

INDEX

List of Practical:		Page no	Sign
1.	Write the following programs for Blockchain in Python:		
a.	A simple client class that generates the private and public keys by using the built- in Python RSA algorithm and test it.		
b.	A transaction class to send and receive money and test it.		
c.	Create multiple transactions and display them.		
d.	Create a blockchain, a genesis block and execute it.		
e.	Create a mining function and test it.		
f.	Add blocks to the miner and dump the blockchain.		
2.	Implement and demonstrate the use of the following in Solidity:		
a.	Variable, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs, Mappings, Conversions, Ether Units, Special Variables.		
b.	Functions, Function Modifiers, View functions, Pure Functions, Fallback Function, Function Overloading, Mathematical functions, Cryptographic functions.		
3.	Implement and demonstrate the use of the following in Solidity:		
a.	Withdrawal Pattern, Restricted Access.		
b.	Contracts, Inheritance, Constructors, Abstract Contracts, Interfaces.		
c.	Libraries, Assembly, Events, Error handling.		
4.	Install hyperledger fabric and composer. Deploy and execute the application.		
5.	Demonstrate the running of the blockchain node.		
6.	Demonstrate the use of Bitcoin Core API.		
7.	Create your own blockchain and demonstrate its use.		

Practical No: 1

Aim: Write the following programs for Blockchain in Python:

1a) A simple client class that generates the private and public keys by using the built in Python RSA algorithm and test it.

Code:

```
import hashlib
import random
import string
import json
import binascii
import numpy as np
import pandas as pd
import pylab as pl
import logging
import datetime
import collections
import Crypto
import Crypto.Random
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
import binascii

class Client:
    def __init__(self):
        random = Crypto.Random.new().read
        self._private_key = RSA.generate(1024, random)
        self._public_key = self._private_key.publickey()
        self._signer = PKCS1_v1_5.new(self._private_key)
    @property
    def identity(self):
        return binascii.hexlify(self._public_key.exportKey(format='DER')).decode('ascii')
Dinesh = Client()
print (Dinesh.identity)
```

Output:

```
30819f300d06092a864886f70d0101050003818d0030818902818100eea39d3e40f737cff2050d40515a4833c80cd9073d88a3629aef45ee77793c78197eb0f8bbf0688c0672e22ee74e691bd53668c
```

1b) A transaction class to send and receive money and test it.

Code:

```
class Transaction:
    def __init__(self, sender, recipient, value):
        self.sender = sender
        self.recipient = recipient
        self.value = value
        self.time = datetime.datetime.now()
    def to_dict(self):
        if self.sender == "Genesis":
            identity = "Genesis"
        else:
            identity = self.sender.identity

        return collections.OrderedDict({'sender': identity, 'recipient': self.recipient, 'value': self.value, 'time' : self.time})
    def sign_transaction(self):
        private_key = self.sender._private_key
        signer = PKCS1_v1_5.new(private_key)
        h = SHA.new(str(self.to_dict()).encode('utf8'))
        return binascii.hexlify(signer.sign(h)).decode('ascii')
Dinesh = Client()
Ramesh = Client()

t = Transaction(Dinesh, Ramesh.identity, 5.0)

signature = t.sign_transaction()
print (signature)
```

Output:

```
0f18c12abcf2904299162ce5c39689df19a9bf583689ec58182cac42617da5b1282d467b6a92ffd358f33052abbeefdf502d1bbfcfcde293962e8ecec7b9f6a249df1bfcacf537a19404913e73079fc58c1f
```

1c) Create Multiple Transaction and display them.

Code:

```
def display_transaction(transaction):
    dict = transaction.to_dict()
    print ("sender: " + dict['sender'])
    print ('-----')
    print ("recipient: " + dict['recipient'])
    print ('-----')
    print ("value: " + str(dict['value']))
    print ('-----')
    print ("time: " + str(dict['time']))
    print ('-----')

transactions = []

Dinesh = Client()
Ramesh = Client()
Seema = Client()
Vijay = Client()

t1 = Transaction(Dinesh, Ramesh.identity, 15.0)
t1.sign_transaction()
transactions.append(t1)

t2 = Transaction(Dinesh, Seema.identity, 6.0)
t2.sign_transaction()
transactions.append(t2)

t3 = Transaction(Ramesh, Vijay.identity, 2.0)
t3.sign_transaction()
transactions.append(t3)

t4 = Transaction(Seema, Ramesh.identity, 4.0)
t4.sign_transaction()
transactions.append(t4)

for transaction in transactions:
    display_transaction (transaction)
    print ('-----')
```

Output:

```
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100c26c34efd785e255a351885faa617e22dbe4b8d1fdac6f833f00172431f5de6eecfe23c9fdcc14fd4de2dc947e11435fb572d39709f057b3039
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100d31bde02b9759d166f6514250948351f541d716b1ef5d7f3d931de74527050671e57bc3f323940186b105c379f8cd6c28935e43ba5752434a
-----
value: 15.0
-----
time: 2022-07-30 17:35:06.809777
-----
-----
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100c26c34efd785e255a351885faa617e22dbe4b8d1fdac6f833f00172431f5de6eecfe23c9fdcc14fd4de2dc947e11435fb572d39709f057b3039
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100a72cfb527dff9615a33eaf34b0cdf26f68d5d604676786c77ea188380e8412420aee1cde5b6dc9e157b31874d7da7bfb40c5c835a0999e2b268
-----
value: 6.0
-----
time: 2022-07-30 17:35:06.811866
-----
-----
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100d31bde02b9759d166f6514250948351f541d716b1ef5d7f3d931de74527050671e57bc3f323940186b105c379f8cd6c28935e43ba5752434a814
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100c7e4e5f5f46085be74cd9c2f0ee1a79942c8a7b10c103a72b542c3246adf0c14f7a3032f5d431942062f5747e2369e6894e51cfbacfb5d578
-----
value: 2.0
-----
time: 2022-07-30 17:35:06.814247
-----
-----
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100a72cfb527dff9615a33eaf34b0cdf26f68d5d604676786c77ea188380e8412420aee1cde5b6dc9e157b31874d7da7bfb40c5c835a0999e2b268
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100d31bde02b9759d166f6514250948351f541d716b1ef5d7f3d931de74527050671e57bc3f323940186b105c379f8cd6c28935e43ba5752434a
-----
value: 4.0
-----
time: 2022-07-30 17:35:06.816476
-----
```

1d) Create a blockchain, a genesis block and execute it.

Code:

```
class Block:
    def __init__(self):
        self.verified_transactions = []
        self.previous_block_hash = ""
        self.Nonce = ""

last_block_hash = ""

Dinesh = Client()

t0 = Transaction
("Genesis",
Dinesh.identity,
500.0
)

block0 = Block()

block0.previous_block_hash = None
Nonce = None

block0.verified_transactions.append(t0)

digest = hash(block0)
last_block_hash = digest

TPCoins = []

def dump_blockchain(self):
    print("Number of blocks in the chain: " + str(len(self)))
    for x in range(len(TPCoins)):
        block_temp = TPCoins[x]
        print("block # " + str(x))
        for transaction in block_temp.verified_transactions:
            display_transaction(transaction)
        print('-----')
        print('=====')  
TPCoins.append(block0)
dump_blockchain(TPCoins)
```

Output:

```
Number of blocks in the chain: 1
block # 0
sender: Genesis
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100a4a79ac0645d8483c7c97fb6b67e1bf90f17d87d1f
-----
value: 500.0
-----
time: 2022-07-30 17:50:16.288802
-----
=====
=====
```

1e) Create a mining function and test it.

Code:

```
def sha256(message):
    return hashlib.sha256(message.encode('ascii')).hexdigest()

def mine(message, difficulty=1):
    assert difficulty >= 1
    prefix = '1' * difficulty
    for i in range(1000):
        digest = sha256(str(hash(message)) + str(i))
        if digest.startswith(prefix):
            print ("after " + str(i) + " iterations found nonce: " + digest)
    return digest

mine ("test message", 2)
```

Output:

```
after 238 iterations found nonce: 1124711db6185591392c6a06c24a3c2ebbaeca647fb8374824f939ada27c09e9
after 353 iterations found nonce: 11e0a4b57bb76496ecc6ab5a3c126165bc9dbba80f664e7e192a422360c3884
after 419 iterations found nonce: 11280dfd9ab05b3dbfa869990153732941408faa4a3b0832819b161f52321c08
after 511 iterations found nonce: 1150944bd7ea429acd052da390b148f27eb881686e0f8bfcd77f833af878e2e
after 822 iterations found nonce: 110e61d41d94b48f6dfcb6f43f9ceafe2ab7168456dab088e05126d43d0366ef
after 924 iterations found nonce: 11147bf32bafeee4505b5cc40c3b227366d8a41ab3bd740e437c74400b7a8127
'6d80c4cf3cd4fb49aad25dec696c3a48e17974ecfb1441e2128517f0152b2'
```

1f) Add blocks to the miner and dump the blockchain

Code:

```
last_transaction_index = 0

block = Block()
for i in range(3):
    temp_transaction = transactions[last_transaction_index]
    # validate transaction
    # if valid
    block.verified_transactions.append (temp_transaction)
    last_transaction_index += 1

    block.previous_block_hash = last_block_hash
    block.Nonce = mine (block, 2)
    digest = hash (block)
    TPCoins.append (block)
    last_block_hash = digest

# Miner 2 adds a block
block = Block()

for i in range(3):
    temp_transaction = transactions[last_transaction_index]
    # validate transaction
    # if valid
    block.verified_transactions.append (temp_transaction)
    last_transaction_index += 1
block.previous_block_hash = last_block_hash
block.Nonce = mine (block, 2)
digest = hash (block)
TPCoins.append (block)
last_block_hash = digest
# Miner 3 adds a block
block = Block()

for i in range(3):
    temp_transaction = transactions[last_transaction_index]
    #display_transaction (temp_transaction)
    # validate transaction
    # if valid
    block.verified_transactions.append (temp_transaction)
    last_transaction_index += 1
```

```
block.previous_block_hash = last_block_hash
block.Nonce = mine (block, 2)
digest = hash (block)

TPCoins.append (block)
last_block_hash = digest

dump_blockchain(TPCoins)
```

Output:

```
after 308 iterations found nonce: 111b1715b007a9920e76b241e9a4dde15a50b00faeed353bf9279311deb59804
after 398 iterations found nonce: 11a65418aca0201df85c2eeacabbcf0624092abb64bf6e159ccf5539cbb1cfe1
after 527 iterations found nonce: 11310267f14590e45be2375fe94e881c65f86020e3fcf7844a849241783680a7
after 718 iterations found nonce: 117ba8286d575afe73007e6f588b6f541580bec444d5165a3f71d9e4b623f3fa
after 308 iterations found nonce: 111b1715b007a9920e76b241e9a4dde15a50b00faeed353bf9279311deb59804
after 398 iterations found nonce: 11a65418aca0201df85c2eeacabbcf0624092abb64bf6e159ccf5539cbb1cfe1
after 527 iterations found nonce: 11310267f14590e45be2375fe94e881c65f86020e3fcf7844a849241783680a7
after 718 iterations found nonce: 117ba8286d575afe73007e6f588b6f541580bec444d5165a3f71d9e4b623f3fa
after 308 iterations found nonce: 111b1715b007a9920e76b241e9a4dde15a50b00faeed353bf9279311deb59804
after 398 iterations found nonce: 11a65418aca0201df85c2eeacabbcf0624092abb64bf6e159ccf5539cbb1cfe1
after 527 iterations found nonce: 11310267f14590e45be2375fe94e881c65f86020e3fcf7844a849241783680a7
after 718 iterations found nonce: 117ba8286d575afe73007e6f588b6f541580bec444d5165a3f71d9e4b623f3fa
```

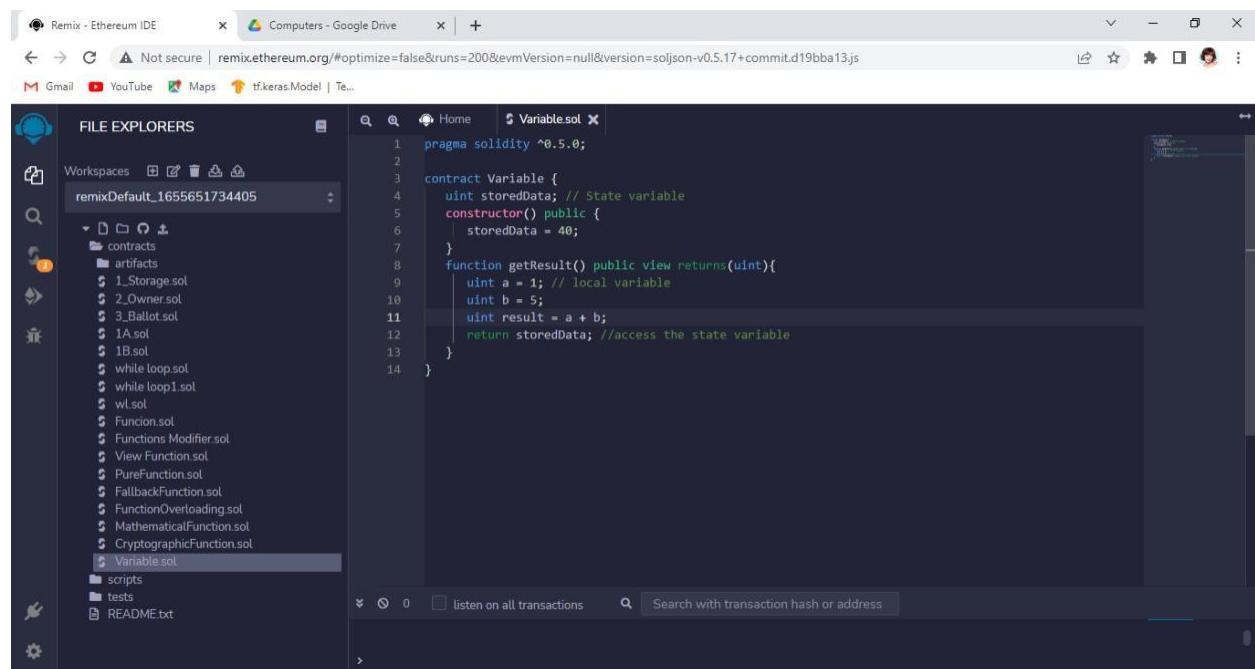
Practical No: 2

Aim: Implement and demonstrate the use of the following in Solidity

2a) Variable, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs, Mappings, Conversions, Ether Units, Special Variables. Variable

Variable

Code:



The screenshot shows the Remix Ethereum IDE interface. The top bar displays the title "Remix - Ethereum IDE" and the URL "remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js". The left sidebar is titled "FILE EXPLORERS" and shows a workspace named "remixDefault_1655651734405". Inside this workspace, there is a folder named "contracts" containing several Solidity files: 1.Storage.sol, 2.Owner.sol, 3.Ballot.sol, 1A.sol, 1B.sol, while.loop.sol, while.loop1.sol, wl.sol, Function.sol, Functions.Modifier.sol, View.Function.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, and Variable.sol. The "Variable.sol" file is currently selected and open in the main editor area. The code in the editor is as follows:

```
pragma solidity ^0.5.0;

contract Variable {
    uint storedData; // State variable
    constructor() public {
        storedData = 40;
    }
    function getResult() public view returns(uint){
        uint a = 1; // local variable
        uint b = 5;
        uint result = a + b;
        return storedData; //access the state variable
    }
}
```

Output:

The screenshot shows the Ethereum IDE (Remix) interface. The top bar displays the title "Remix - Ethereum IDE" and the URL "remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js". The sidebar on the left has sections for "DEPLOY & RUN TRANSACTIONS" and "Deployed Contracts". The main area shows a Solidity code editor with the file "Variable.sol" containing the following code:

```
1 pragma solidity ^0.5.0;
2
3 contract Variable {
4     uint storedData; // State variable
5     constructor() public {
6         storedData = 40;
7     }
8     function getResult() public view returns(uint){
9         uint a = 1; // local variable
10        uint b = 5;
11        uint result = a + b;
12        return storedData; //access the state variable
13    }
14 }
```

Below the code editor, under "Deployed Contracts", there is a section for "VARIABLE AT 0XEF9...10EBF (MEMO)". It shows a button labeled "getResult" which, when clicked, displays the output "o: uint256: 40". There is also a "Low level interactions" section with a "CALLDATA" tab and a "Tranact" button.

Operators:

Arithmetic Operator

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several Solidity files: wl.sol, Funcion.sol, Functions Modifier.sol, View Function.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, Variable.sol, BitwiseOperator.sol, 1A.sol, AssignmentOperator.sol, ArithmeticOperator.sol, WithdrawalPattern.sol, RestrictedAccess.sol, Contracts.sol, Inheritance.sol, Constructors.sol, AbstractConstructor.sol, Interfaces.sol, Libraries.sol, Assembly.sol, Events.sol, and ErrorHandling.sol. The file "ArithmeticOperator.sol" is currently selected and shown in the main code editor area. The code defines a contract named "ArithmeticOperator" with variables a (uint16) and b (uint16), and functions sum, diff, mul, div, mod, dec, and inc. The right panel shows the transaction history with two entries: a constructor call from 0x5B3...edd4 to the contract, and a creation of the contract itself.

```
pragma solidity ^0.5.0;

contract ArithmeticOperator {
    uint16 public a = 50;
    uint16 public b = 20;

    uint public sum = a + b;
    uint public diff = a - b;
    uint public mul = a * b;

    uint public div = a / b;
    uint public mod = a % b;

    uint public dec = --b;
    uint public inc = ++a;
}
```

Output:

The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" sidebar open. This sidebar lists various function names (a, b, dec, diff, div, mod, mul, inc, sum) each associated with a placeholder value (e.g., o: uint16: 51 for 'a'). The main code editor area is identical to the previous screenshot, showing the "ArithmeticOperator.sol" code. The right panel shows the transaction history with calls to the contract's functions: a call to .mul() and a call to .sum().

```
pragma solidity ^0.5.0;

contract ArithmeticOperator {
    uint16 public a = 50;
    uint16 public b = 20;

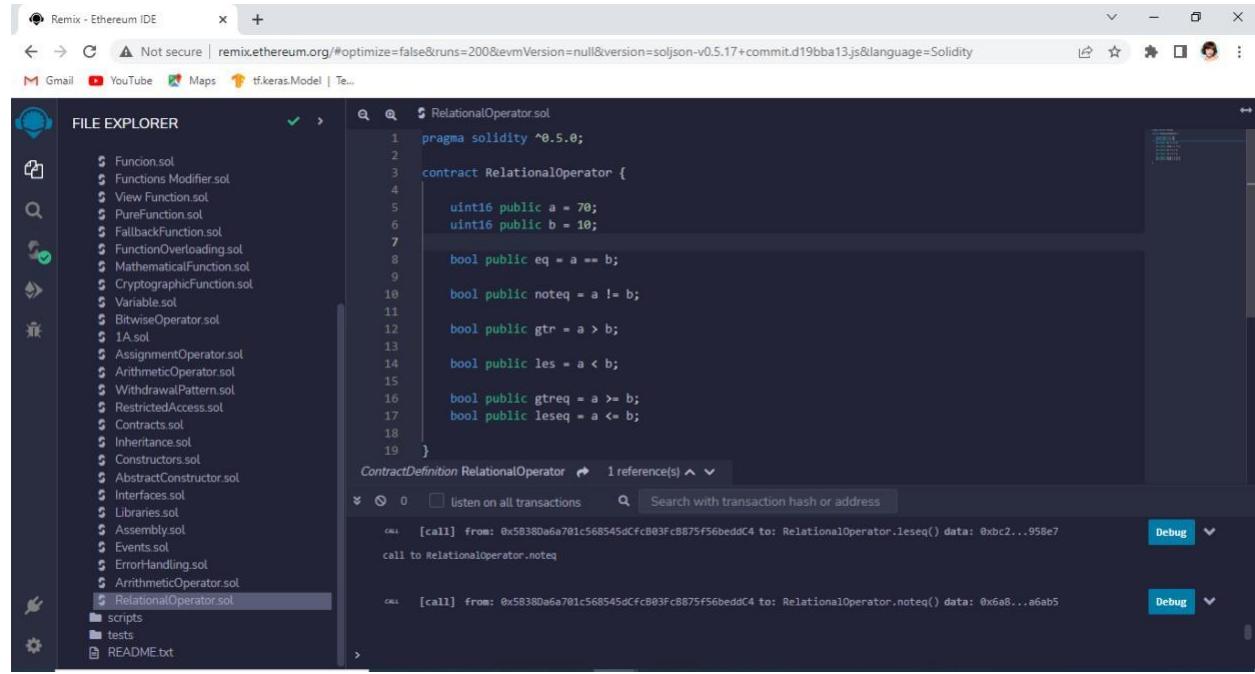
    uint public sum = a + b;
    uint public diff = a - b;
    uint public mul = a * b;

    uint public div = a / b;
    uint public mod = a % b;

    uint public dec = --b;
    uint public inc = ++a;
}
```

Relational Operator

Code:



The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several Solidity files: Funcion.sol, Functions Modifier.sol, View Function.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, Variable.sol, BitwiseOperator.sol, 1A.sol, AssignmentOperator.sol, ArithmeticOperator.sol, WithdrawalPattern.sol, RestrictedAccess.sol, Contracts.sol, Inheritance.sol, Constructors.sol, AbstractConstructor.sol, Interfaces.sol, Libraries.sol, Assembly.sol, Events.sol, ErrorHandling.sol, ArithmeticOperator.sol, and RelationalOperator.sol. The "RelationalOperator.sol" file is currently selected. The main editor area displays the following Solidity code:

```
pragma solidity ^0.5.0;

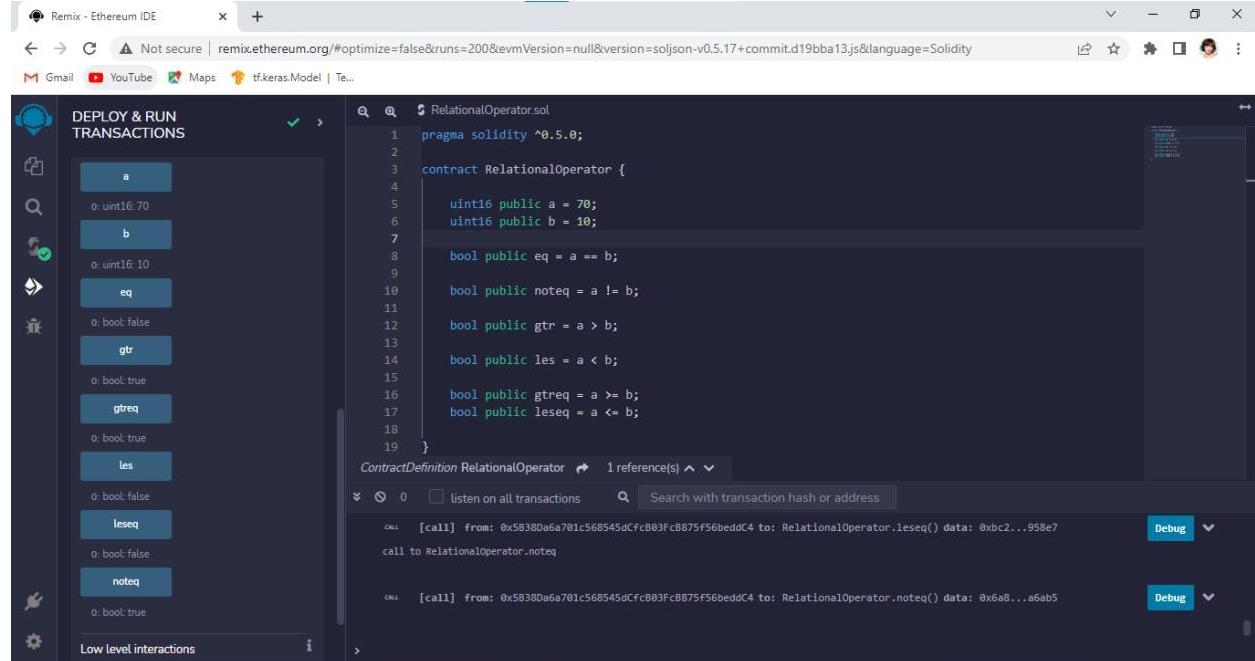
contract RelationalOperator {
    uint16 public a = 70;
    uint16 public b = 10;

    bool public eq = a == b;
    bool public noteq = a != b;
    bool public gtr = a > b;
    bool public les = a < b;
    bool public gtreq = a >= b;
    bool public leseq = a <= b;
}
```

Below the code, the "ContractDefinition RelationalOperator" section shows "1 reference(s)". The transaction history pane at the bottom shows two calls:

- [call] from: 0x58380a6a701c568545dCfcB03Fc8875f56beddC4 to: RelationalOperator.leseq() data: 0xb0c2...958e7
- [call] from: 0x58380a6a701c568545dCfcB03Fc8875f56beddC4 to: RelationalOperator.noteq() data: 0x6a8...a6ab5

Output:



The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" tab selected. On the left, there is a sidebar with buttons for "a", "b", "eq", "gtr", "gtreq", "les", "lesq", and "noteq". Each button has a value below it: "a" is 0:uint16:70, "b" is 0:uint16:10, "eq" is 0:bool:false, "gtr" is 0:bool:true, "gtreq" is 0:bool:true, "les" is 0:bool:false, "lesq" is 0:bool:false, and "noteq" is 0:bool:true. The main editor area shows the same Solidity code as the previous screenshot. The transaction history pane at the bottom shows the same two calls as before:

- [call] from: 0x58380a6a701c568545dCfcB03Fc8875f56beddC4 to: RelationalOperator.leseq() data: 0xb0c2...958e7
- [call] from: 0x58380a6a701c568545dCfcB03Fc8875f56beddC4 to: RelationalOperator.noteq() data: 0x6a8...a6ab5

Logical Operator

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several Solidity files. The main area displays the code for "LogicalOperator.sol". The code defines a contract named "logicalOperator" with three functions: Logic, which takes two bool parameters and returns three bool values; AND operator, which returns true if both inputs are true; OR operator, which returns true if either input is true; and NOT operator, which returns the inverse of the input.

```
pragma solidity ^0.5.0;
// Creating a contract
contract logicalOperator{
    function Logic(
        bool a, bool b) public view returns(
        bool, bool, bool){
        // Logical AND operator
        bool and = a&b;
        // Logical OR operator
        bool or = a||b;
        // Logical NOT operator
        bool not = !a;
        return (and, or, not);
    }
}
```

Output:

The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" sidebar open. It displays a deployed contract named "LOGICALOPERATOR AT 0XB27...0". The "Logic" function has been called with parameters a: true and b: false, returning the values 0: bool: false, 1: bool: true, and 2: bool: false.

