eddC4 to: FunctionModifier.increaseX(uint256) 0xd45...E60DC value: 0 wei data: 0x975...00003
logs: 0 hash: 0x7a8...79505
transact to FunctionModifier.decrement pending ...

transact to FunctionModifier.decrement errored: VM error: revert.

revert
        The transaction has been reverted to the initial state.
Note: The called function should be payable if you send value and the value you send should be less than your current balance.
Debug the transaction to get more information.

[vm]  from: 0x5B3...eddC4 to: FunctionModifier.decrement(uint256) 0xd45...E60DC value: 0 wei data: 0x3a9...00016
logs: 0 hash: 0x874...62d55
transact to FunctionModifier.changeOwner errored: Error encoding arguments: Error: types/values length mismatch (count={"types":1,"values":2}, value={"types

call to FunctionModifier.x

CALL  [call]  from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: FunctionModifier.x() data: 0x0c5...5699c
transact to FunctionModifier.changeOwner errored: Error encoding arguments: Error: invalid address (argument="address", value="122", code=INVALID_ARGUMENT,

call to FunctionModifier.x
```

PROGRAM 5.4

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract Function {
    function returnMany() public pure returns (uint, bool, uint) {
        return (1, true, 2);
    }

    function named() public pure returns (uint x, bool b, uint y) {
        return (1, true, 2);
    }

    function assigned() public pure returns (uint x, bool b, uint y) {
        x = 1;
        b = true;
        y = 2;
    }

    function destructingAssigments() public pure returns (uint, bool, uint, uint, uint) {
        (uint i, bool b, uint j) = returnMany();
        (uint x, , uint y) = (4, 5, 6);
        return (i, b, j, x, y);
    }

    function arrayInput(uint[] memory _arr) public {}
    uint[] public arr;

    function arrayOutput() public view returns (uint[] memory) {
        return arr;
    }

    // New function to return -2 and true
    function returnTwo() public pure returns (int a, bool bl) {
        a = -2;
        bl = true;
    }
}
```



Balance: 0 ETH

arrayInput: aayush

arr: aa

0: int256: a -2

1: bool: bl true

CALLDATA

```
uint[] public arr;

function arrayOutput() public view returns (uint[] memory) {    infinite gas
    return arr;
```

listen on all transactions

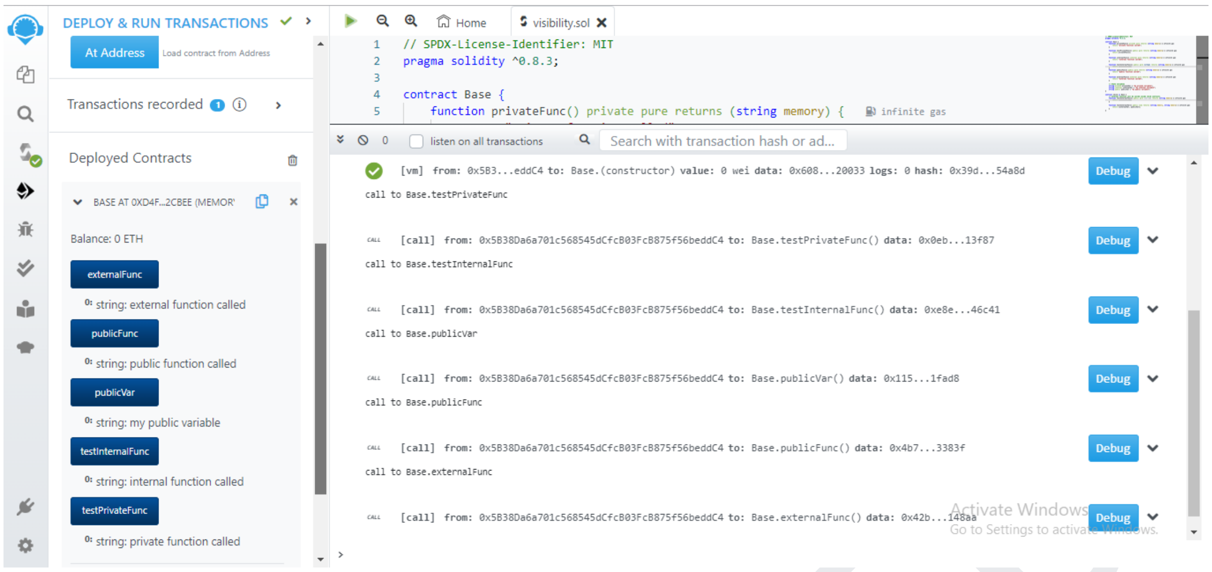
Search with transaction hash or ad...

```
call to Function.assigned

CALL  [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Function.assigned() data: 0xadb...4d990
call to Function.destructingAssigments

CALL  [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Function.destructingAssigments() data: 0x53a...6aadb
call to Function.returnTwo

CALL  [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Function.returnTwo() data: 0x946...e182c
creation of Function pending...

✓  [vm] from: 0x5B3...eddC4 to: Function.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0x012...89912
call to Function.returnTwo

CALL  [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Function.returnTwo() data: 0x946...e182c
```