Bitwise Operator

Code:

The screenshot shows the Ethereum IDE interface. On the left, the FILE EXPLORER sidebar lists several Solidity files, with `BitwiseOperator.sol` selected. The main area displays the source code for `BitwiseOperator.sol`:

```
pragma solidity ^0.5.0;

contract BitwiseOperator {

    uint16 public a = 20;
    uint16 public b = 50;

    uint16 public and = a & b;
    uint16 public or = a | b;
    uint16 public xor = a ^ b;
    uint16 public leftshift = a << b;
    uint16 public rightshift = a >> b;
    uint16 public not = ~a;
}
```

Output:

The screenshot shows the Ethereum IDE interface with the `DEPLOY & RUN TRANSACTIONS` tab active. A deployment transaction has been made to address `0xae0...9e`. The interface displays the results of calling different bitwise functions on the deployed contract:

- `a`: `o: uint16: 20`
- `and`: `o: uint16: 16`
- `b`: `o: uint16: 50`
- `leftshift`: `o: uint16: 0`
- `not`: `o: uint16: 65515`
- `or`: `o: uint16: 54`
- `rightshift`: `o: uint16: 0`
- `xor`: `o: uint16: 38`

At the bottom, a log message is shown: `[VM] from: 0x583...addC4 to: LogicalOperator.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x5df...62867`. There is also a `Debug` button.

Assignment Operator

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several Solidity files. The file "AssignmentOperator.sol" is currently selected and highlighted in blue. The main editor area displays the Solidity code for the "AssignmentOperator" contract. The code defines a public assignment variable and implements methods to perform arithmetic operations (add, sub, mul, div, mod) on it. A transaction history section at the bottom shows a pending creation transaction and a recent call to the assignment_sub function.

```
pragma solidity ^0.5.0;

contract AssignmentOperator {
    uint16 public assignment = 20;
    uint16 public assignment_add = 50;
    uint16 public assignment_sub = 50;
    uint16 public assignment_mul = 10;
    uint16 public assignment_div = 50;
    uint16 public assignment_mod = 32;

    function getResult() public {
        assignment_add += 10;
        assignment_sub -= 20;
        assignment_mul *= 10;
        assignment_div /= 10;
        assignment_mod %= 20;
        return;
    }
}
```

Output:

The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" sidebar open. It displays the "Deployed Contracts" section, which shows the "ASSIGNMENTOPERATOR AT 0x9c..." address. Below this, a list of functions is shown, each with a corresponding button to interact with the deployed contract. The main editor area shows the same "AssignmentOperator.sol" code as the previous screenshot. A transaction history section at the bottom shows a call to the assignment_sub function.

```
pragma solidity ^0.5.0;

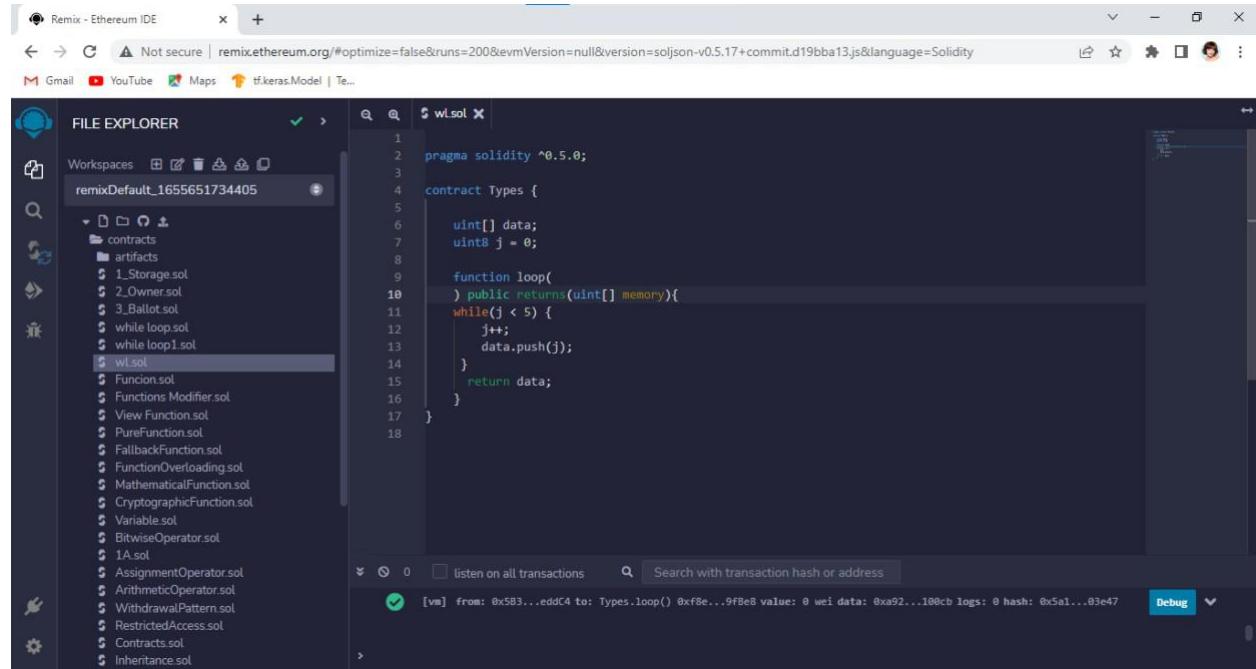
contract AssignmentOperator {
    uint16 public assignment = 20;
    uint16 public assignment_add = 50;
    uint16 public assignment_sub = 50;
    uint16 public assignment_mul = 10;
    uint16 public assignment_div = 50;
    uint16 public assignment_mod = 32;

    function getResult() public {
        assignment_add += 10;
        assignment_sub -= 20;
        assignment_mul *= 10;
        assignment_div /= 10;
        assignment_mod %= 20;
        return;
    }
}
```

Loops

While Loop

Code:



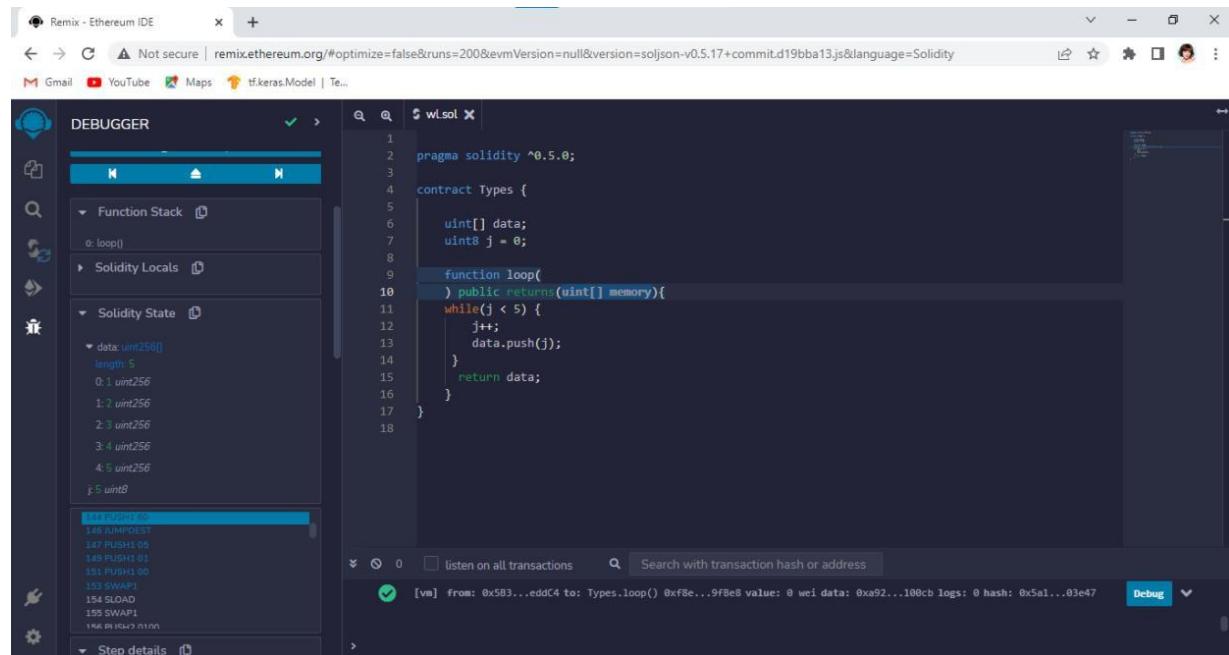
The screenshot shows the Remix Ethereum IDE interface. The left sidebar is the File Explorer, displaying a workspace named "remixDefault_1655651734405" containing several Solidity files like 1_Storage.sol, 2_Owner.sol, 3_Ballot.sol, and others. The central area is a code editor with the file "wl.sol" open. The code defines a contract "Types" with a function "loop" that pushes integers from 0 to 4 into an array "data". The right sidebar shows transaction logs and a "Debug" button.

```
pragma solidity ^0.5.0;

contract Types {
    uint[] data;
    uint8 j = 0;

    function loop()
        public returns(uint[] memory)
    {
        while(j < 5) {
            j++;
            data.push(j);
        }
        return data;
    }
}
```

Output:



The screenshot shows the Remix IDE with the "DEBUGGER" tab selected. The left sidebar displays the Function Stack (with "loop" listed), Solidity Locals, and Solidity State. The Solidity State pane shows an array "data" with length 5, containing values 0, 1, 2, 3, 4. The code editor shows the same "wl.sol" file as before. The bottom pane shows assembly code with steps like JUMPDEST, PUSH1 05, and SWAP1. The "Step details" section at the bottom is expanded.

```
pragma solidity ^0.5.0;

contract Types {
    uint[] data;
    uint8 j = 0;

    function loop()
        public returns(uint[] memory)
    {
        while(j < 5) {
            j++;
            data.push(j);
        }
        return data;
    }
}
```

Dowhile loop

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists various Solidity test files. The main editor window displays the following Solidity code:

```
pragma solidity ^0.5.0;
contract Dowhileloop {
    uint[] data;
    uint8 j = 0;
    function loop()
        public returns(uint[] memory){
        do{
            j++;
            data.push(j);
        }while(j < 8);
        return data;
    }
}
```

The status bar at the bottom indicates a transaction from 0x583...eddc4 to Dowhileloop.loop() with a value of 0 wei, data: 0xa92...100cb, logs: 0, and hash: 0x3d7...90655.

Output:

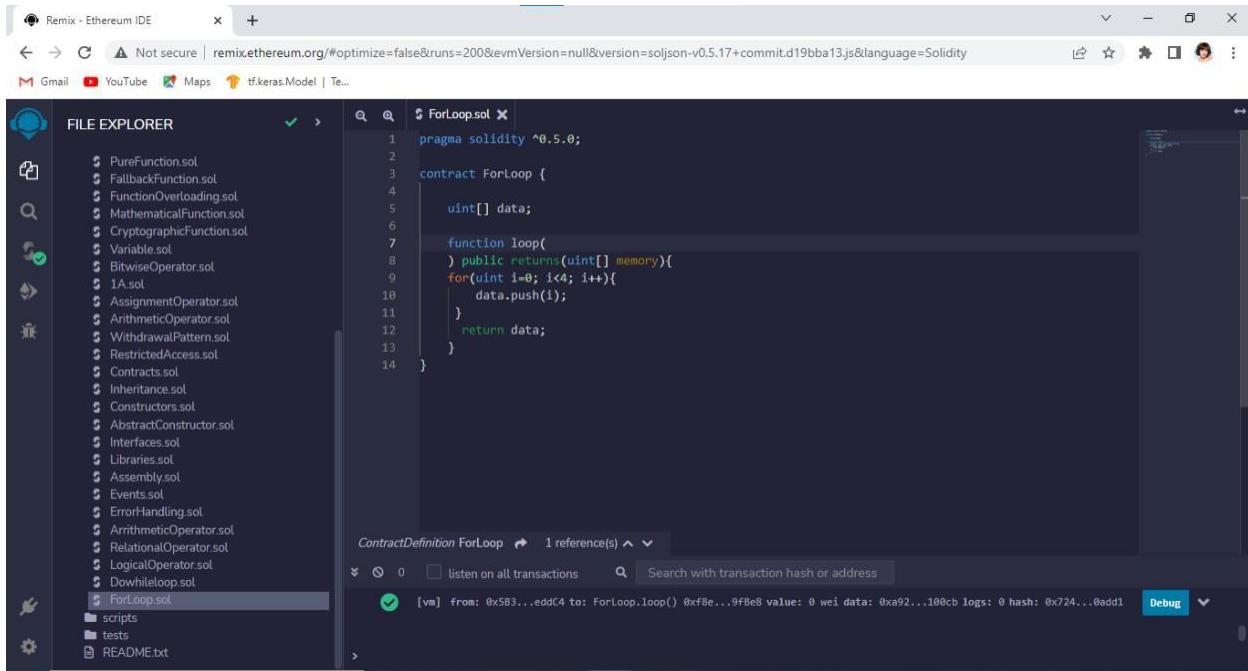
The screenshot shows the Remix Ethereum IDE interface with the "DEBUGGER" tab selected. The left sidebar shows the "Solidity Locals" and "Solidity State" panes. The "Solidity State" pane displays the state of the "data" array:

Index	Type	Value
0:1	uint256	1
1:2	uint256	2
2:3	uint256	3
3:4	uint256	4
4:5	uint256	5
5:6	uint256	6
6:7	uint256	7
7:8	uint256	8

The right pane shows the Solidity code for the "Dowhileloop" contract. The bottom status bar indicates a transaction from 0x583...eddc4 to Dowhileloop.loop() with a value of 0 wei, data: 0xa92...100cb, logs: 0, and hash: 0x3d7...90655.

For loop

Code:

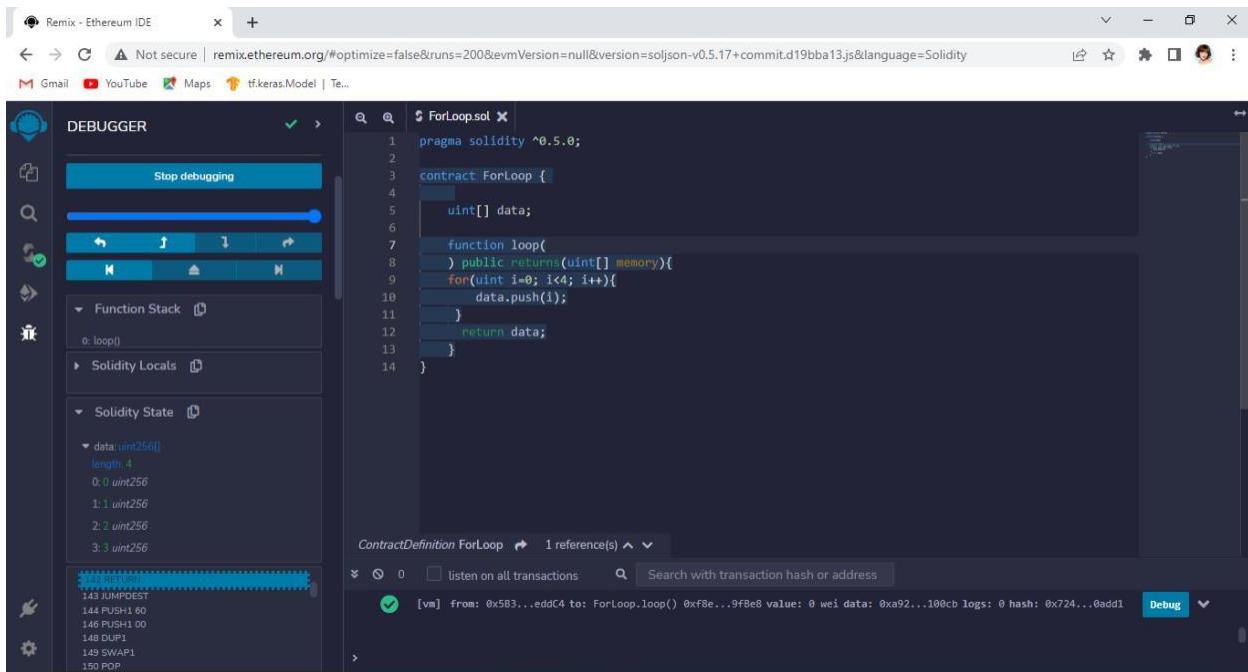


The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists various Solidity files. The main editor window contains the following Solidity code:

```
pragma solidity ^0.5.0;
contract ForLoop {
    uint[] data;
    function loop() public returns(uint[] memory){
        for(uint i=0; i<4; i++){
            data.push(i);
        }
        return data;
    }
}
```

The status bar at the bottom indicates a transaction from address 0x583...eddC4 to the contract, with a value of 0 wei, data of 0xa92...100cb, and a log hash of 0x724...0add1. A "Debug" button is visible.

Output:



The screenshot shows the Remix Ethereum IDE interface with the "DEBUGGER" tab selected. The left sidebar now displays the "Debugger" section, which includes buttons for "Stop debugging", "Step over", "Step into", and "Step out". The "Function Stack" shows a single entry: "0: loop()". The "Solidity Locals" and "Solidity State" sections are also visible. The main editor window shows the same Solidity code as before. The status bar at the bottom shows the transaction details again, and the "Debug" button is highlighted.

Decision making

If statement

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar is the File Explorer, listing various Solidity files. The main editor area contains the following Solidity code:

```
pragma solidity ^0.5.0;
contract IfStatement {
    uint i = 20;
    function decision_making() public returns(bool){
        if(i<20){
            return true;
        }
    }
}
```

The bottom right pane shows transaction logs:

- [vm] from: 0x583...edd4 to: ForLoop.loop() 0xf8e...9f8e8 value: 0 wei data: 0xa92...100cb logs: 0 hash: 0x724...0add1 creation of IfStatement pending...
- [vm] from: 0x583...edd4 to: IfStatement.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x484...7d5c1 transact to IfStatement.decision_making pending ...

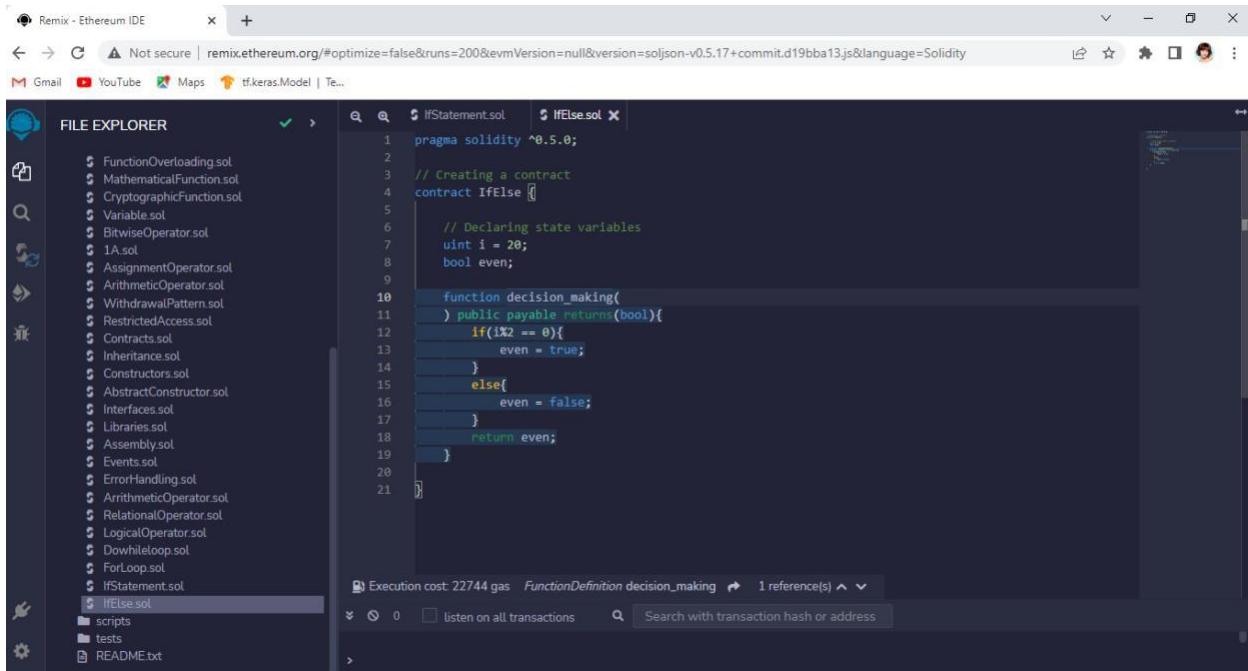
Output:

The screenshot shows the Remix Ethereum IDE interface with the Debugger tab selected. The left sidebar includes a Debugger Configuration section with a checked checkbox for "Use generated sources (Solidity >= v0.7.2)". The main editor area contains the same Solidity code as before. The bottom right pane displays the debugger's state:

- ContractDefinition IfStatement 1 reference(s)
- Function Stack: 0: decision_making()
- Solidity Locals: (empty)
- Solidity State: i: 20 uint256
- CALLS PENDING: 077 JUMPDEST
078 PUSH1 00
080 PUSH1 14
082 PUSH1 00
084 SLOAD
085 LT
086 JUMPI
- input: 0x887...3ef24
- decoded input: ()
- decoded output: {
 "0": "bool: false"
}
- logs: []
- val: 0 wei

If...else statement

Code:



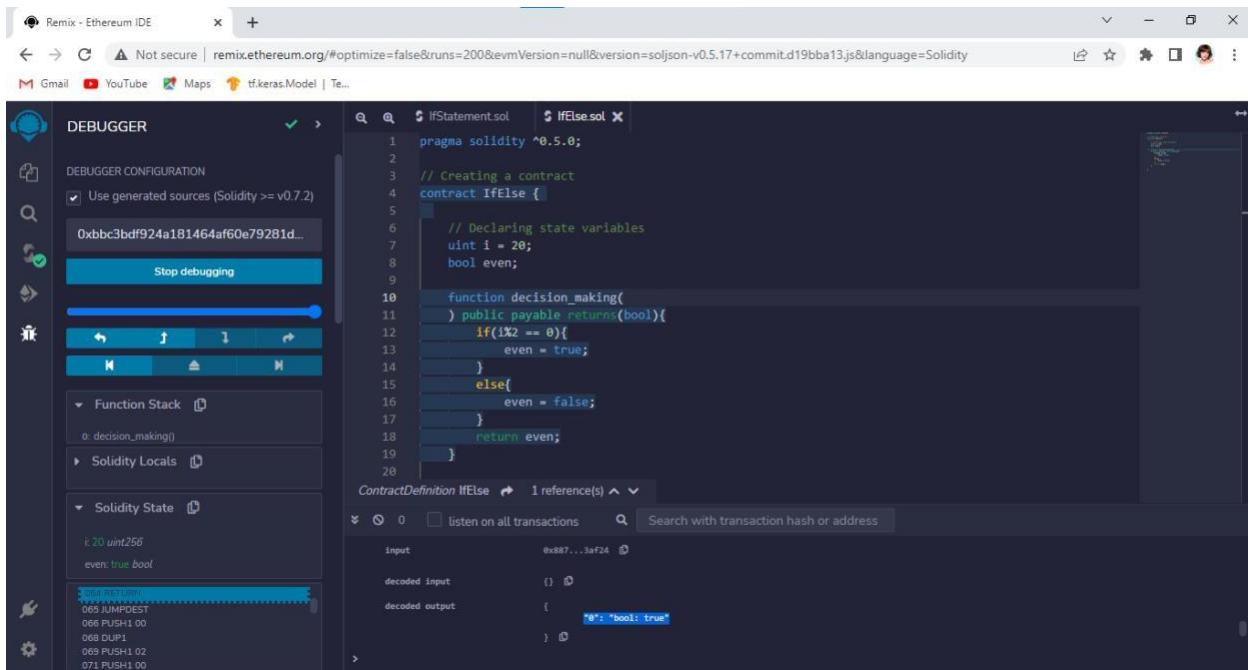
The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled 'FILE EXPLORER' and lists several Solidity files. The main editor window contains the following Solidity code:

```
pragma solidity ^0.5.0;
// Creating a contract
contract IfElse {
    // Declaring state variables
    uint i = 20;
    bool even;

    function decision_making()
        public payable returns(bool){
        if(i%2 == 0){
            even = true;
        }
        else{
            even = false;
        }
        return even;
    }
}
```

The status bar at the bottom indicates an execution cost of 22744 gas for the `decision_making` function.

Output:

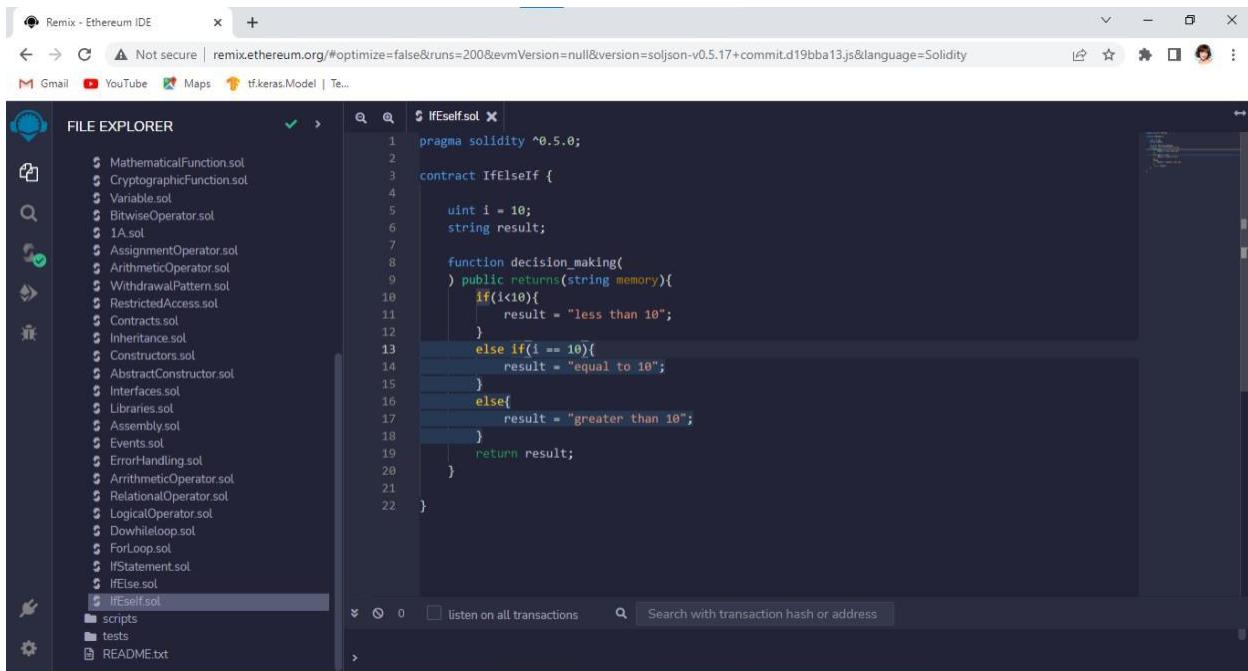


The screenshot shows the Remix IDE with the 'DEBUGGER' tab selected. The left sidebar has a 'DEBUGGER CONFIGURATION' section with a checked checkbox for 'Use generated sources (Solidity >= v0.7.2)'. The main editor window shows the same Solidity code as before. The debugger interface includes a 'Function Stack' showing a call to `decision_making()`, and a 'Solidity State' pane showing the variable `i: 20 uint256` and `even: true bool`. The bottom pane displays the assembly code for the function:

```
0x00 JUMPDEST
0x06 PUSH1 00
0x08 DUP1
0x09 PUSH1 02
0x0A PUSH1 00
```

If...else if...else statement

Code:

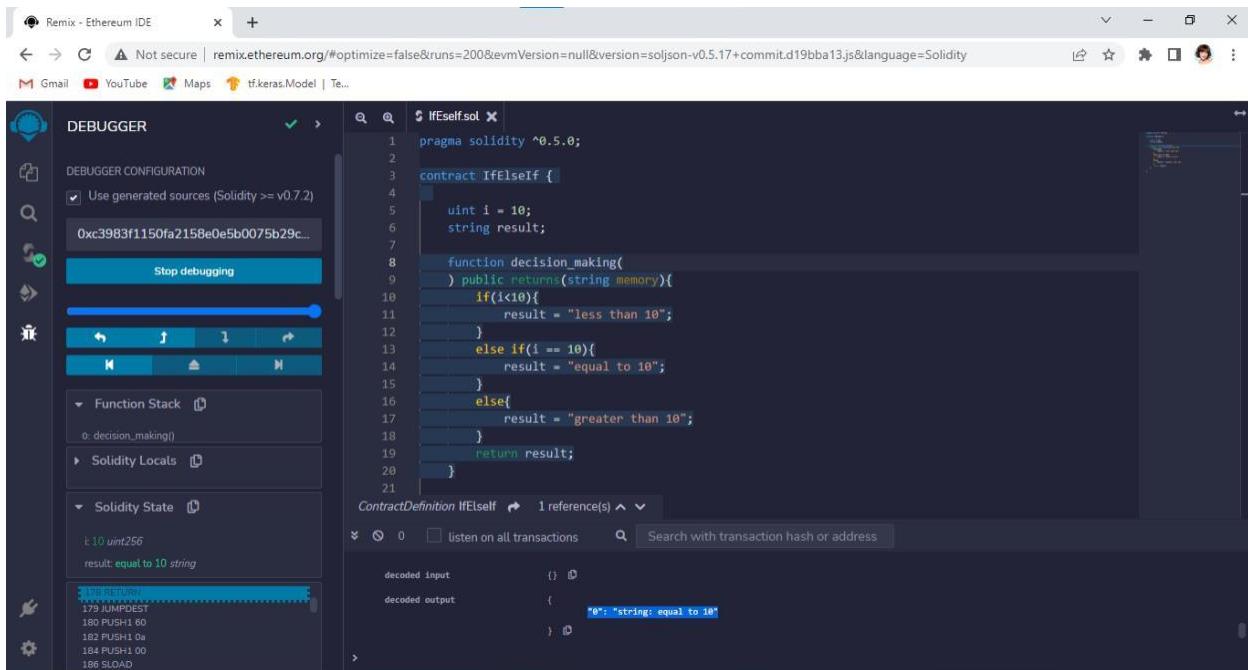


```
pragma solidity ^0.5.0;

contract IfElseIf {
    uint i = 10;
    string result;

    function decision_making()
        public returns(string memory){
        if(i<10){
            result = "less than 10";
        }
        else if(i == 10){
            result = "equal to 10";
        }
        else{
            result = "greater than 10";
        }
        return result;
    }
}
```

Output:



The screenshot shows the Remix Ethereum IDE with the DEBUGGER tab selected. The code remains the same as above. In the DEBUGGER section, the 'Solidity Locals' panel shows 'i: 10' and 'result: equal to 10'. The 'Solidity State' panel shows the assembly code for the current state:

```
0x3983f1150fa2158e0e5b0075b29c...
0: decision_making()
1: 10 uint256
2: result equal to 10 string
3: 100 JUMPDEST
4: 101 PUSH1 60
5: 102 PUSH1 0a
6: 103 PUSH1 00
7: 104 SLOAD
```

String

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several Solidity files. The main editor area contains the following Solidity code:

```
pragma solidity ^0.5.0;

contract StringsExample
{
    string text;

    function setText () public returns (string memory)
    {
        text = "Hello World";
        return text;
    }
}
```

The code defines a contract named `StringsExample` with a single state variable `text` of type `string`. It includes a public function `setText` that sets the value of `text` to `"Hello World"` and returns it.

Output:

The screenshot shows the Remix Ethereum IDE interface with the "DEBUGGER" tab selected. The left sidebar displays a "DEBUGGER CONFIGURATION" section with a checked checkbox for "Use generated sources (Solidity >= v0.7.2)". Below this is a "Function Stack" showing a single entry: `o: setText()`. The main editor area contains the same Solidity code as the previous screenshot. The bottom pane shows the assembly output for the `setText` function:

```
ContractDefinition StringsExample 1 reference(s) ↗
0x02b5a78ec919c54377899e8881ef... 0 listen on all transactions Search with transaction hash or address

decoded input (0) ↗
decoded output (1) ↗
    0: PUSH1 60
    1: DUP1
    2: PUSH1 40
    3: MLOAD
    4: DUP1
    5: PUSH1 40
    6: ADD
    7: JUMPDEST
    8: PUSH1 40
    9: DUP1
    10: MLOAD
    11: DUP1
    12: PUSH1 40
    13: ADD
    14: RETURN
```

The assembly code shows the deployment of the contract, followed by the execution of the `setText` function. The function starts with `PUSH1 60`, which is then copied to the stack via `DUP1`. It then pushes `40` onto the stack, performs an `MLOAD` at the current stack position, and copies it back to the stack via `DUP1`. This pattern repeats for the second argument. Finally, the `ADD` instruction adds the two arguments together, and the `RETURN` instruction exits the function.

Array

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several Solidity files: Variable.sol, BitwiseOperator.sol, 1A.sol, AssignmentOperator.sol, ArithmeticOperator.sol, WithdrawalPattern.sol, RestrictedAccess.sol, Contracts.sol, Inheritance.sol, Constructors.sol, AbstractConstructor.sol, Interfaces.sol, Libraries.sol, Assembly.sol, Events.sol, ErrorHandling.sol, ArithmeticOperator.sol, RelationalOperator.sol, LogicalOperator.sol, Downhillloop.sol, ForLoop.sol, IfStatement.sol, IfElse.sol, Ifself.sol, String.sol, and Array.sol. The "Array.sol" file is currently selected. The main editor area contains the following Solidity code:

```
// Creating a contract
contract ArrayExample {
    uint[] data
    = [10, 20, 30, 40, 50];
    int[] data1;
    function dynamic_array() public returns(
        uint[] memory, int[] memory){
        data1
        = [int(-60), 70, -80, 90, -100, -120, 140];
        return (data, data1);
    }
}
```

Below the code, the "ContractDefinition Types" section shows "1 reference(s)". A transaction log is displayed:

- [vm] from: 0x583...eddC4 to: ArrayExample.(constructor) value: 0 wei data: 0x600...10032 logs: 0 hash: 0x625...00b09
- call to ArrayExample.testArray

The bottom status bar shows the date and time: 7/24/2022 3:34 PM.

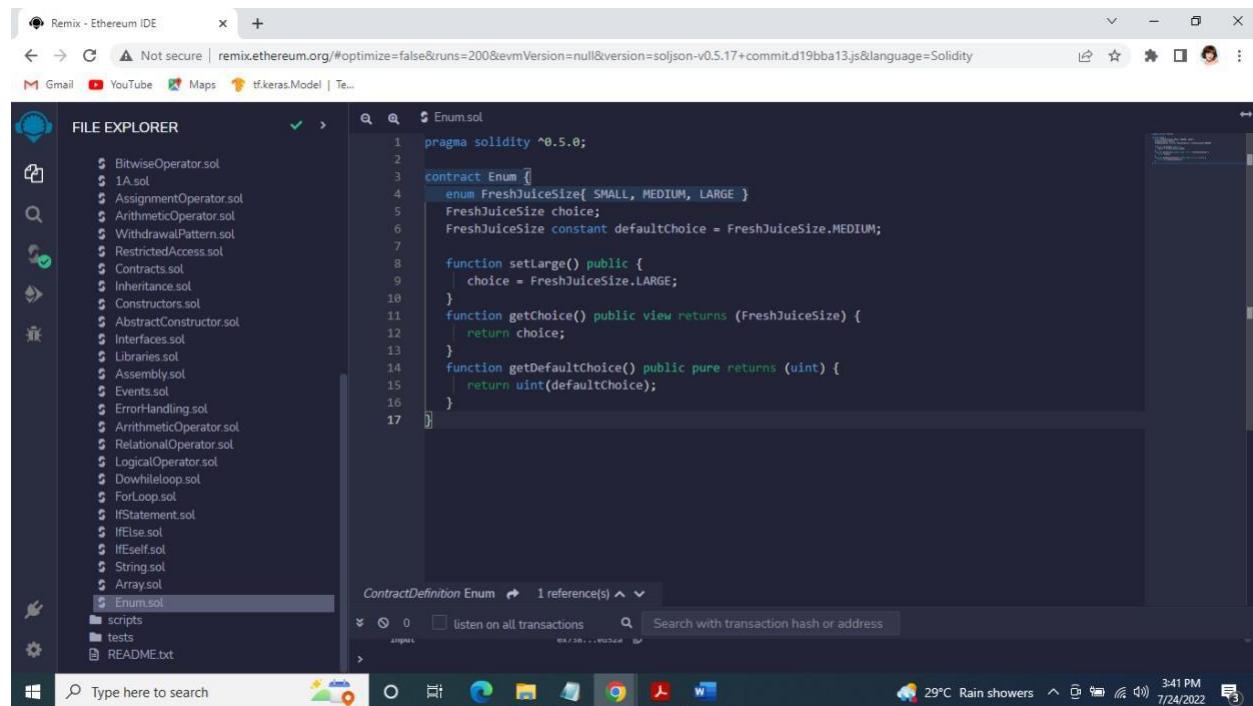
Output:

The screenshot shows the Remix Ethereum IDE interface with the "DEBUGGER" tab selected. The left sidebar includes "DEBUGGER CONFIGURATION" with the option "Use generated sources (Solidity >= v0.7.2)" checked, and a "Function Stack" and "Solidity Locals" section. The main editor area shows the same Solidity code as before. The bottom pane displays the debugger output:

- input: 0x735733decb9b850d092d87c6aea4...
- decoded input: ()
- decoded output: {
 - "0": "uint256": 10, 20, 30, 40, 50"
 - "1": "int256": -60, 70, -80, 90, -100, -120, 140"}
- logs: ()

Enums

Code:



The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several Solidity files. The main editor window is titled "Enum.sol" and contains the following Solidity code:

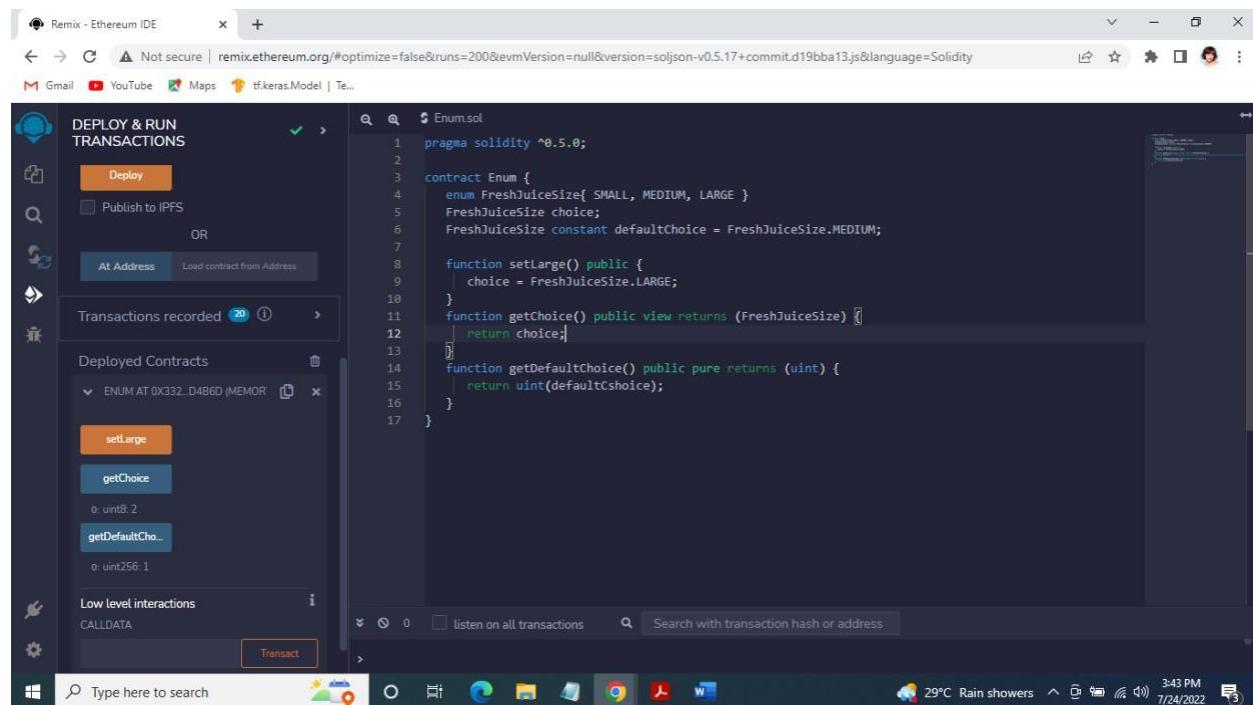
```
pragma solidity ^0.5.0;

contract Enum {
    enum FreshJuiceSize{ SMALL, MEDIUM, LARGE }
    FreshJuiceSize choice;
    FreshJuiceSize constant defaultChoice = FreshJuiceSize.MEDIUM;

    function setLarge() public {
        choice = FreshJuiceSize.LARGE;
    }
    function getChoice() public view returns (FreshJuiceSize) {
        return choice;
    }
    function getDefaultChoice() public pure returns (uint) {
        return uint(defaultChoice);
    }
}
```

The code defines an enum named "FreshJuiceSize" with three values: SMALL, MEDIUM, and LARGE. It also defines a contract "Enum" with a variable "choice" of type "FreshJuiceSize" and a constant "defaultChoice" set to MEDIUM. The contract includes functions to set the choice to LARGE, get the current choice, and get the default choice as a uint.

Output:



The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" sidebar open. The sidebar includes buttons for "Deploy" and "Publish to IPFS", and dropdowns for "At Address" and "Load contract from Address". Below these are sections for "Transactions recorded" and "Deployed Contracts". The "Deployed Contracts" section shows a single contract named "ENUM AT 0X332-D4B6D (MEMORY)". Under this contract, there are three buttons: "setLarge", "getChoice", and "getDefaultCho...". The "getChoice" button is currently highlighted. The main editor window shows the same Solidity code as the previous screenshot. The status bar at the bottom indicates the date and time as 7/24/2022 3:43 PM.

Struct

Code:

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORER sidebar lists several Solidity files, including 1A.sol, AssignmentOperator.sol, ArithmeticOperator.sol, WithdrawalPattern.sol, RestrictedAccess.sol, Contracts.sol, Inheritance.sol, Constructors.sol, AbstractConstructor.sol, Interfaces.sol, Libraries.sol, Assembly.sol, Events.sol, ErrorHandling.sol, ArithmeticOperator.sol, RelationalOperator.sol, LogicalOperator.sol, DoWhileLoop.sol, ForLoop.sol, IfStatement.sol, IfElse.sol, Ifself.sol, String.sol, Array.sol, Enum.sol, and Struct.sol. The Struct.sol file is currently selected and open in the main editor area. The code defines a contract named Struct with a struct Book containing title, author, and book_id fields. It includes a setBook function that creates a new Book instance and a getBookId function that returns the book_id. The Solidity version pragma is ^0.5.0.

```
pragma solidity ^0.5.0;

contract Struct {
    struct Book {
        string title;
        string author;
        uint book_id;
    }
    Book book;

    function setBook() public {
        book = Book('Learn Java', 'TP', 1);
    }
    function getBookId() public view returns (uint) {
        return book.book_id;
    }
}
```

Output:

The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" tab active. The left sidebar displays the Deploy section with options for Deploy, Publish to IPFS, At Address, and Load contract from Address. Below this, the Transactions recorded section shows 24 transactions. The Deployed Contracts section lists a deployed contract named STRUCT AT 0X93F...C96CC (MEMO). Under this contract, two functions are listed: setBook and getBookId. The setBook function has a value of 0: uint256: 4. The getBookId function is highlighted with a blue background. The bottom section shows Low level interactions with a CALldata input field and a Transact button.

Mapping

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several other .sol files. The main editor window is titled "Mapping.sol" and contains the following Solidity code:

```
pragma solidity ^0.4.18;

contract mapping_example {

    struct student {
        string name;
        string subject;
        uint8 marks;
    }

    mapping (address => student) result;
    address[] public student_result;

    function adding_values() public {
        var student
        = result[0x0EE7796E89C82C368Add1375076f39069FafE252];

        student.name = "John";
        student.subject = "Chemistry";
        student.marks = 88;
        student_result.push(
            0x0EE7796E89C82C368Add1375076f39069FafE252) -1;
    }
}
```

The code defines a contract named "mapping_example" with a struct "student" and a mapping from address to "student". It also has an array "student_result" and a function "adding_values" that adds a new entry to the mapping.