Activate Windows
Go to Settings to activate Windows.

PROGRAM 6

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.3;


contract Base {

    function privateFunc() private pure returns (string memory) {

        return "private function called";

    }


    function testPrivateFunc() public pure returns (string memory) {

        return privateFunc();

    }


    function internalFunc() internal pure returns (string memory) {

        return "internal function called";

    }


    function testInternalFunc() public pure virtual returns (string memory) {

        return internalFunc();
    }
    function publicFunc() public pure returns (string memory) {
        return "public function called";
    }

    function externalFunc() external pure returns (string memory) {
        return "external function called";
    }

    // State variables
    string private privateVar = "my private variable";
    string internal internalVar = "my internal variable";
    string public publicVar = "my public variable";
}

contract Child is Base {
    // Internal function call be called inside child contracts.
    function testInternalFunc() public pure override returns (string memory) {
        return internalFunc();
    }

    function testInternalVar() public view returns (string memory, string memory) {
        return (internalVar, publicVar);
    }
}
```



0: string: external function called
0: string: public function called
0: string: my public variable
0: string: internal function called
0: string: private function called

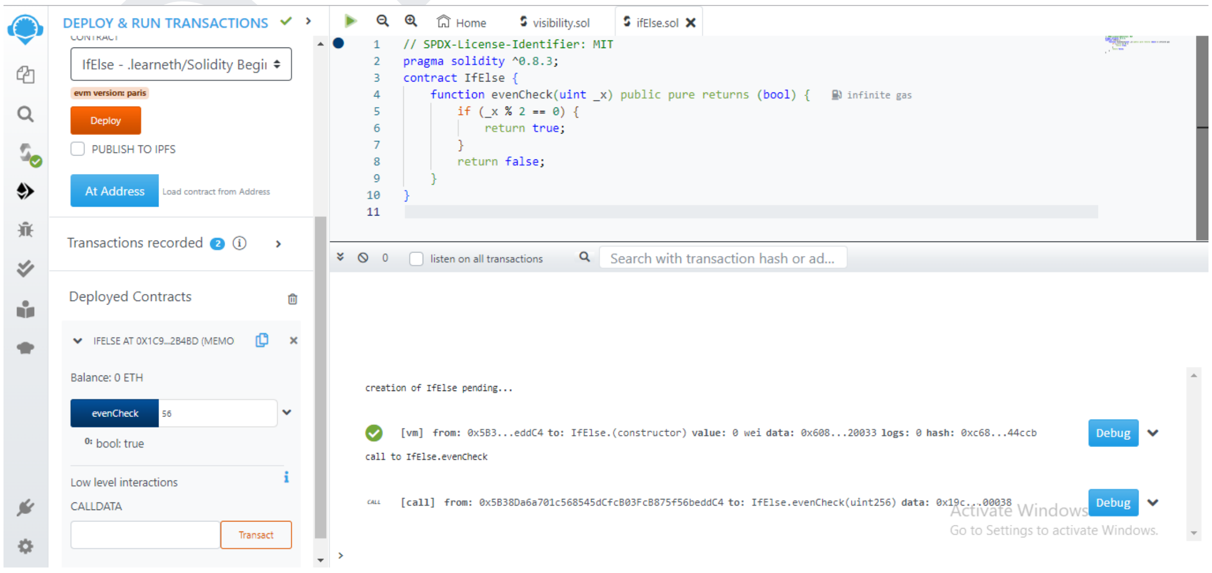
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract Base {
    function privateFunc() private pure returns (string memory) {    infinite gas
```

```
[vm] from: 0x5B3...eddC4 to: Base.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0x39d...54a8d
call to Base.testPrivateFunc

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Base.testPrivateFunc() data: 0x0eb...13f87
call to Base.testInternalFunc

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Base.testInternalFunc() data: 0xe8e...46c41
call to Base.publicVar

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Base.publicVar() data: 0x115...1fad8
call to Base.publicFunc

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Base.publicFunc() data: 0x4b7...3383f
call to Base.externalFunc

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Base.externalFunc() data: 0x42b...148aa
```

Activate Windows
Go to Settings to activate Windows.