Output:

The screenshot shows the Remix Ethereum IDE with the "DEBUGGER" tab selected. The left sidebar shows the "Function Stack" and "Solidity Locals". The main editor window is still titled "Mapping.sol" and shows the same Solidity code as above. The bottom pane displays the debugger logs, which show a transaction being processed:

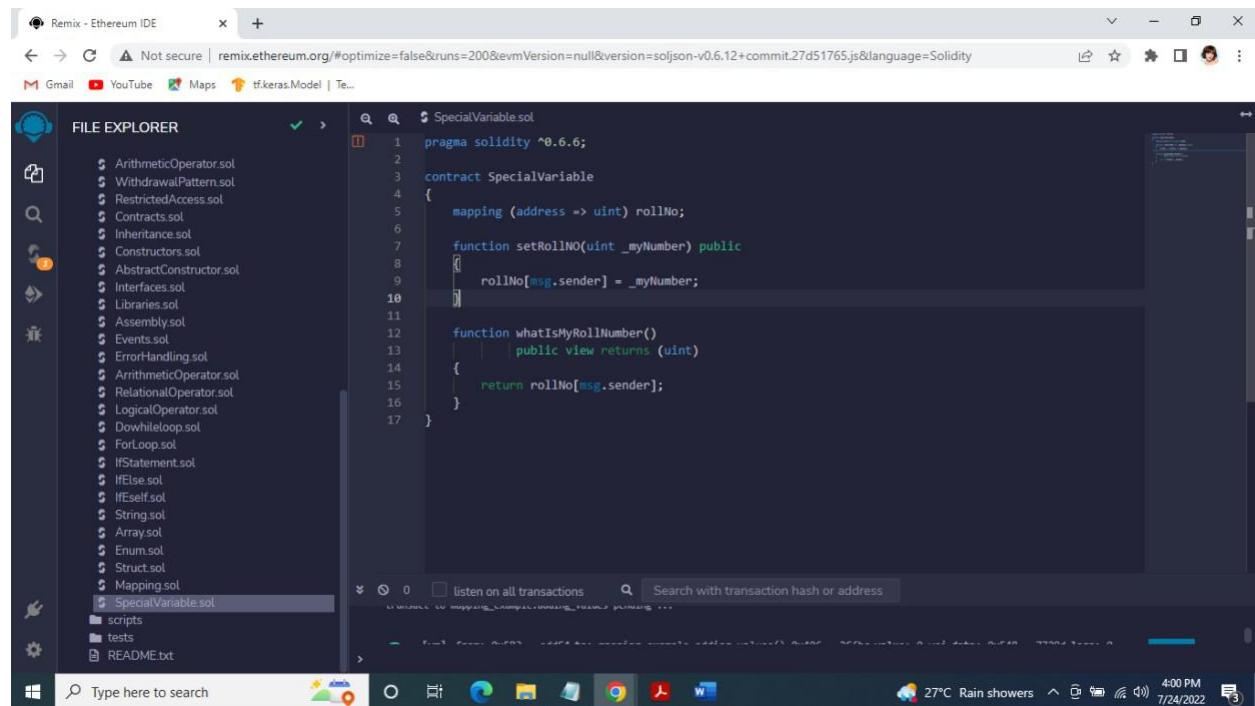
```
[vm] from: 0x583...eddC4 to: mapping_example.(constructor) value: 0 wei data: 0x608...40029 logs: 0 hash: 0x806...231b7
transact to mapping_example.adding_values pending ...

[vm] from: 0x583...eddC4 to: mapping_example.adding_values() 0x406...2CFbc value: 0 wei data: 0x548...7729d logs: 0
hash: 0x16b...f8d79
call to mapping_example.student_result errored: Error encoding arguments: Error: invalid BigNumber string (argument="value", value="", code=INVALID_ARGUMENT)
```

The logs indicate that a transaction was sent to the constructor and another to the "adding_values" function, with the second one currently pending.

Special Variable

Code:



The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several Solidity files. The main editor window displays the code for "SpecialVariable.sol". The code defines a contract named "SpecialVariable" with two functions: "setRollNo" and "whatIsMyRollNumber". The "setRollNo" function is a public function that takes a uint parameter "_myNumber" and updates the mapping "rollNo" with the sender's address as the key and the value. The "whatIsMyRollNumber" function is a public view function that returns the value from the "rollNo" mapping for the sender's address.

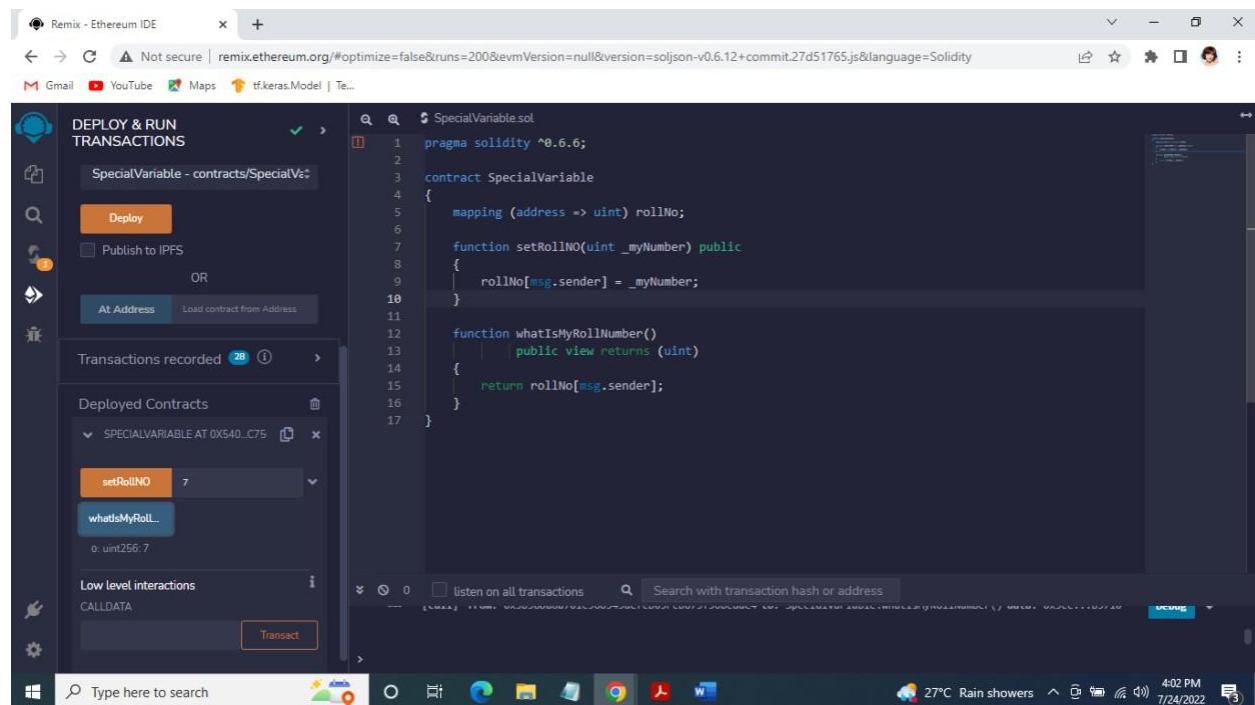
```
pragma solidity ^0.6.6;

contract SpecialVariable
{
    mapping (address => uint) rollNo;

    function setRollNo(uint _myNumber) public
    {
        rollNo[msg.sender] = _myNumber;
    }

    function whatIsMyRollNumber()
    public view returns (uint)
    {
        return rollNo[msg.sender];
    }
}
```

Output:

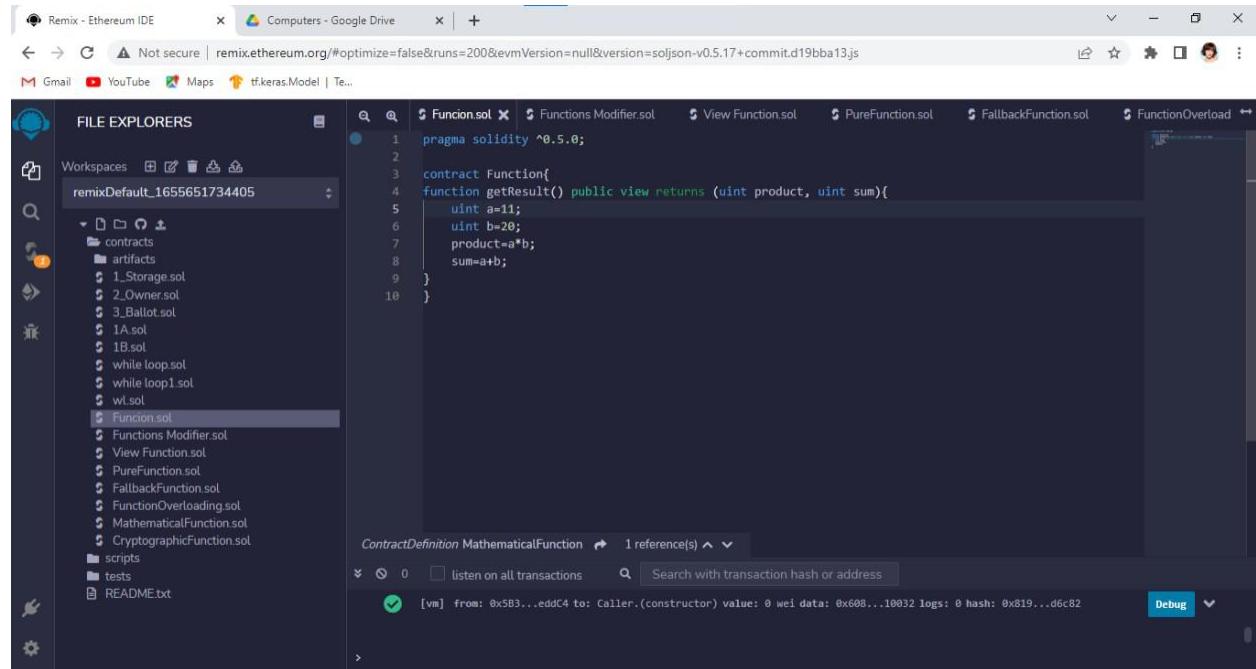


The screenshot shows the Remix Ethereum IDE interface after the contract has been deployed. The left sidebar now includes a "DEPLOY & RUN TRANSACTIONS" section. The "Deploy" button is highlighted. Below it, there are options to "Publish to IPFS" or "At Address". The "At Address" option is selected, and the address "SPECIALVARIABLE AT 0x540...C75" is shown. The main editor window still displays the "SpecialVariable.sol" code. The bottom pane shows a "Transactions recorded" section with 28 entries, a "Deployed Contracts" section listing the deployed contract, and a "Low level interactions" section with a "Transact" button.

2b) Functions, Function Modifiers, View functions, Pure Functions, Fallback Function, Function Overloading, Mathematical functions, Cryptographic functions.

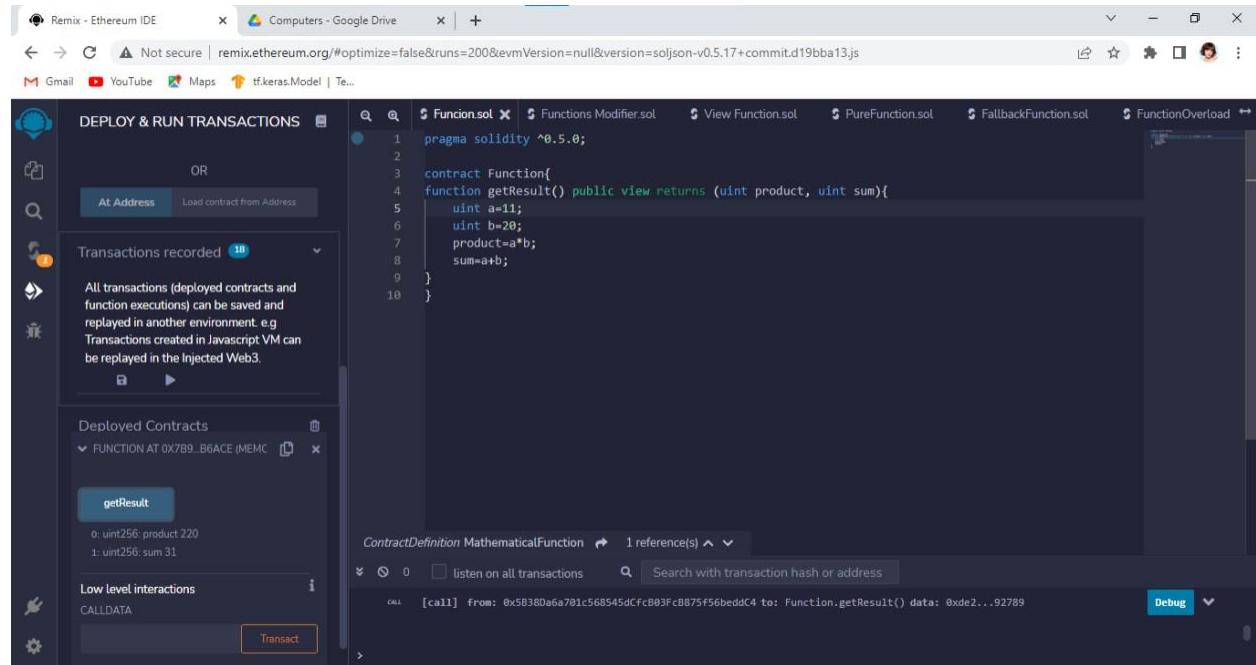
Function:

Code:



```
pragma solidity ^0.5.0;
contract Function{
    function getResult() public view returns (uint product, uint sum){
        uint a=11;
        uint b=20;
        product=a*b;
        sum=a+b;
    }
}
```

Output:



DEPLOY & RUN TRANSACTIONS

At Address: FUNCTION AT 0x7B9...B6ACE (MEMC)

getResult

0: uint256: product 220
1: uint256: sum 31

Low level interactions

CALLDATA

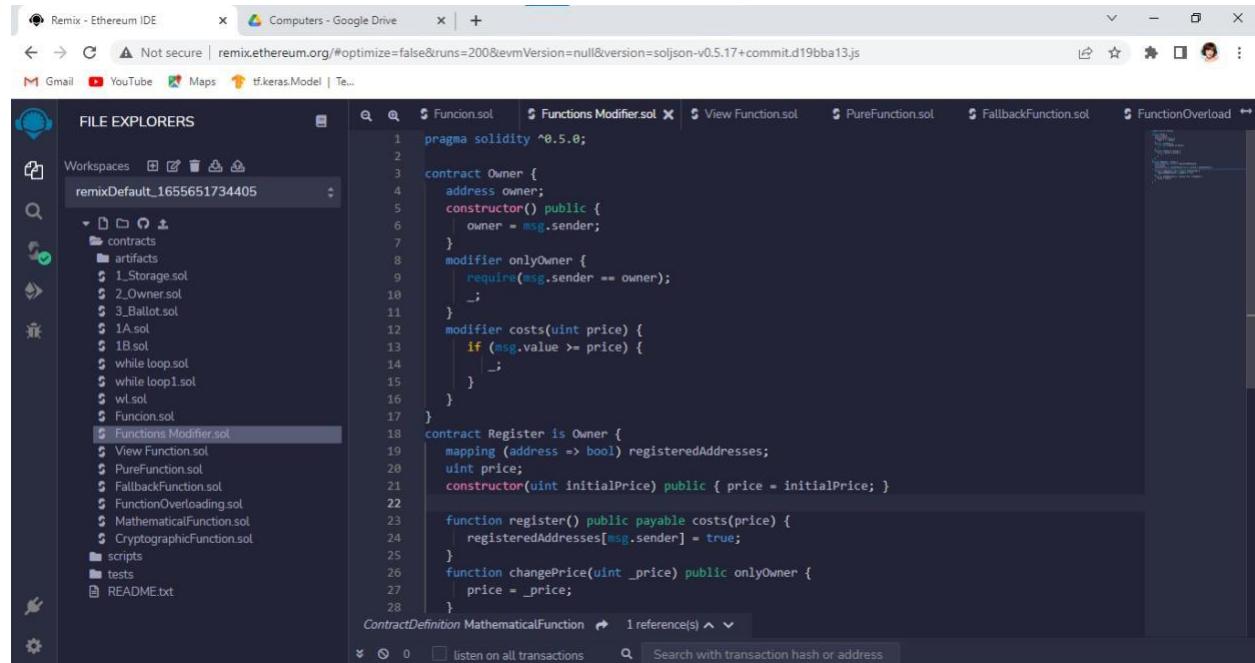
Transact

ContractDefinition MathematicalFunction 1 reference(s) ▾

[call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: Function.getResult() data: 0xde2...92789

Functions Modifiers

Code:



The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file tree under 'FILE EXPLORERS' for workspace 'remixDefault_1655651734405'. The current file selected is 'Functions Modifier.sol'. The code editor on the right contains the following Solidity code:

```
pragma solidity ^0.5.0;

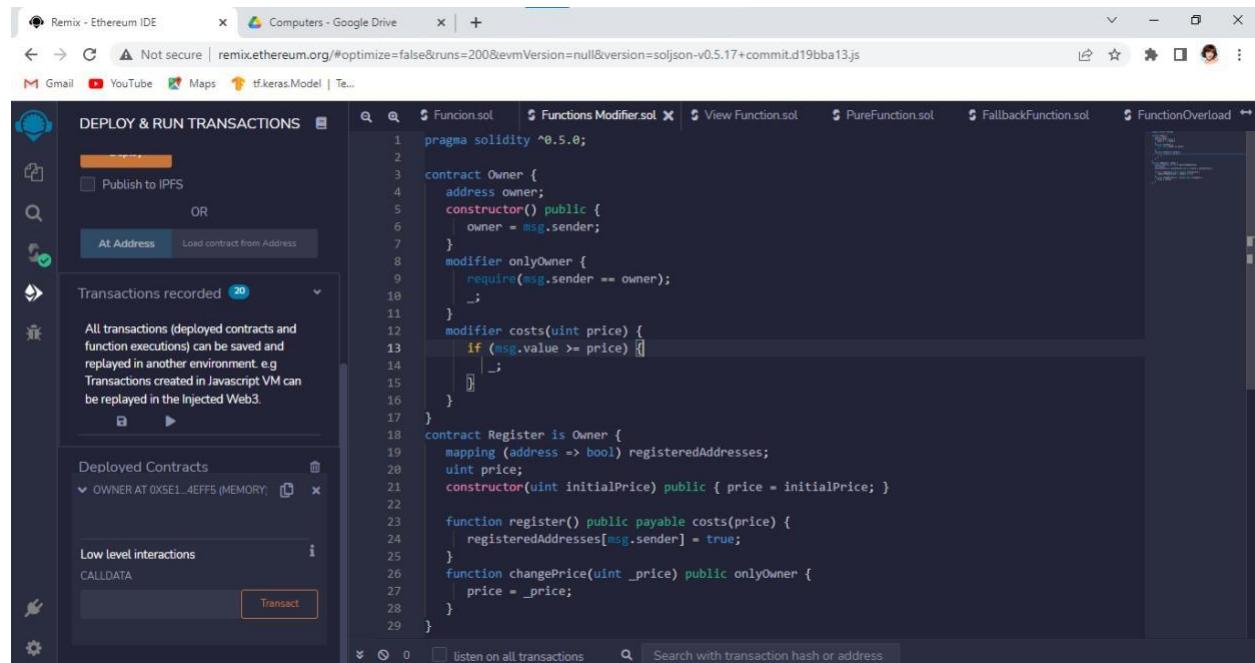
contract Owner {
    address owner;
    constructor() public {
        owner = msg.sender;
    }
    modifier onlyOwner {
        require(msg.sender == owner);
    }
    modifier costs(uint price) {
        if (msg.value >= price) {
        }
    }
}

contract Register is Owner {
    mapping (address => bool) registeredAddresses;
    uint price;
    constructor(uint initialPrice) public { price = initialPrice; }

    function register() public payable costs(price) {
        registeredAddresses[msg.sender] = true;
    }
    function changePrice(uint _price) public onlyOwner {
        price = _price;
    }
}
```

The code defines two contracts: 'Owner' and 'Register'. The 'Owner' contract has a constructor that sets the owner to the message sender. It includes two modifiers: 'onlyOwner' which checks if the sender is the owner, and 'costs' which requires the sender to send at least the specified price. The 'Register' contract inherits from 'Owner' and adds a mapping of addresses to booleans for tracking registrations. It has a constructor setting an initial price. It includes a 'register' function that adds the sender to the mapping if the price is paid, and a 'changePrice' function that allows the owner to change the price.

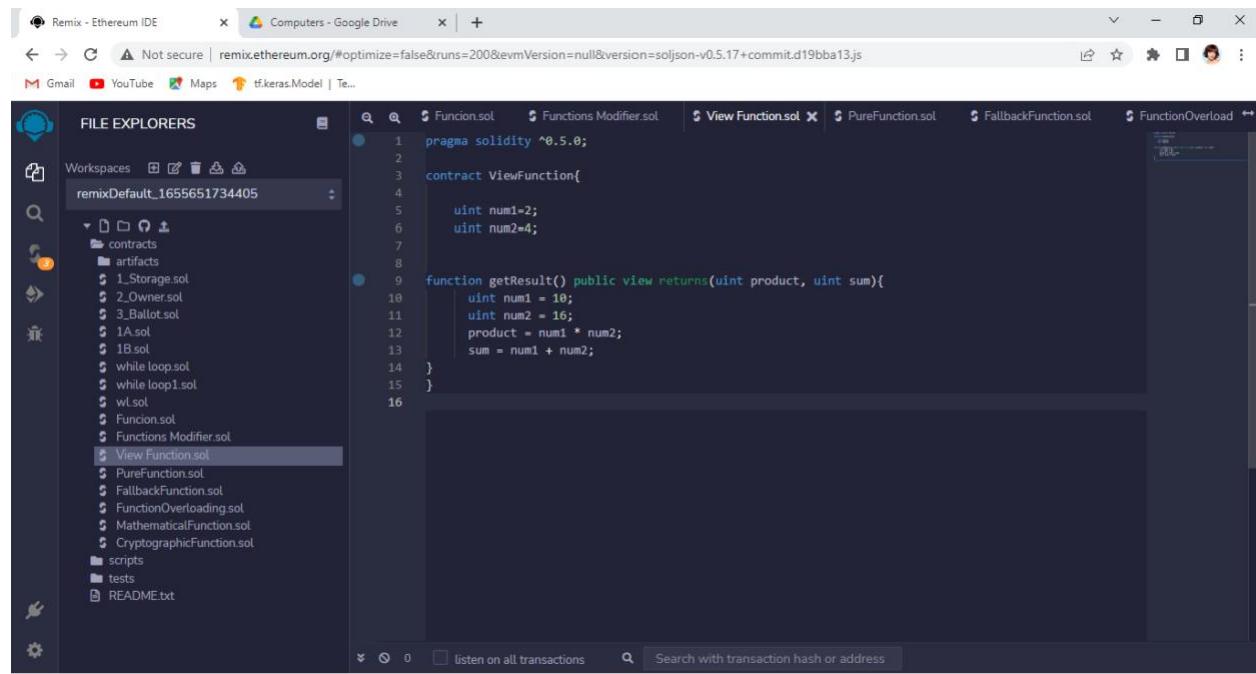
Output:



The screenshot shows the Remix Ethereum IDE interface with the 'DEPLOY & RUN TRANSACTIONS' sidebar open. The sidebar shows deployment options like 'Publish to IPFS' and 'At Address'. It also displays a list of recorded transactions and a list of deployed contracts. A specific contract named 'OWNER' is highlighted, with its address shown as 'OWNER AT 0X5E1_4EFF5 (MEMORY)'. The code editor on the right is identical to the one in the previous screenshot, showing the same Solidity code for the 'Owner' and 'Register' contracts.

View function

Code:



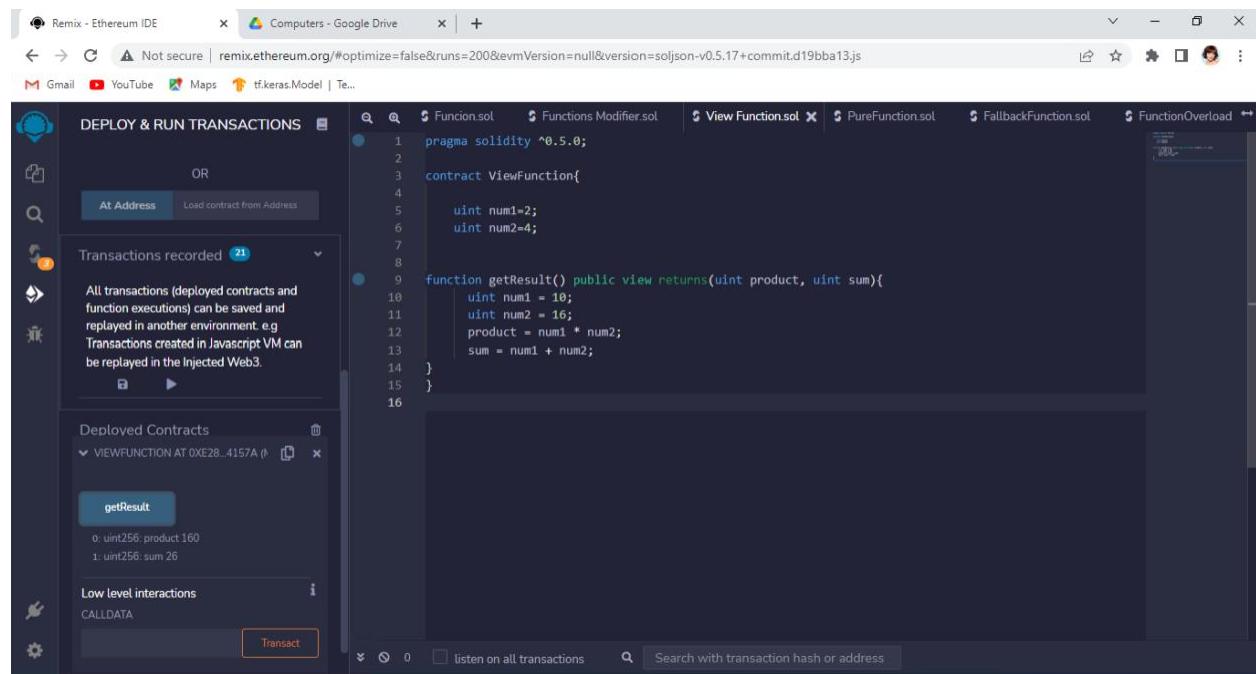
The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file explorer with several Solidity files: Funcion.sol, Functions Modifier.sol, View Function.sol (which is selected), PureFunction.sol, FallbackFunction.sol, and FunctionOverload.sol. The main editor area contains the following Solidity code:

```
pragma solidity ^0.5.0;

contract ViewFunction{
    uint num1=2;
    uint num2=4;

    function getResult() public view returns(uint product, uint sum){
        uint num1 = 10;
        uint num2 = 16;
        product = num1 * num2;
        sum = num1 + num2;
    }
}
```

Output:

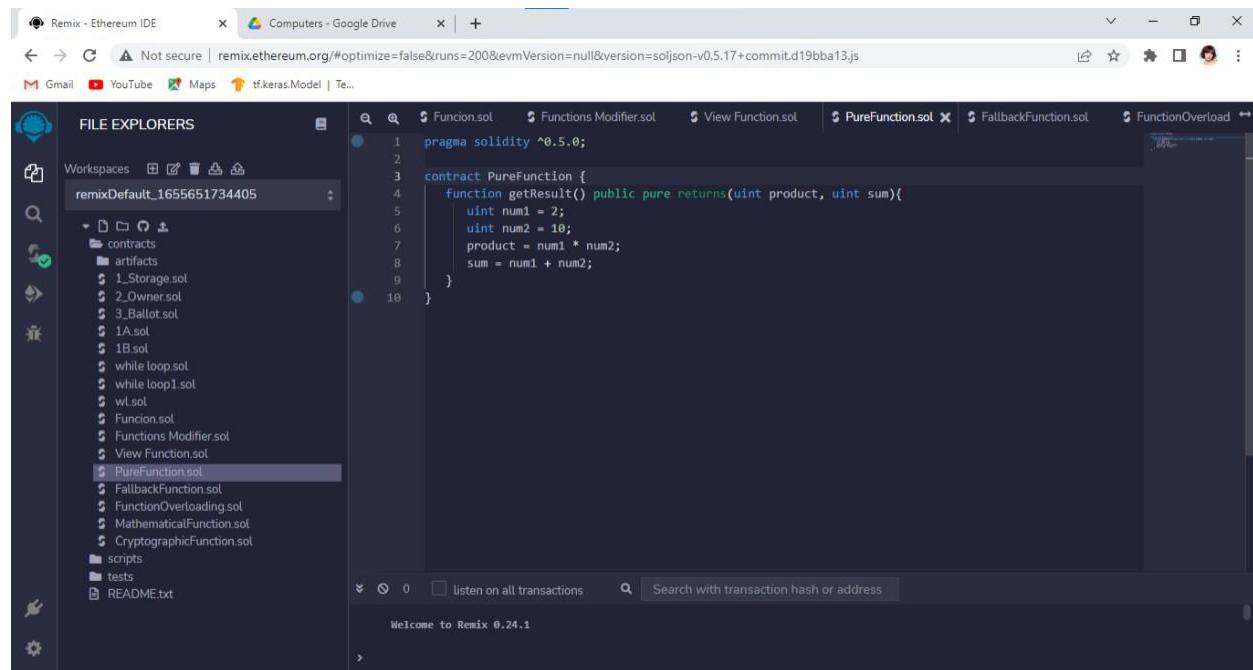


The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" tab active. The left sidebar shows a deployment log with the message "Transactions recorded 21". Below it, the "Deployed Contracts" section lists "VIEWFUNCTION AT 0xE28...4157A". The main editor area contains the same Solidity code as the previous screenshot. In the bottom left corner, there is a callout box showing the result of the "getResult" function call:

getResult
0: uint256: product 160
1: uint256: sum 26

Pure function

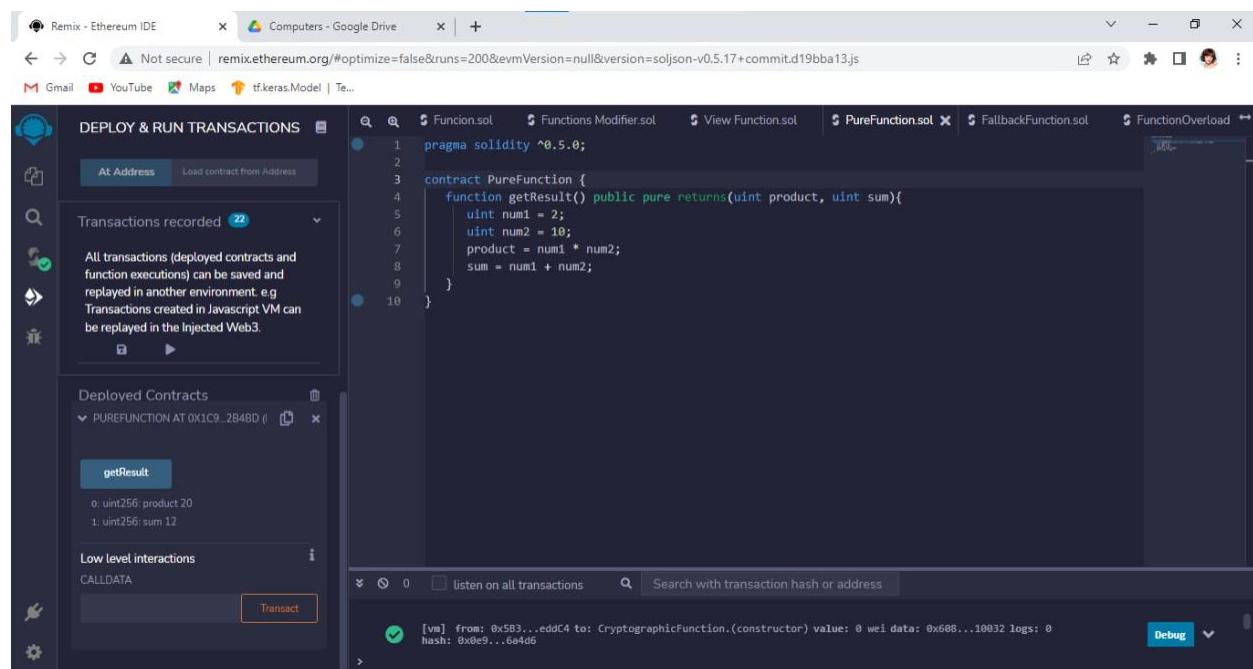
Code:



```
pragma solidity ^0.5.0;

contract PureFunction {
    function getResult() public pure returns(uint product, uint sum){
        uint num1 = 2;
        uint num2 = 10;
        product = num1 * num2;
        sum = num1 + num2;
    }
}
```

Output:



Deploy & Run Transactions

At Address Load contract from Address

Transactions recorded 22

All transactions (deployed contracts and function executions) can be saved and replayed in another environment, e.g. Transactions created in Javascript VM can be replayed in the Injected Web3.

Deployed Contracts

PUREFUNCTION AT 0x1C9...2B4BD ()

getResult

0: uint256: product 20
1: uint256: sum 12

Low level interactions

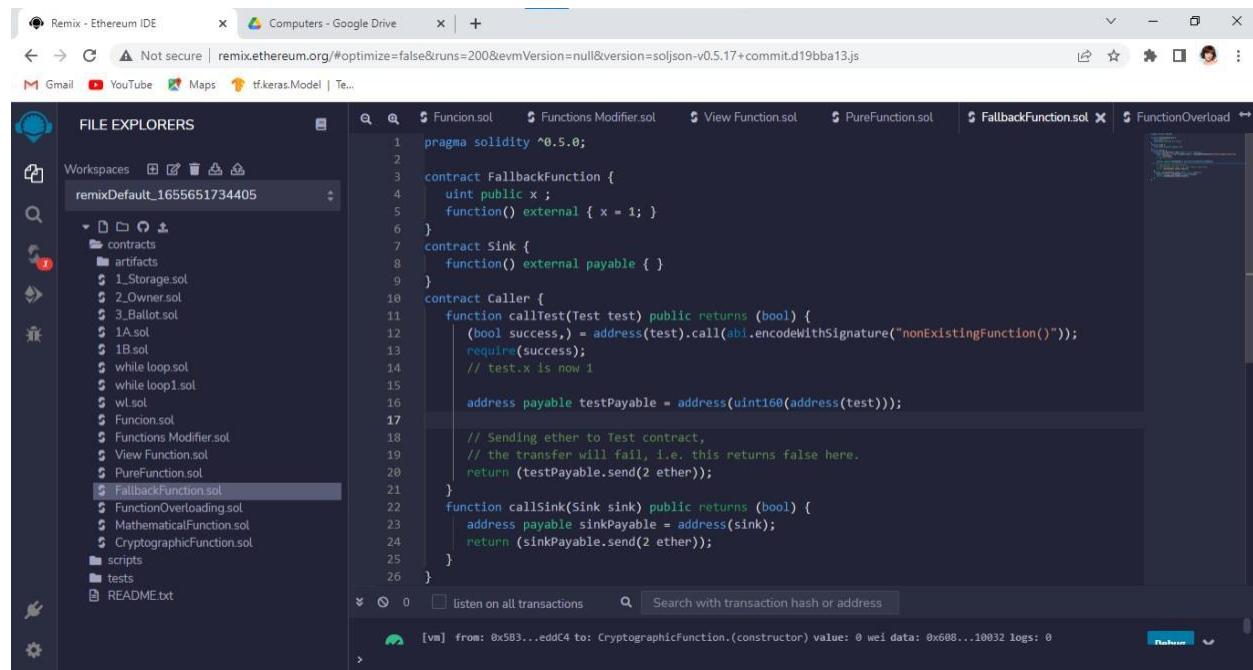
CALLDATA

Transact

[vm] from: 0x5B3...eddC4 to: CryptographicFunction.(constructor) value: 0 wei data: 0x600...10032 logs: 0 hash: 0x0e9...6a4d6

Fallback function

Code:



The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file explorer with several Solidity files: remixDefault_1655651734405, contracts (with artifacts like 1_Storage.sol, 2_Owner.sol, 3_Ballot.sol), 1A.sol, 1B.sol, while.loop.sol, while.loops.sol, wl.sol, Funcion.sol, Functions.Modifier.sol, View.Function.sol, PureFunction.sol, FallbackFunction.sol (which is selected), FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, scripts, tests, and README.txt. The right panel contains the Solidity code for FallbackFunction.sol:

```
pragma solidity ^0.5.0;

contract FallbackFunction {
    uint public x;
    function() external { x = 1; }
}

contract Sink {
    function() external payable {}
}

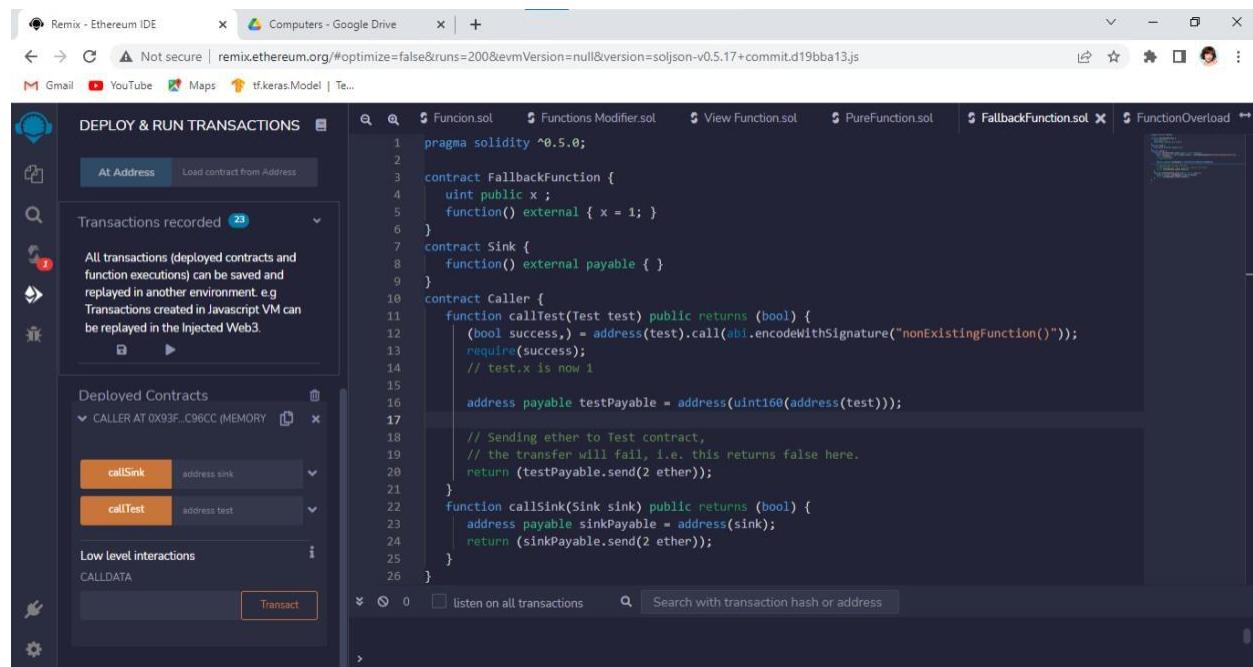
contract Caller {
    function callTest(Test test) public returns (bool) {
        (bool success,) = address(test).call(abi.encodeWithSignature("nonExistingFunction()"));
        require(success);
        // test.x is now 1
    }

    address payable testPayable = address(uint160(address(test)));

    // Sending ether to Test contract,
    // the transfer will fail, i.e. this returns false here.
    return (testPayable.send(2 ether));
}

function callSink(Sink sink) public returns (bool) {
    address payable sinkPayable = address(sink);
    return (sinkPayable.send(2 ether));
}
```

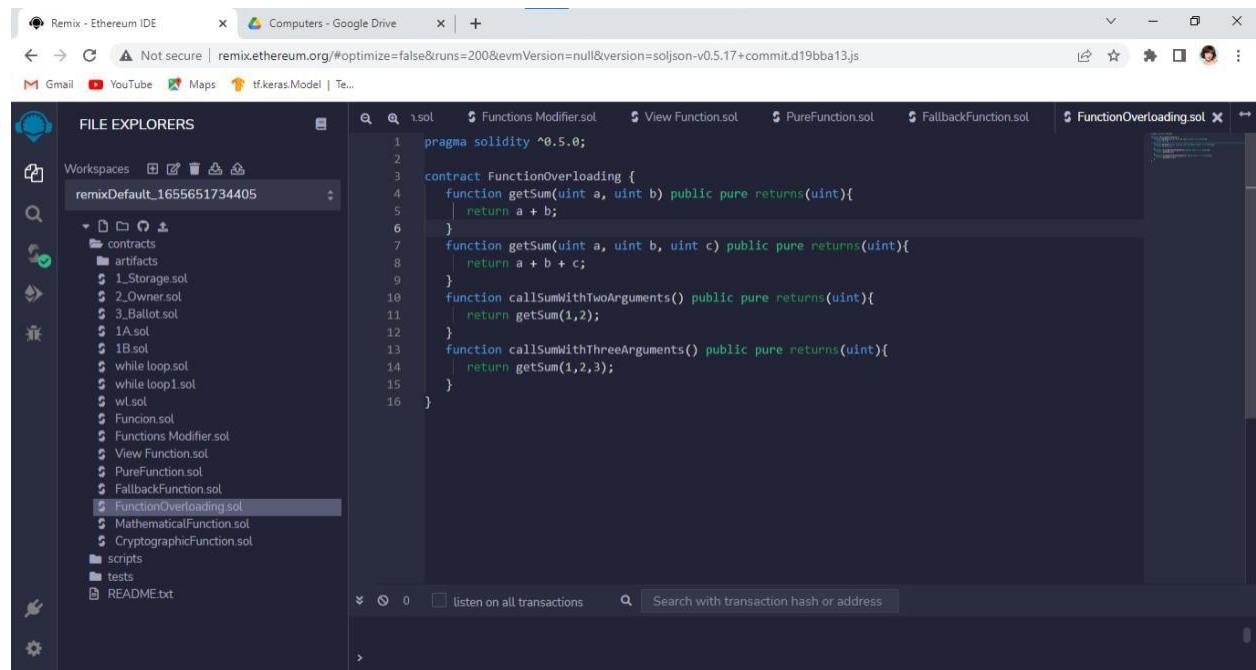
Output:



The screenshot shows the Remix Ethereum IDE interface with the "Deploy & Run Transactions" tab selected. The left sidebar displays a "DEPLOY & RUN TRANSACTIONS" section with "Transactions recorded 23". It includes a note: "All transactions (deployed contracts and function executions) can be saved and replayed in another environment, e.g. Transactions created in Javascript VM can be replayed in the Injected Web3." Below this is a "Deployed Contracts" section showing "CALLER AT 0X93F...C96CC (MEMORY)". Under "Low level interactions", there are two entries: "callSink" and "callTest", each with a dropdown menu showing their respective addresses. The right panel contains the same Solidity code as the previous screenshot.

Function Overloading

Code:

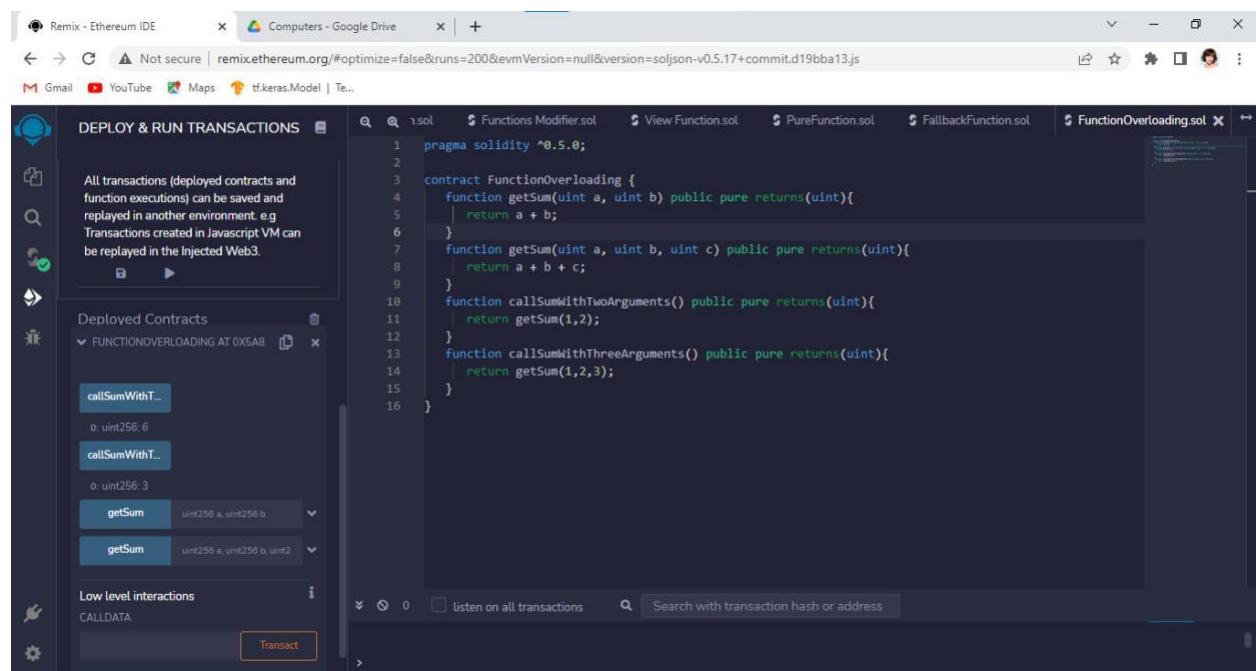


The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORERS" and lists several contracts and scripts. The "FunctionOverloading.sol" file is currently selected and highlighted in blue. The main code editor window displays the Solidity code for the "FunctionOverloading" contract, which includes three functions: getSum, callSumWithTwoArguments, and callSumWithThreeArguments.

```
pragma solidity ^0.5.0;

contract FunctionOverloading {
    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;
    }
    function callSumWithTwoArguments() public pure returns(uint){
        return getSum(1,2);
    }
    function callSumWithThreeArguments() public pure returns(uint){
        return getSum(1,2,3);
    }
}
```

Output:



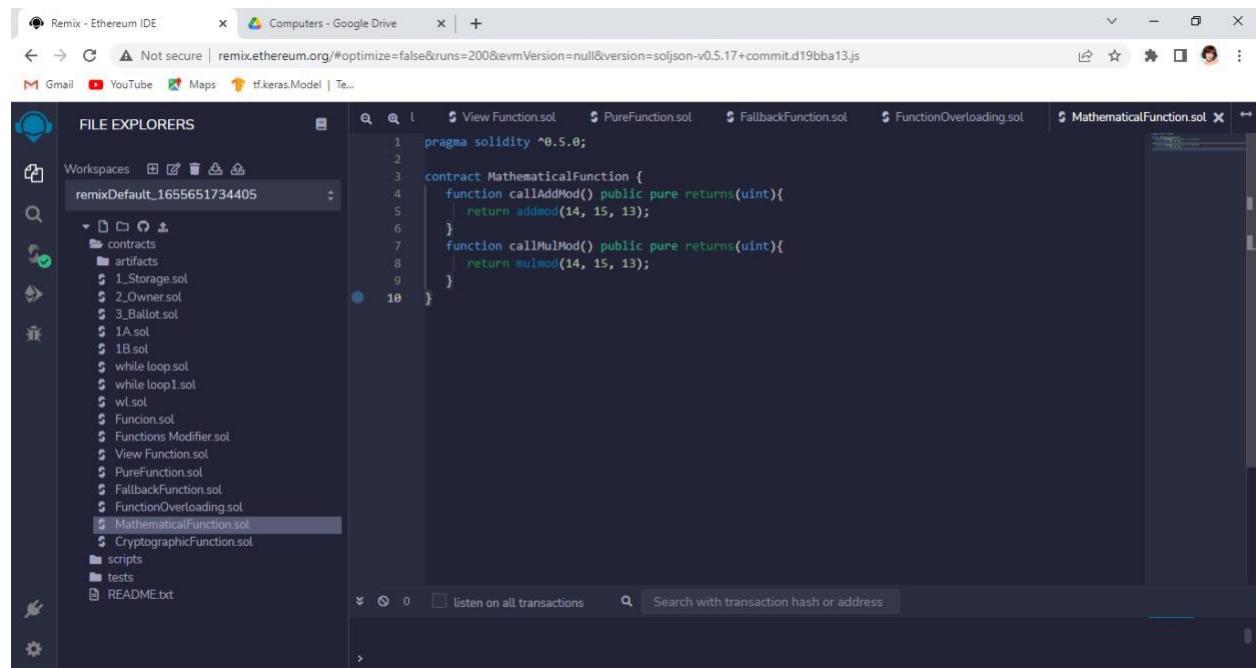
The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" tab active. On the left, there is a message about saved transactions and a list of deployed contracts. Under "Deployed Contracts", the "FUNCTIONOVERLOADING AT 0X5AB" contract is listed. Below it, two transaction history items are shown: "callSumWithT..." and "callSumWithT...". Each item has a "getSum" button with a dropdown menu showing "uint256 a, uint256 b" and "getSum" with "uint256 a, uint256 b, uint2". The main code editor window on the right shows the same Solidity code as the previous screenshot.

```
pragma solidity ^0.5.0;

contract FunctionOverloading {
    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;
    }
    function callSumWithTwoArguments() public pure returns(uint){
        return getSum(1,2);
    }
    function callSumWithThreeArguments() public pure returns(uint){
        return getSum(1,2,3);
    }
}
```

Mathematical Function

Code:

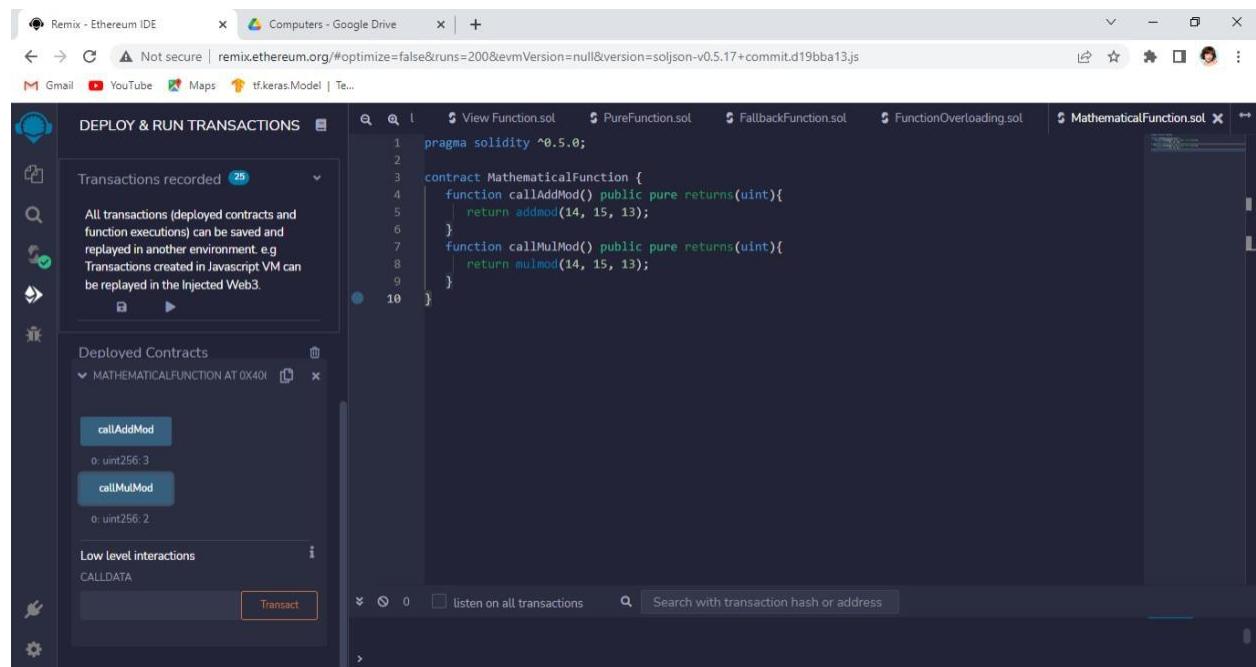


The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file explorer with several Solidity files listed under 'remixDefault_1655651734405'. The central area contains the code for `MathematicalFunction.sol`:

```
pragma solidity ^0.5.0;

contract MathematicalFunction {
    function callAddMod() public pure returns(uint){
        return addmod(14, 15, 13);
    }
    function callMulMod() public pure returns(uint){
        return mulmod(14, 15, 13);
    }
}
```

Output:



The screenshot shows the Remix Ethereum IDE interface with the 'DEPLOY & RUN TRANSACTIONS' tab selected. It displays the deployed contract `MATHEMATICALFUNCTION` at address `0x40f...`. The interface shows two functions: `callAddMod` and `callMulMod`, each returning a value of `uint256: 3` and `uint256: 2` respectively. The left sidebar shows a list of recorded transactions.