PROGRAM 7.1

```
pragma solidity ^0.8.3;

contract IfElse {

  function evenCheck(uint _x) public pure returns (bool) {

    if (_x % 2 == 0) {

      return true;

    }

    return false;

  }

}
```



evenCheck 56

0: bool: true

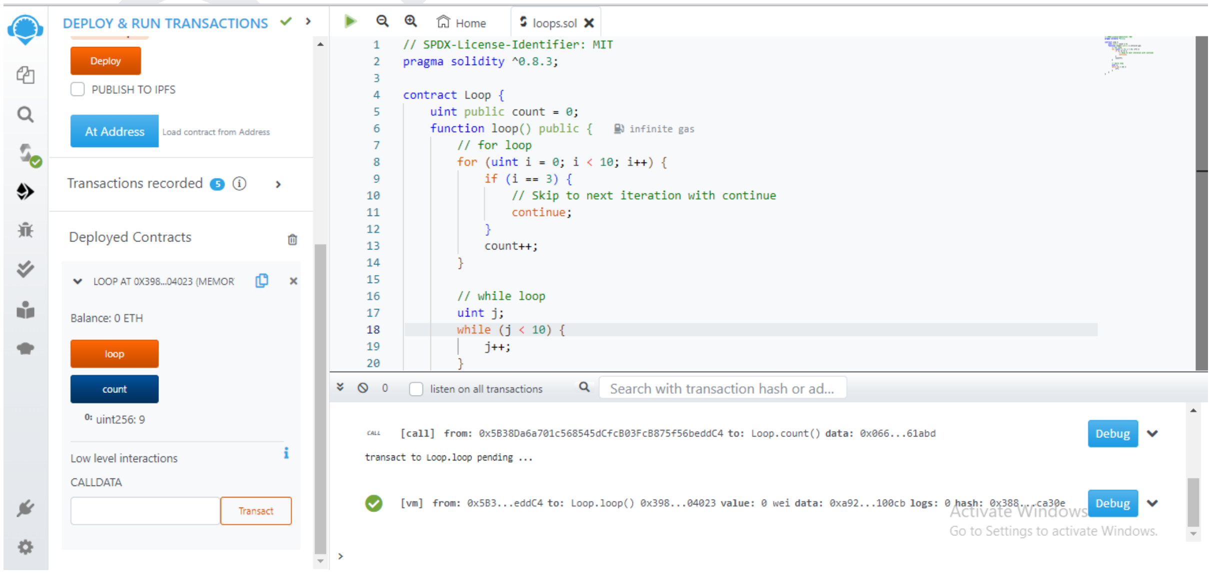
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;
contract IfElse {
    function evenCheck(uint _x) public pure returns (bool) {    infinite gas
        if (_x % 2 == 0) {
            return true;
        }
        return false;
    }
}
```

```
creation of IfElse pending...

[vm] from: 0x5B3...eddC4 to: IfElse.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0xc68...44ccb
call to IfElse.evenCheck

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: IfElse.evenCheck(uint256) data: 0x19c...00038
```

Activate Windows
Go to Settings to activate Windows.

PROGRAM 7.2

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;
contract Loop {
    uint public count = 0;
    function loop() public {
        // for loop
        for (uint i = 0; i < 10; i++) {
            if (i == 3) {
                // Skip to next iteration with continue
                continue;
            }
            count++;
        }
        // while loop
        uint j;
        while (j < 10) {
            j++;
        }
    }
}
```



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract Loop {
    uint public count = 0;
    function loop() public {
        // for loop
        for (uint i = 0; i < 10; i++) {
            if (i == 3) {
                // Skip to next iteration with continue
                continue;
            }
            count++;
        }

        // while loop
        uint j;
        while (j < 10) {
            j++;
        }
```

```
CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Loop.count() data: 0x066...61abd
transact to Loop.loop pending ...

[vm] from: 0x5B3...eddC4 to: Loop.loop() 0x398...04023 value: 0 wei data: 0xa92...100cb logs: 0 hash: 0x388...ca30e
```