Cryptographic function

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORERS" and lists several Solidity files under "remixDefault_165651734405": contracts (artifacts, 1.Storage.sol, 2_Owner.sol, 3_Ballot.sol, 1A.sol, 1B.sol, while.loop.sol, while.loop1.sol, wl.sol, Funcion.sol, Functions Modifier.sol, View Function.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol), scripts, tests, and README.txt. The file "CryptographicFunction.sol" is currently selected and highlighted in blue. The main workspace displays the Solidity code for the "CryptographicFunction" contract:

```
pragma solidity ^0.5.0;
contract CryptographicFunction {
    function callKeccak256() public pure returns(bytes32 result){
        return keccak256("ABC");
    }
}
```

Output:

The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" sidebar open. The "At Address" field is active, and the address "CRYPTOGRAPHICFUNCTION AT 0x4..." is displayed. The sidebar also shows "Transactions recorded" with a count of 26. The main workspace displays the same Solidity code as the previous screenshot. The "Deployed Contracts" section shows the deployed contract "CRYPTOGRAPHICFUNCTION AT 0x4..." with a "callKeccak256" button. Below it, a transaction history entry shows the result of a call to "callKeccak256": "0: bytes32 result 0xe1629b69ddaa060ba30c7908 346fbaf1891c16773fa148d3366701fbba35d54". The "Low level interactions" section includes "CALLDATA" and a "Transact" button.

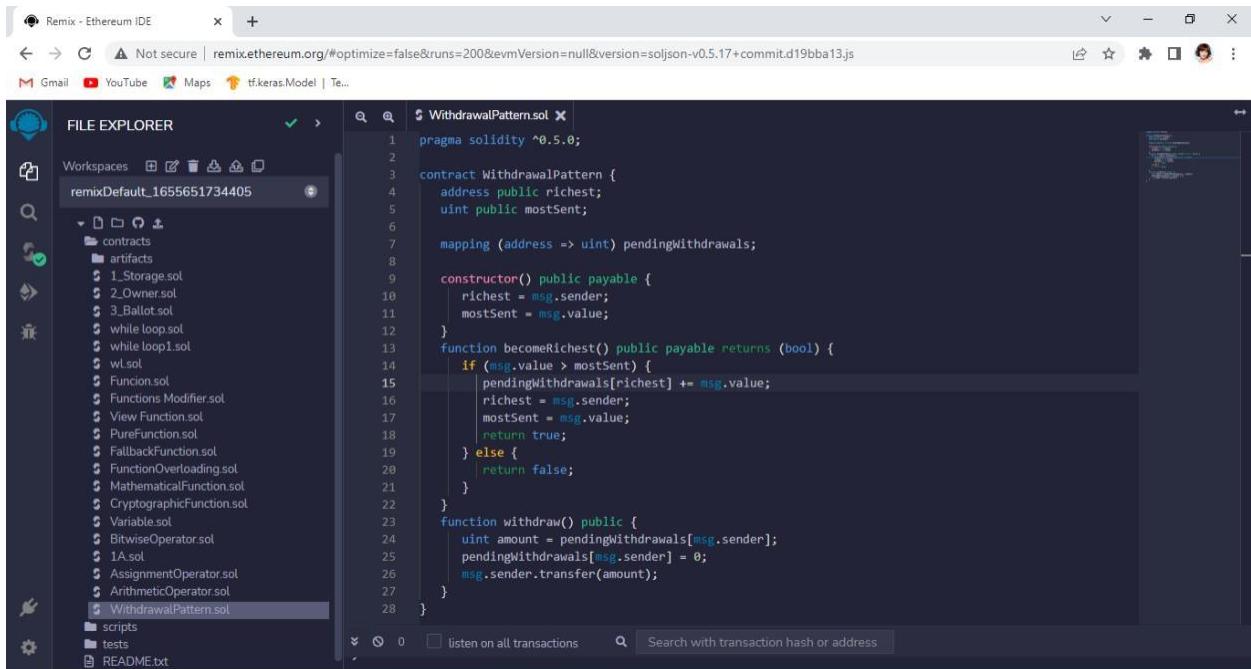
Practical No: 3

Aim: Implement and demonstrate the use of the following in Solidity:

3a) Withdrawal Pattern, Restricted Access.

Withdrawal Pattern

Code:



The screenshot shows the Remix Ethereum IDE interface. The top bar displays the title "Remix - Ethereum IDE" and a URL "remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js". Below the title bar is a toolbar with various icons. The main area is divided into sections: "FILE EXPLORER" on the left showing project files like contracts, artifacts, and tests; a central editor window displaying the Solidity code for "WithdrawalPattern.sol"; and a "CONTRACTS" section on the right showing deployed contracts. The code in the editor is as follows:

```
pragma solidity ^0.5.0;

contract WithdrawalPattern {
    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;
        }
    }

    function withdraw() public {
        uint amount = pendingWithdrawals[msg.sender];
        pendingWithdrawals[msg.sender] = 0;
        msg.sender.transfer(amount);
    }
}
```

Output:

The screenshot shows the Ethereum IDE interface in Remix. On the left, there's a sidebar with options like "DEPLOY & RUN TRANSACTIONS" (with "Deploy" and "Publish to IPFS" buttons), "Transactions recorded" (listing "becomeRichest", "withdraw", "mostSent", and "richest" functions), and "Low level interactions". The main area displays the Solidity code for `WithdrawalPattern.sol`:

```
pragma solidity ^0.5.0;

contract WithdrawalPattern {
    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;
        }
    }

    function withdraw() public {
        require(pendingWithdrawals[msg.sender] > 0);
        msg.sender.transfer(pendingWithdrawals[msg.sender]);
        delete pendingWithdrawals[msg.sender];
    }

    function mostSent() public view returns (uint) {
        return mostSent;
    }

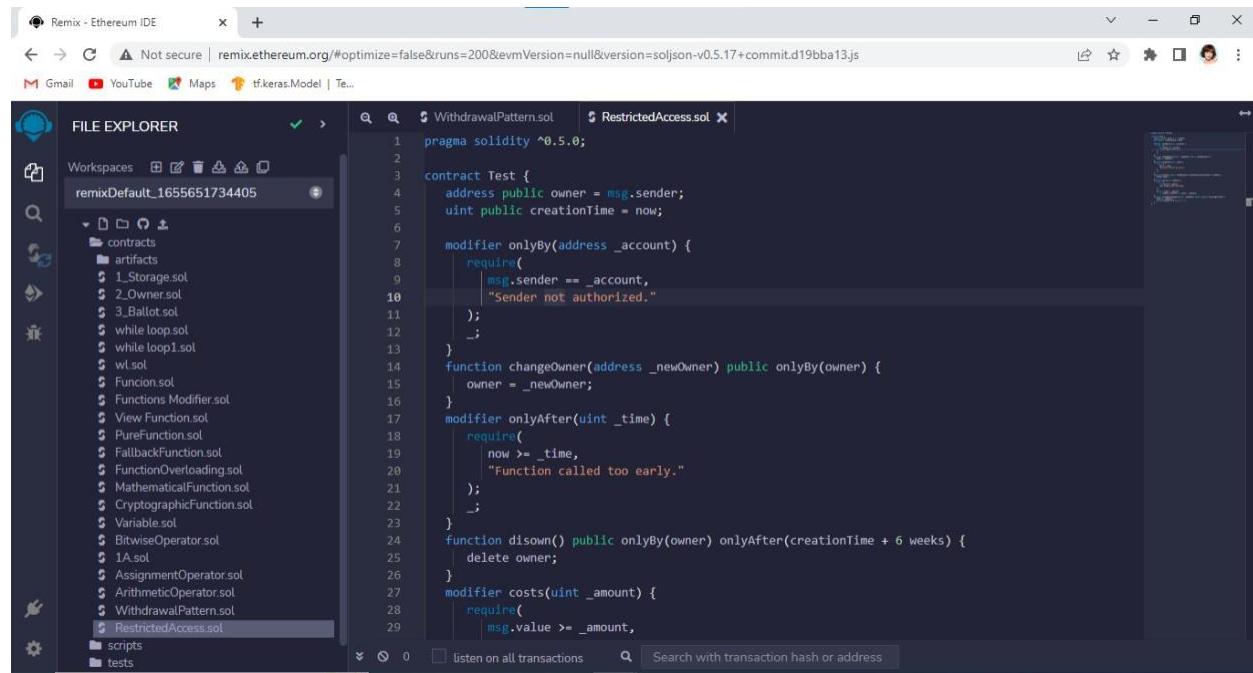
    function richest() public view returns (address) {
        return richest;
    }
}
```

Below the code, there are two transaction logs:

- [vm] from: 0x58...edd4 to: WithdrawalPattern.(constructor) value: 0 wei data: 0x608...10032 logs: 0
hash: 0xa8e...1a9ed
transact to WithdrawalPattern.becomeRichest pending ...
- [vm] from: 0x58...edd4 to: WithdrawalPattern.becomeRichest() 0xd91...39138 value: 0 wei data: 0x699...34ee7 logs: 0
hash: 0x5b3...0b3aa
transact to WithdrawalPattern.withdraw pending ...

Restricted Access

Code:



The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file tree under 'remixDefault_1655651734405' workspace, with 'RestrictedAccess.sol' selected. The main editor pane shows the Solidity code for the 'RestrictedAccess' contract. The code includes modifiers for owner validation, time constraints, and value requirements.

```
pragma solidity ^0.5.0;

contract Test {
    address public owner = msg.sender;
    uint public creationTime = now;

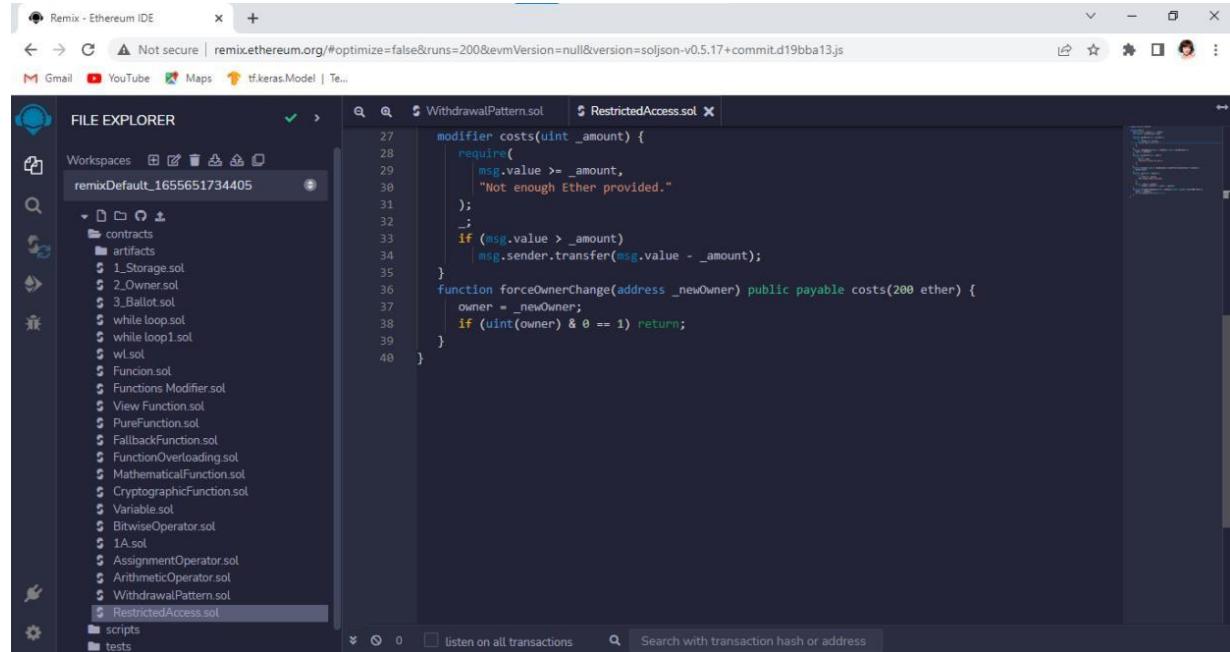
    modifier onlyBy(address _account) {
        require(
            msg.sender == _account,
            "Sender not authorized."
        );
    }

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

    modifier onlyAfter(uint _time) {
        require(
            now >= _time,
            "Function called too early."
        );
    }

    function disown() public onlyBy(owner) onlyAfter(creationTime + 6 weeks) {
        delete owner;
    }

    modifier costs(uint _amount) {
        require(
            msg.value >= _amount,
            "Not enough Ether provided."
        );
    }
}
```

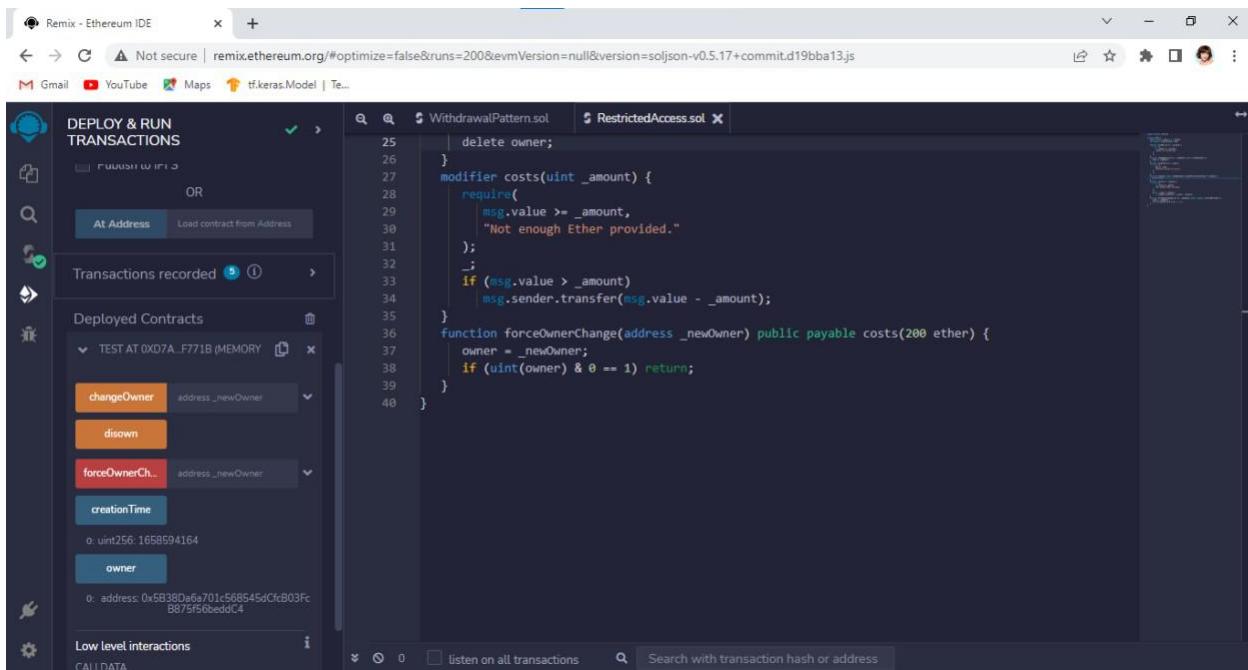


The screenshot shows the continuation of the Remix Ethereum IDE interface. The file tree and workspace remain the same. The main editor pane now displays the latter part of the 'RestrictedAccess.sol' code, specifically focusing on the 'costs' modifier and the 'forceOwnerChange' function.

```
modifier costs(uint _amount) {
    require(
        msg.value >= _amount,
        "Not enough Ether provided."
    );
}

function forceOwnerChange(address _newOwner) public payable costs(200 ether) {
    owner = _newOwner;
    if (uint(owner) & 0 == 1) return;
}
```

Output:



DEPLOY & RUN TRANSACTIONS

Transactions recorded 5 ⓘ

Deployed Contracts

TEST AT 0xd7A...F771B (MEMORY)

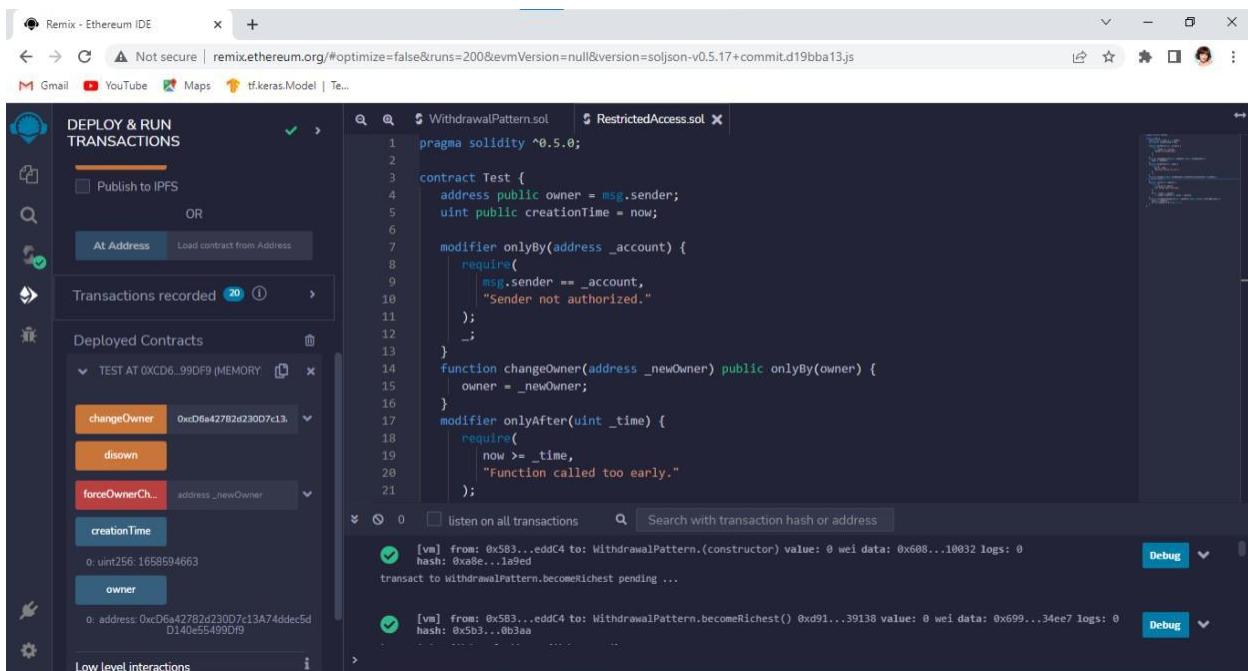
- changeOwner address _newOwner
- disown
- forceOwnerCh... address _newOwner
- creationTime
- owner

Low level interactions

CALLDATA

Search with transaction hash or address

```
25 | delete owner;
26 |
27 | modifier costs(uint _amount) {
28 |   require(
29 |     msg.value >= _amount,
30 |     "Not enough Ether provided."
31 |   );
32 |   if (msg.value > _amount)
33 |     msg.sender.transfer(msg.value - _amount);
34 |
35 |   function forceOwnerChange(address _newOwner) public payable costs(200 ether) {
36 |     owner = _newOwner;
37 |     if (uint(owner) & 0 == 1) return;
38 |   }
39 |
40 }
```



DEPLOY & RUN TRANSACTIONS

Transactions recorded 20 ⓘ

Deployed Contracts

TEST AT 0xCD6...990F9 (MEMORY)

- changeOwner 0xD6a42782d230D7c13A74dec5d
- disown
- forceOwnerCh... address _newOwner
- creationTime
- owner

Low level interactions

Search with transaction hash or address

```
1 pragma solidity ^0.5.0;
2
3 contract Test {
4   address public owner = msg.sender;
5   uint public creationTime = now;
6
7   modifier onlyBy(address _account) {
8     require(
9       msg.sender == _account,
10      "Sender not authorized."
11    );
12  }
13 }
14 function changeOwner(address _newOwner) public onlyBy(owner) {
15   owner = _newOwner;
16 }
17 modifier onlyAfter(uint _time) {
18   require(
19     now >= _time,
20     "Function called too early."
21   );
22 }
```

[vm] from: 0x5B3...edd4 to: WithdrawalPattern.(constructor) value: 0 wei data: 0x608...10032 logs: 0
hash: 0xa8e...la9ed
transact to withdrawalPattern.becomeRichest pending ...

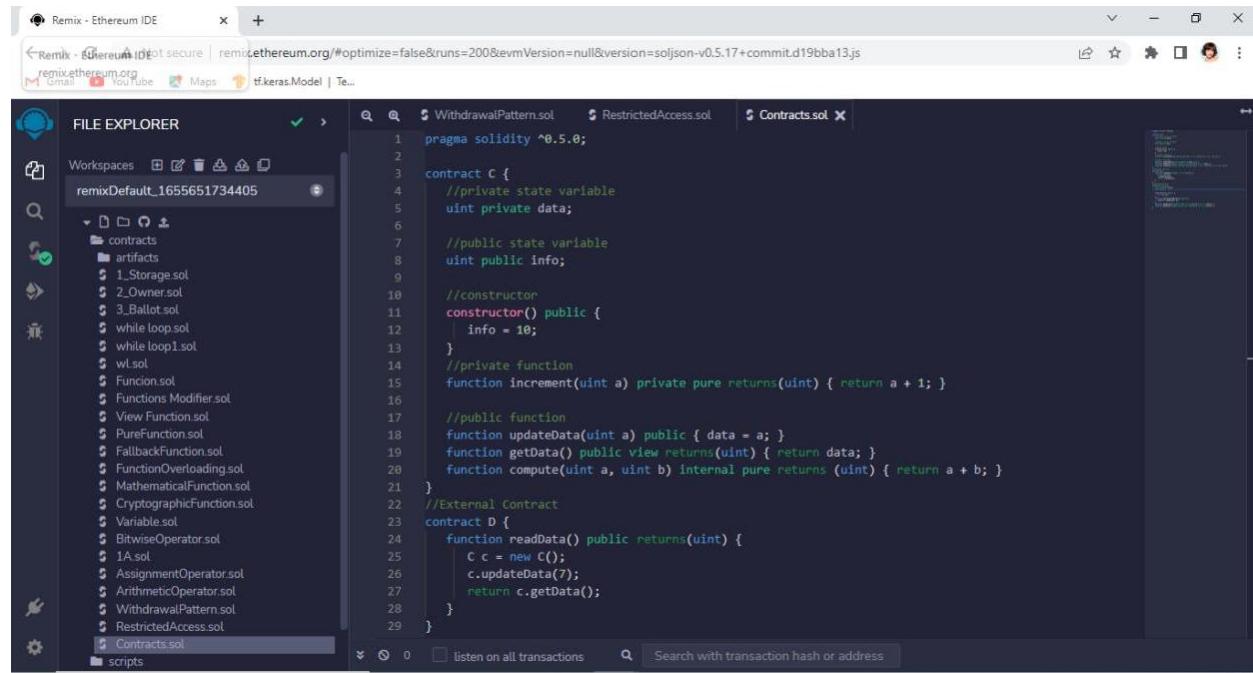
[vm] from: 0x5B3...edd4 to: WithdrawalPattern.becomeRichest() 0xd91...39138 value: 0 wei data: 0x699...34ee7 logs: 0
hash: 0xb53...0b3aa

Debug

3b) Contracts, Inheritance, Constructors, Abstract Contracts, Interfaces.

Contracts

Code:



The screenshot shows the Remix IDE interface with the Solidity code for Contracts.sol. The code defines two contracts, C and D. Contract C has a private state variable 'data' and a public state variable 'info'. It includes a constructor that initializes 'info' to 10, and a private function 'increment' that returns the value of 'a' plus 1. It also has a public function 'updateData' that sets 'data' to 'a', and two view functions 'getData' and 'compute' that return 'data' and the sum of 'a' and 'b' respectively. Contract D is an external contract that inherits from C. It contains a constructor that creates a new instance of C and initializes its 'data' to 7. It also includes a public view function 'getComputedResult' that returns the result of calling the 'compute' function on the inherited C contract with inputs 3 and 5.

```
pragma solidity ^0.5.0;

contract C {
    //private state variable
    uint private data;

    //public state variable
    uint public info;

    //constructor
    constructor() public {
        info = 10;
    }

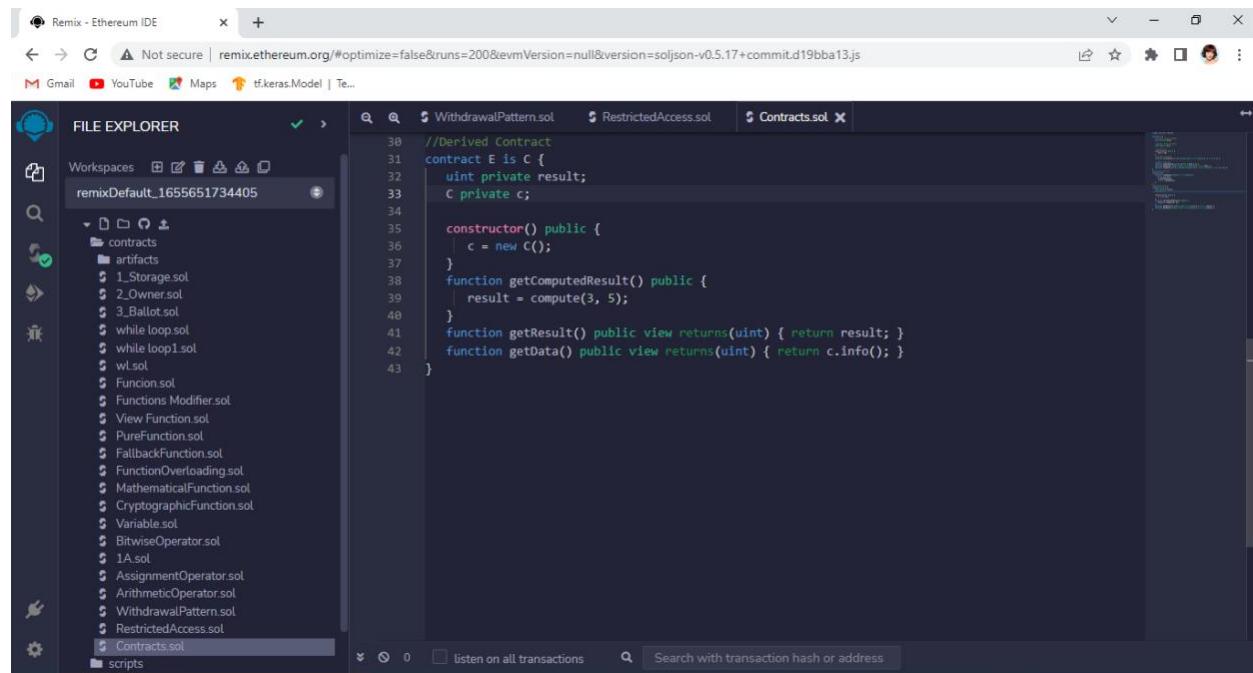
    //private function
    function increment(uint a) private pure returns(uint) { return a + 1; }

    //public function
    function updateData(uint a) public { data = a; }

    function getData() public view returns(uint) { return data; }

    function compute(uint a, uint b) internal pure returns (uint) { return a + b; }
}

//External Contract
contract D {
    function readData() public returns(uint) {
        C c = new C();
        c.updateData(7);
        return c.getData();
    }
}
```



The screenshot shows the Remix IDE interface with the Solidity code for Contracts.sol, which now includes inheritance. Contract D now inherits from Contract C. The code is identical to the previous version, except for the addition of the inheritance declaration 'contract D is C;' in the definition of Contract D.

```
//Derived Contract
contract D is C {
    uint private result;
    C private c;

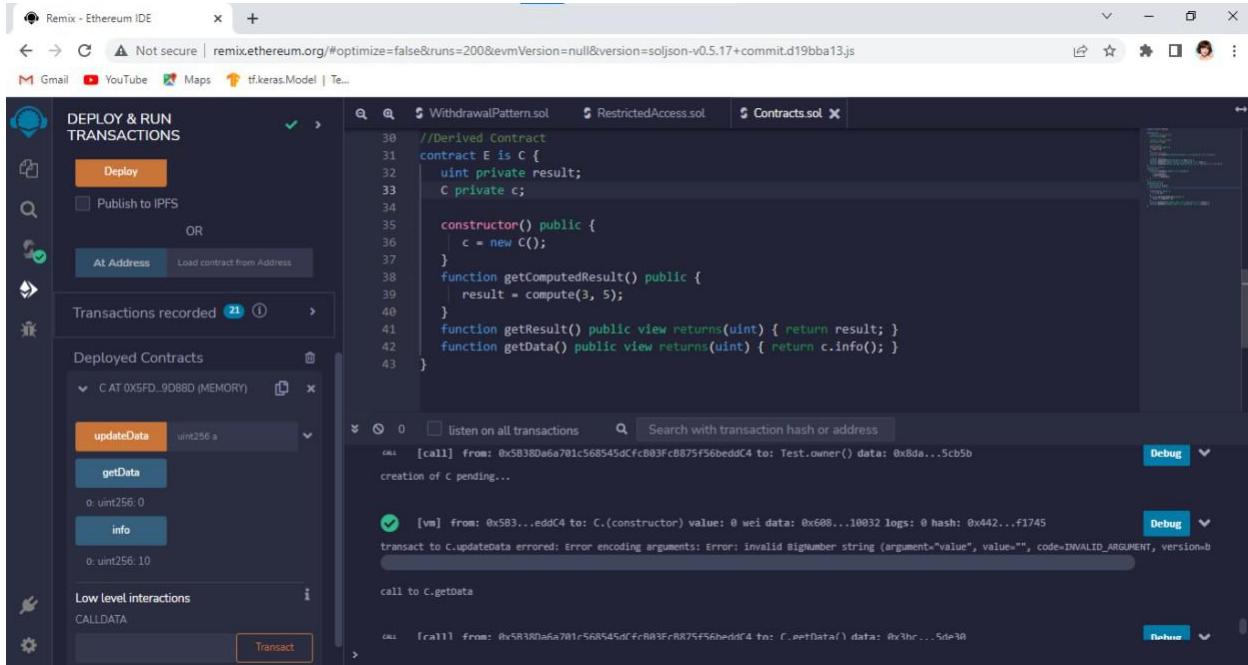
    constructor() public {
        c = new C();
    }

    function getComputedResult() public {
        result = compute(3, 5);
    }

    function getResult() public view returns(uint) { return result; }

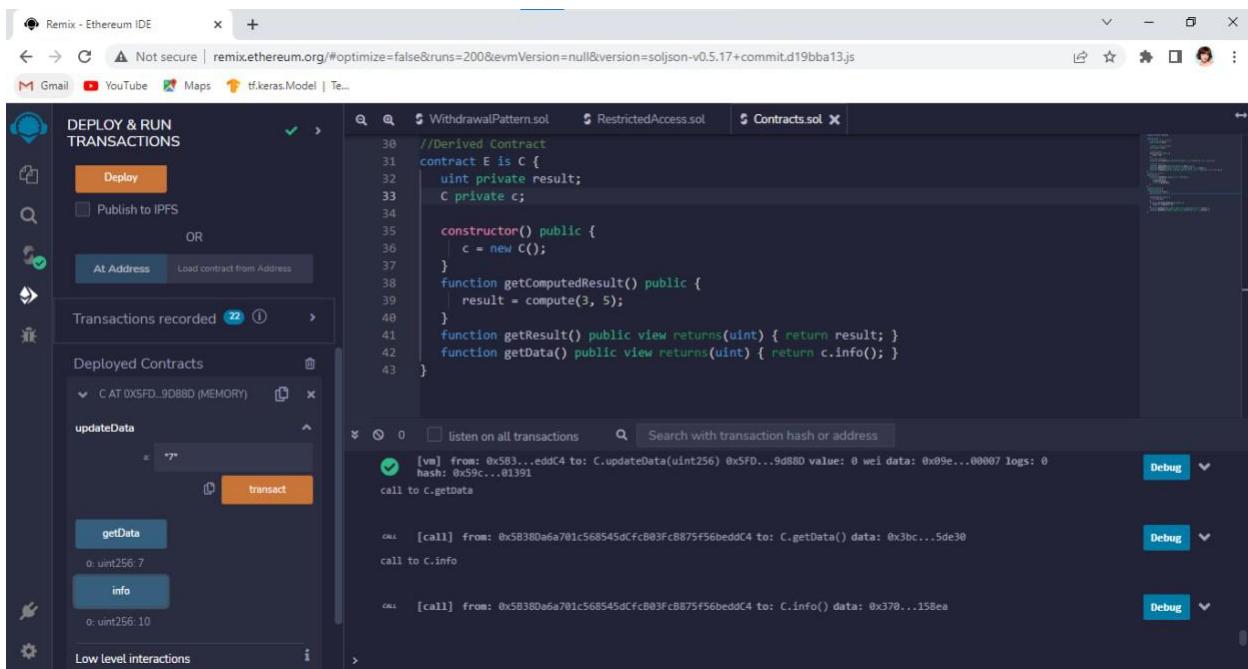
    function getData() public view returns(uint) { return c.info(); }
}
```

Output:



```
30 //Derived Contract
31 contract E is C {
32     uint private result;
33     C private c;
34
35     constructor() public {
36         c = new C();
37     }
38     function getComputedResult() public {
39         result = compute(3, 5);
40     }
41     function getResult() public view returns(uint) { return result; }
42     function getData() public view returns(uint) { return c.info(); }
43 }
```

The Remix IDE interface includes a sidebar for "DEPLOY & RUN TRANSACTIONS" with options like Deploy, Publish to IPFS, At Address, and Load contract from Address. Below this is a "Transactions recorded" section showing 21 entries. A "Deployed Contracts" section shows "C AT 0x5FD...9D88D (MEMORY)". Under "C", there are buttons for updateData, getData, info, and Low level interactions (CALLDATA). The "updateData" button is highlighted. The "Low level interactions" section has a "Transact" button.



```
30 //Derived Contract
31 contract E is C {
32     uint private result;
33     C private c;
34
35     constructor() public {
36         c = new C();
37     }
38     function getComputedResult() public {
39         result = compute(3, 5);
40     }
41     function getResult() public view returns(uint) { return result; }
42     function getData() public view returns(uint) { return c.info(); }
43 }
```

The Remix IDE interface is identical to the first screenshot, showing the same deployed contract C at address 0x5FD...9D88D. The transaction history and interaction buttons are also the same.

Inheritance

Code:

```
pragma solidity ^0.5.0;

contract C {
    //private state variable
    uint private data;

    //public state variable
    uint public info;

    //constructor
    constructor() public {
        info = 20;
    }

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

    //public function
    function updateData(uint a) public { data = a; }
    function getData() public view returns(uint) { return data; }
    function compute(uint a, uint b) internal pure returns (uint) { return a + b; }
}

//Derived Contract
contract E is C {
    uint private result;
    C private c;
    constructor() public {
        c = new C();
    }

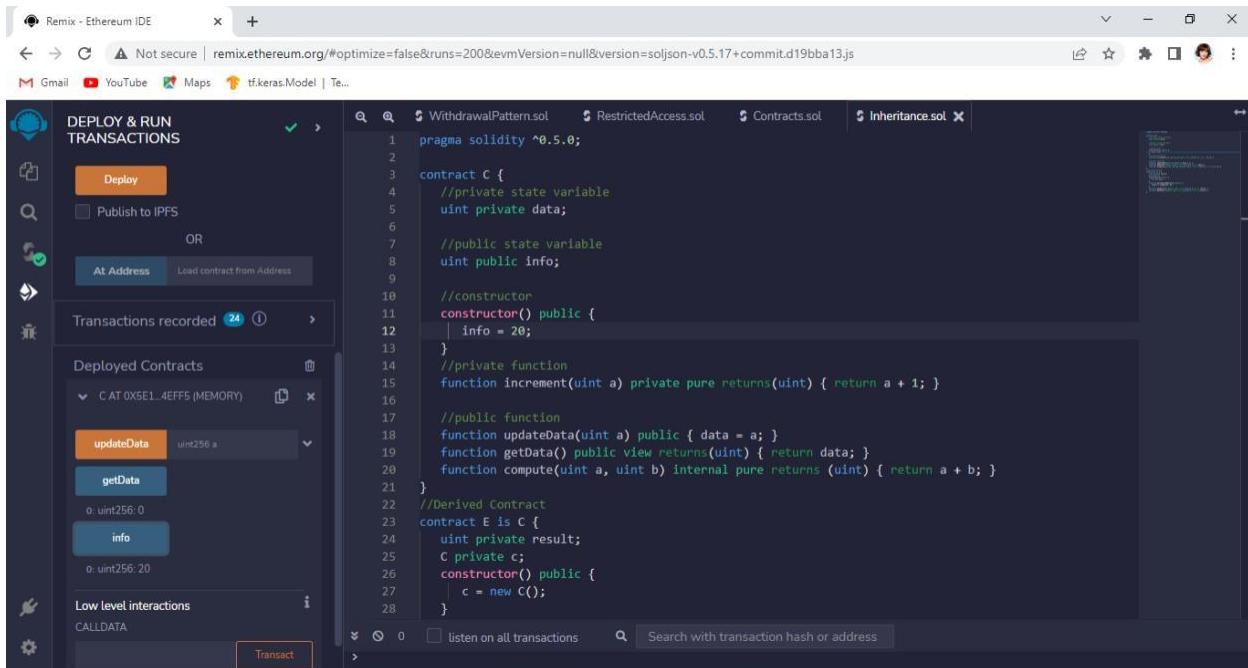
    function getComputedResult() public {
        result = compute(3, 5);
    }

    function getResult() public view returns(uint) { return result; }

    function getData() public view returns(uint) { return c.info(); }
}
```

```
function increment(uint a) public override {
    require(a > 0);
    return a + 1;
}
```

Output:

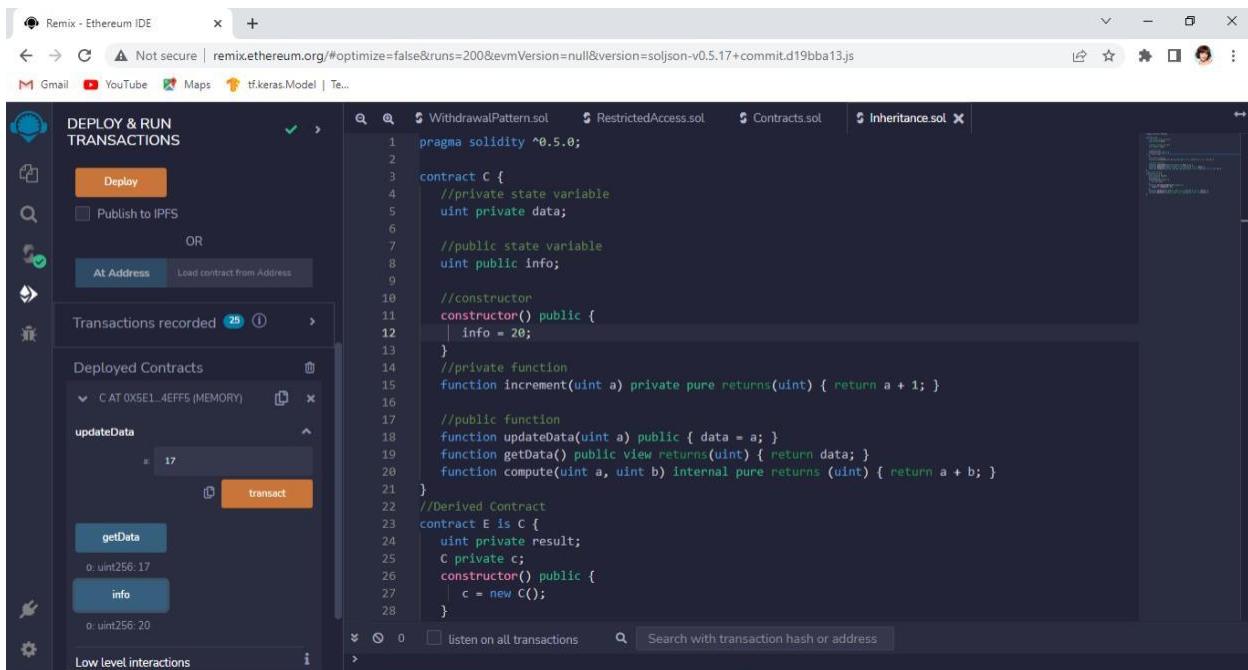


The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar titled "DEPLOY & RUN TRANSACTIONS" with buttons for "Deploy", "Publish to IPFS", and "At Address". Below these are sections for "Transactions recorded" (24), "Deployed Contracts" (C AT 0X5E1_4EFF5 (MEMORY)), and "Low level interactions" (CALLDATA). The main area contains the Solidity code for Contract C and Contract E. The code includes private state variables, public state variables, and functions like increment, updateData, and compute.

```
pragma solidity ^0.5.0;

contract C {
    //private state variable
    uint private data;
    //public state variable
    uint public info;
    //constructor
    constructor() public {
        info = 20;
    }
    //private function
    function increment(uint a) private pure returns(uint) { return a + 1; }
    //public function
    function updateData(uint a) public { data = a; }
    function getData() public view returns(uint) { return data; }
    function compute(uint a, uint b) internal pure returns (uint) { return a + b; }
}

//Derived Contract
contract E is C {
    uint private result;
    C private c;
    constructor() public {
        c = new C();
    }
}
```



This screenshot is similar to the first one but shows a transaction being initiated. The "updateData" button has been clicked, and a transaction hash "0x17" is displayed next to it. A "transact" button is visible below the transaction details. The rest of the interface and code are identical to the first screenshot.

```
pragma solidity ^0.5.0;

contract C {
    //private state variable
    uint private data;
    //public state variable
    uint public info;
    //constructor
    constructor() public {
        info = 20;
    }
    //private function
    function increment(uint a) private pure returns(uint) { return a + 1; }
    //public function
    function updateData(uint a) public { data = a; }
    function getData() public view returns(uint) { return data; }
    function compute(uint a, uint b) internal pure returns (uint) { return a + b; }
}

//Derived Contract
contract E is C {
    uint private result;
    C private c;
    constructor() public {
        c = new C();
    }
}
```

Constructors

Code:

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORER sidebar lists various Solidity files such as contracts, artifacts, and tests. The Contracts folder contains files like 1_Storage.sol, 2_Owner.sol, 3_Ballot.sol, while loop.sol, while loop1.sol, wl.sol, Funcion.sol, Functions Modifier.sol, View Function.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, Variable.sol, BitwiseOperator.sol, I1.sol, AssignmentOperator.sol, ArithmeticOperator.sol, WithdrawalPattern.sol, RestrictedAccess.sol, Contracts.sol, Inheritance.sol, Constructors.sol, scripts, tests, and README.txt. The Constructors.sol file is currently selected and displayed in the main code editor area. The code defines a contract named constructorExample with a constructor that initializes a string str to "This is a example of Constructor". It also includes a public view function getValue() that returns the value of str.

```
pragma solidity ^0.5.0;

contract constructorExample {
    string str;

    constructor() public{
        str="This is a example of Constructor";
    }

    function getValue()
    public view returns(
        string memory)
    {
        return str;
    }
}
```

Output:

The screenshot shows the Remix Ethereum IDE interface with the Deploy & Run Transactions sidebar open. The CONTRACT field is set to constructorExample - contracts/Constructors.sol. Below it, there are options to Deploy or Publish to IPFS, and a choice between At Address or Load contract from Address. The At Address option is selected. The Deployed Contracts section shows a single entry: CONSTRUCTOREXAMPLE AT 0x1C. Under this entry, the getValue function is called, and the output is shown as "0: string: This is a example of Constructor". The Low level interactions section shows the CALLDATA for the transaction. The right side of the interface displays the same Solidity code as the previous screenshot, along with the transaction details for the creation of the contract and the call to its getValue function. The transaction hash for the creation is 0xe3c...1809c, and for the call, it is 0x58380a6a701c568545dCfcB03FcB875f56beddC4.

```
pragma solidity ^0.5.0;

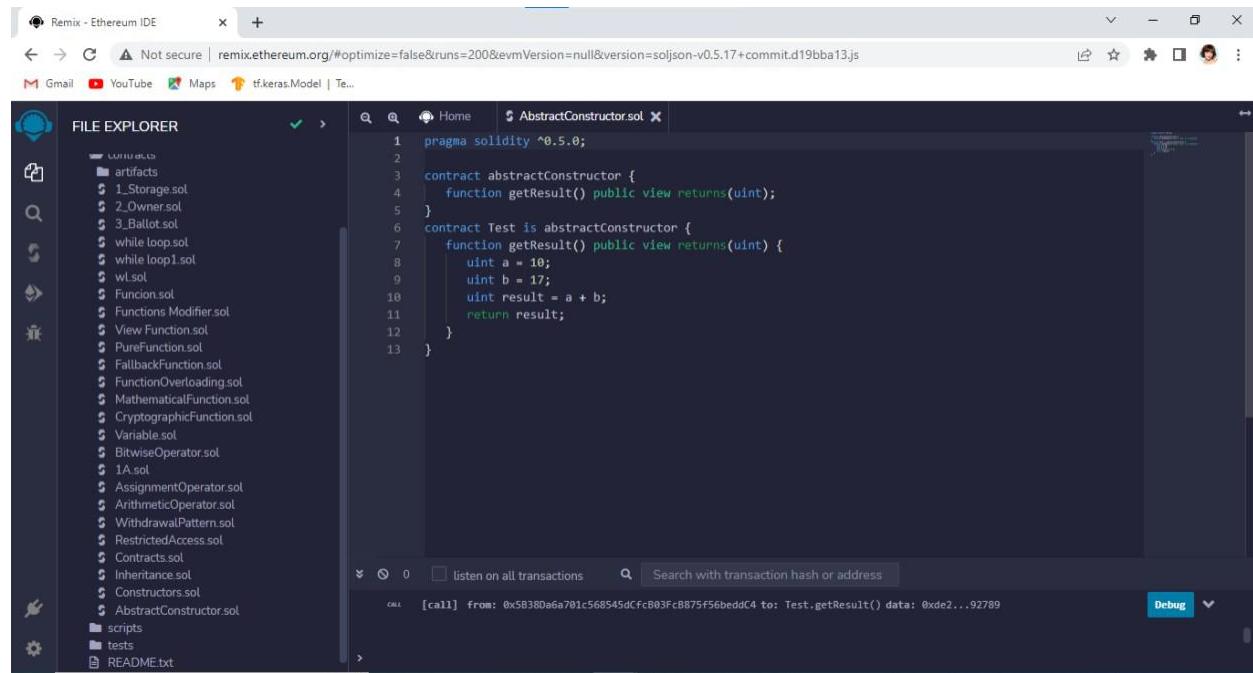
contract constructorExample {
    string str;

    constructor() public{
        str="This is a example of Constructor";
    }

    function getValue()
    public view returns(
        string memory)
    {
        return str;
    }
}
```

Abstract Contracts

Code:



The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several Solidity files under the "contracts" folder, including 1_Storage.sol, 2_Owner.sol, 3_Ballot.sol, while_loop.sol, while_loop1.sol, wl.sol, Funcion.sol, Functions.Modifier.sol, View.Function.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, Variable.sol, BitwiseOperator.sol, IA.sol, AssignmentOperator.sol, ArithmeticOperator.sol, WithdrawalPattern.sol, RestrictedAccess.sol, Contracts.sol, Inheritance.sol, Constructors.sol, AbstractConstructor.sol, scripts, tests, and README.txt. The main editor window displays the "AbstractConstructor.sol" file with the following code:

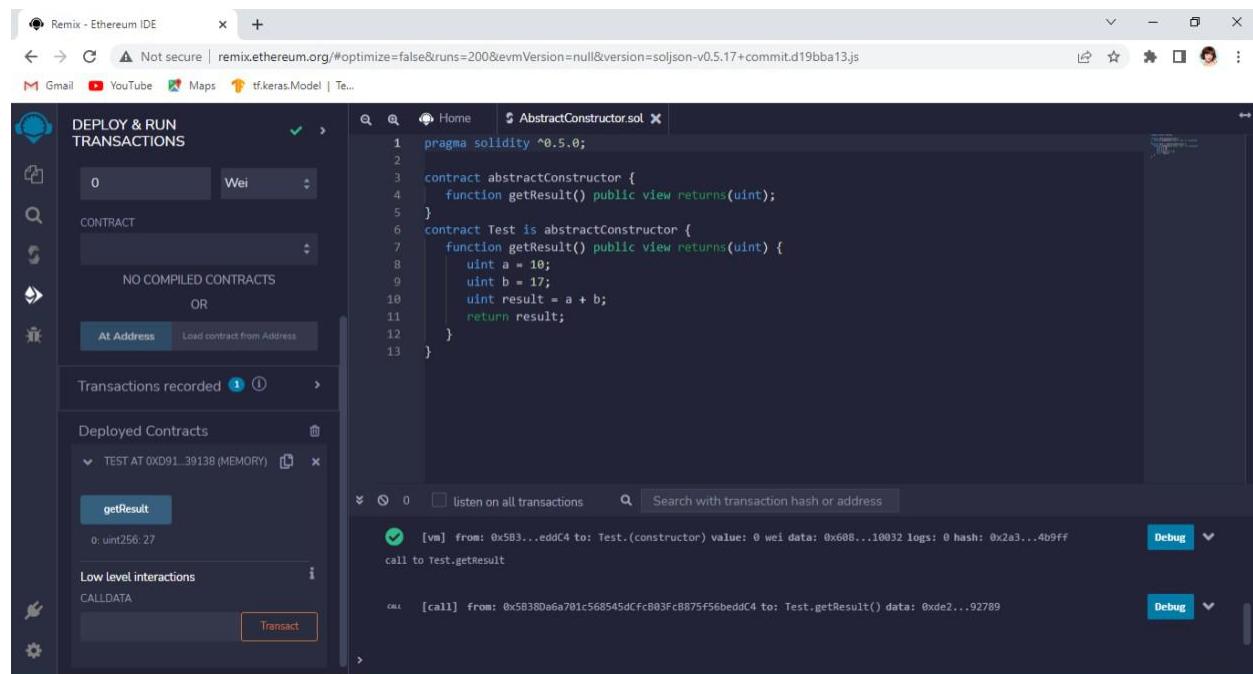
```
pragma solidity ^0.5.0;

contract abstractConstructor {
    function getResult() public view returns(uint);
}

contract Test is abstractConstructor {
    function getResult() public view returns(uint) {
        uint a = 10;
        uint b = 17;
        uint result = a + b;
        return result;
    }
}
```

The bottom right corner of the editor has a "Debug" button.

Output:



The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" tab selected. The left sidebar shows a "TRANSACTIONS" section with a "Wei" dropdown set to 0, and a "CONTRACT" section indicating "NO COMPILED CONTRACTS". Below these are buttons for "At Address" and "Load contract from Address". The main editor window shows the same "AbstractConstructor.sol" code as before. The bottom right corner of the editor has a "Debug" button.

The bottom half of the interface shows the transaction history and logs. It includes sections for "Transactions recorded" and "Deployed Contracts". Under "Deployed Contracts", there is an entry for "TEST AT 0xD91...39I3B (MEMORY)". The "getResult" function is highlighted, showing its parameters: "0: uint256: 27". The "Low level interactions" section shows a "CALLDATA" input field and a "Transact" button.

The logs pane shows two entries:

- [vm] from: 0x5B3...eddC4 to: Test.(constructor) value: 0 wei data: 0x600...10032 logs: 0 hash: 0x2a3...40ff
- [call] from: 0x5B30a6a701c568545dCfcB03FcB875f56beddC4 to: Test.getResult() data: 0xde2...92789

The bottom right corner of the logs pane has a "Debug" button.

Interfaces

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several Solidity files under the "CONTRACTS" folder, including 1_Storage.sol, 2_Owner.sol, 3_Ballot.sol, while_loop.sol, while_loop1.sol, wl.sol, Funcion.sol, Functions.Modifier.sol, View.Function.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, Variable.sol, BitwiseOperator.sol, I.sol, AssignmentOperator.sol, ArithmeticOperator.sol, WithdrawalPattern.sol, RestrictedAccess.sol, Contracts.sol, Inheritance.sol, Constructors.sol, AbstractConstructor.sol, Interfaces.sol, scripts, and tests.

The main editor window displays the following Solidity code:

```
pragma solidity ^0.5.0;

interface Interface {
    function getResult() external view returns(uint);
}

contract Test is Interface {
    constructor() public {}
    function getResult() external view returns(uint){
        uint a = 11;
        uint b = 67;
        uint result = a + b;
        return result;
    }
}
```

The bottom status bar shows transaction details: "[vm] from: 0x583...edd4 to: Test.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x137...54395 call to Test.getResult".

Output:

The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" sidebar open. The "CONTRACT" dropdown is set to "Test - contracts/Interfaces.sol". The "Deploy" button is highlighted in orange. Below it, there are options for "Publish to IPFS" and "At Address". The "Transactions recorded" section shows one entry: "TEST AT 0x08B...33FAB (MEMORY)". Under "Deployed Contracts", "TEST AT 0x08B...33FAB" is listed. The "getResult" button is highlighted in blue. The "Low level interactions" section shows a "TRANSACTION" button.

The main editor window displays the same Solidity code as the previous screenshot.

The bottom status bar shows two transaction details: "[vm] from: 0x583...edd4 to: Test.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x137...54395 call to Test.getResult" and "[call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: Test.getResult() data: 0xde2...92789".

3c) Libraries, Assembly, Events, Error handling.

Libraries

Code:

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORER sidebar lists several Solidity files: 2_Owner.sol, 3_Ballot.sol, while.loop.sol, while.loop1.sol, wl.sol, Funcion.sol, Functions.Modifier.sol, View.Function.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, Variable.sol, BitwiseOperator.sol, I1.sol, AssignmentOperator.sol, ArithmeticOperator.sol, WithdrawalPattern.sol, RestrictedAccess.sol, Contracts.sol, Inheritance.sol, Constructors.sol, AbstractConstructor.sol, Interfaces.sol. The file Libraries.sol is currently selected. The main editor area displays the following Solidity code:

```
pragma solidity ^0.5.0;

library Search {
    function indexOf(uint[] storage self, uint value) public view returns (uint) {
        for (uint i = 0; i < self.length; i++) if (self[i] == value) return i;
        return uint(-1);
    }
}

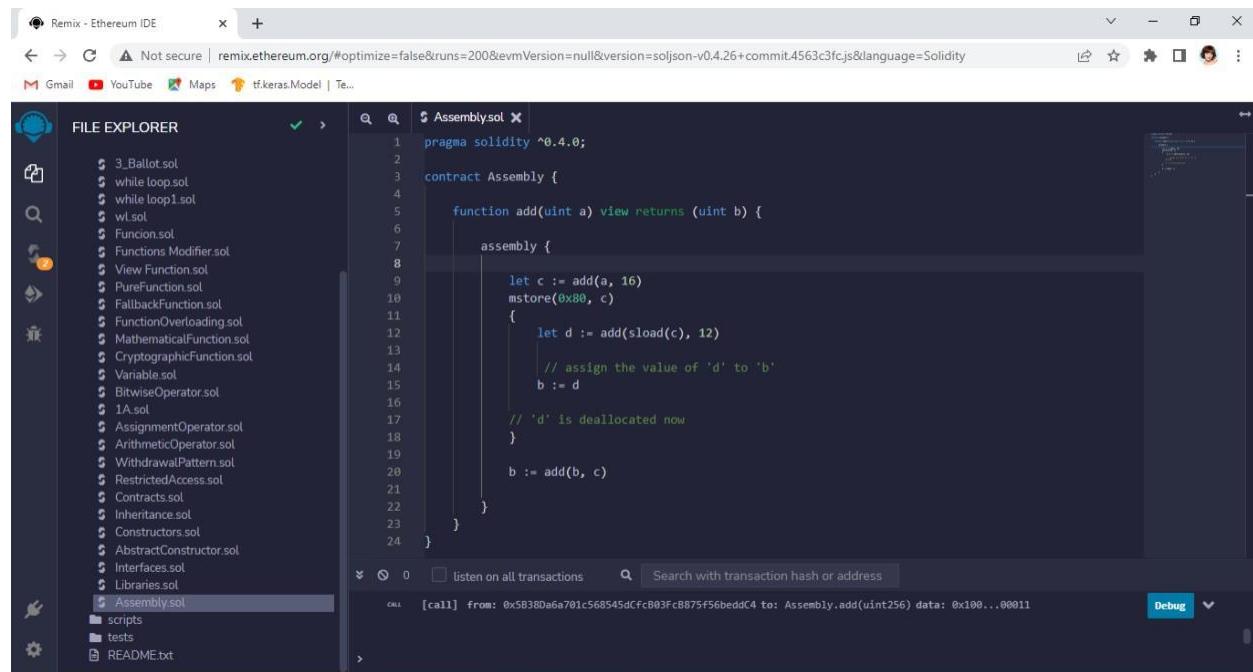
contract Libraries {
    uint[] data;
    constructor() public {
        data.push(1);
        data.push(2);
        data.push(3);
        data.push(4);
        data.push(5);
    }
    function isValuePresent() external view returns(uint){
        uint value = 4;
        //search if value is present in the array using Library function
        uint index = Search.indexOf(data, value);
        return index;
    }
}
```

Output:

The screenshot shows the Remix Ethereum IDE interface with the DEPLOY & RUN TRANSACTIONS tab active. The CONTRACT section shows "Libraries - contracts/Libraries.sol" selected. Below it, there is a Deploy button and an option to "Publish to IPFS". The Transactions recorded section shows a deployed contract named "LIBRARIES AT 0X358...D5EE3 (METH)". The Low level interactions section shows a transaction with the value "0: uint256: 3". The right side of the screen shows the same Solidity code as the previous screenshot. A log message at the bottom indicates a successful deployment: "[vm] from: 0x583...eddC4 to: Libraries.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x67f...dc766".

Assembly

Code:



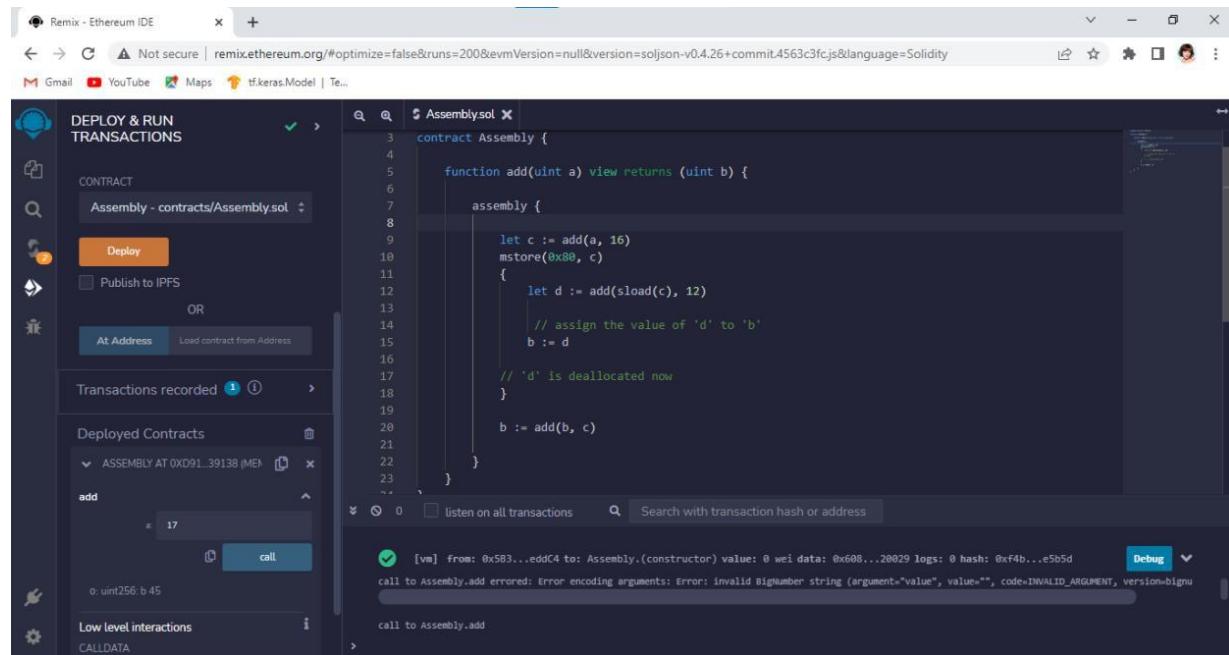
The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several Solidity files. The main editor window is titled "Assembly.sol" and contains the following Solidity code:

```
pragma solidity ^0.4.0;

contract Assembly {
    function add(uint a) view returns (uint b) {
        assembly {
            let c := add(a, 16)
            mstore(0x0, c)
            {
                let d := add(sload(c), 12)
                // assign the value of 'd' to 'b'
                b := d
            }
            // 'd' is deallocated now
        }
        b := add(b, c)
    }
}
```

The status bar at the bottom shows a call from address 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to the contract's constructor, with data 0x100...00011.

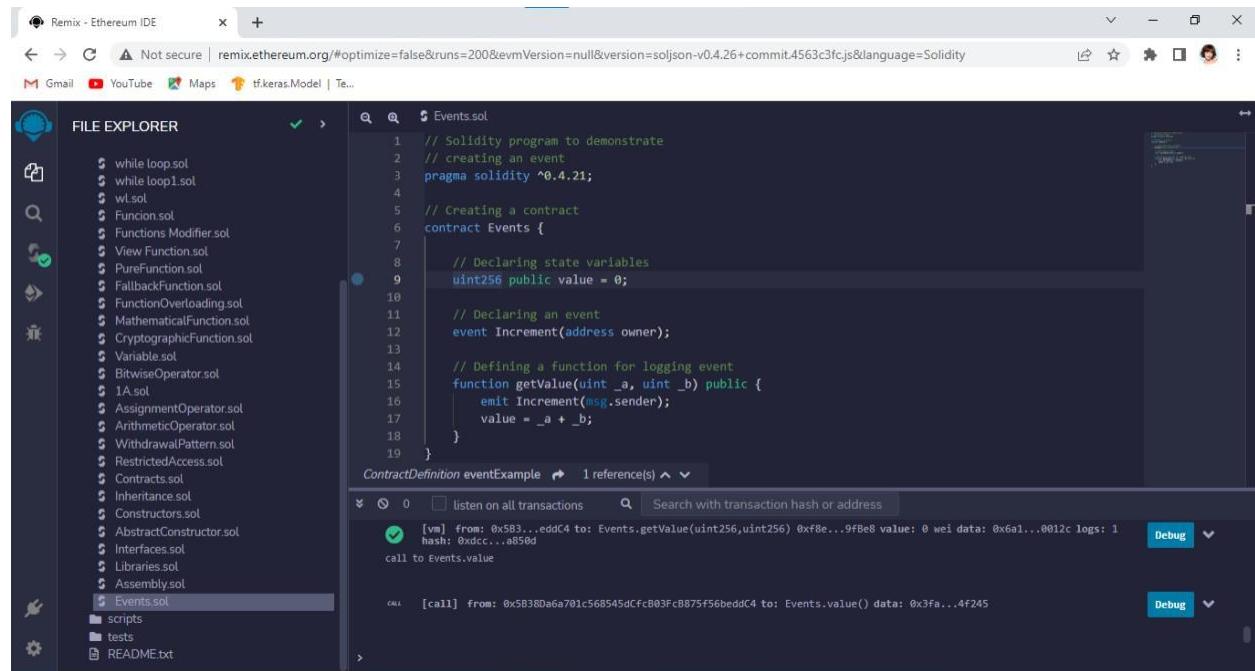
Output:



The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" tab selected. The "CONTRACT" dropdown is set to "Assembly - contracts/Assembly.sol". The "Deploy" button is highlighted. The "Transactions recorded" section shows one transaction. The "Deployed Contracts" section shows the contract "ASSEMBLY AT 0xD91...39138 (METH)". The "add" function is selected, and the "call" button is highlighted. The status bar at the bottom shows a call to the constructor with error message: "call to Assembly.add errored: Error encoding arguments: Error: invalid BigNumber string (argument='value', value='', code=INVALID_ARGUMENT, version=bignumber@2.3.1)".

Events

Code:



The screenshot shows the Ethereum IDE interface with the FILE EXPLORER on the left displaying various Solidity files. The main editor window shows the code for `Events.sol`:

```
// Solidity program to demonstrate
// creating an event
pragma solidity ^0.4.21;

// Creating a contract
contract Events {

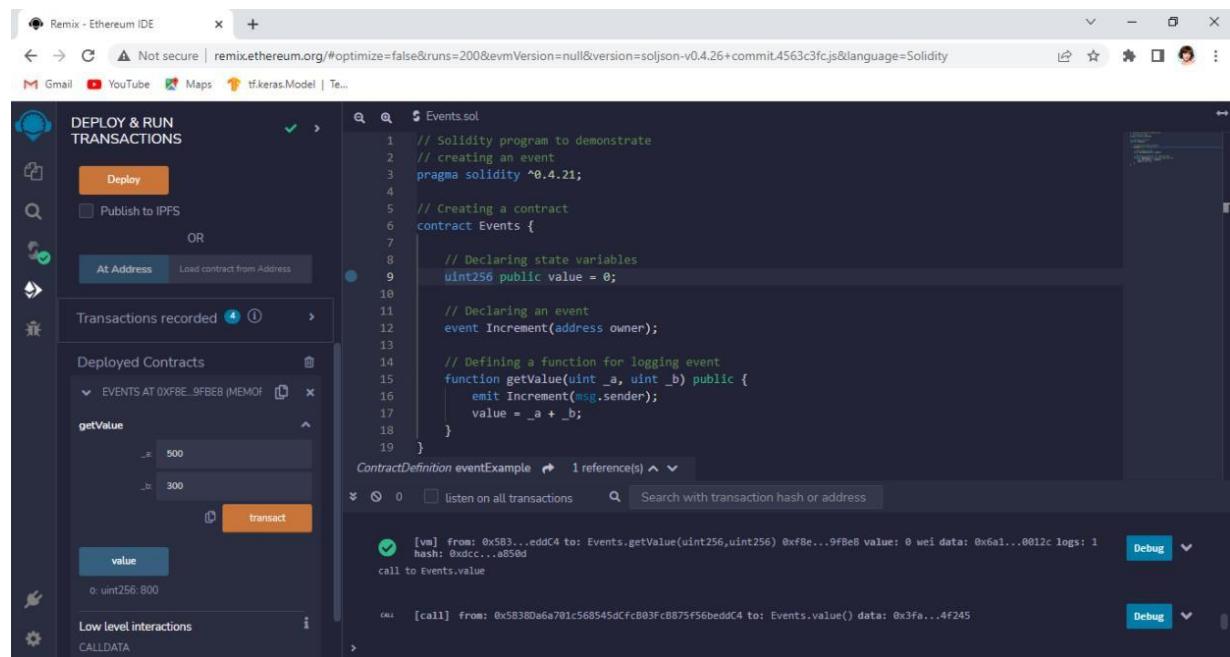
    // Declaring state variables
    uint256 public value = 0;

    // Declaring an event
    event Increment(address owner);

    // Defining a function for logging event
    function getValue(uint _a, uint _b) public {
        emit Increment(msg.sender);
        value = _a + _b;
    }
}
```

The code defines a contract named `Events` with a state variable `value` and an event `Increment`. It also contains a function `getValue` that emits the `Increment` event and updates the `value` state variable.

Output:

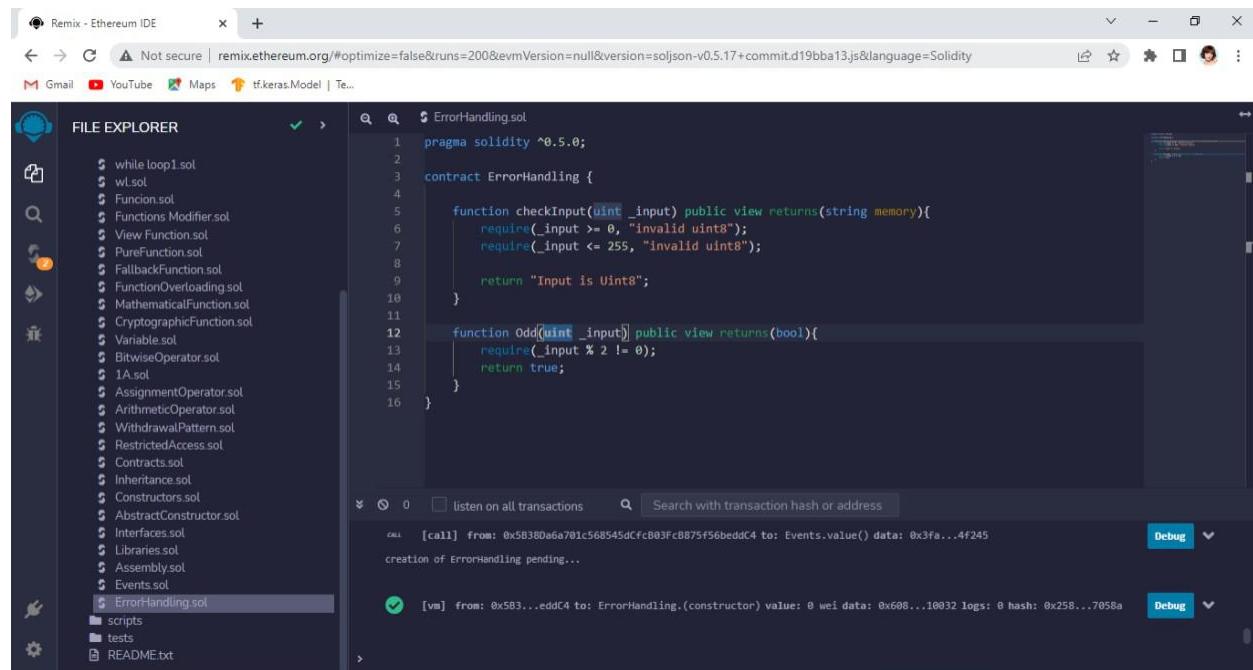


The screenshot shows the Ethereum IDE interface with the DEPLOY & RUN TRANSACTIONS panel on the left. The main editor window shows the same `Events.sol` code as before.

In the Deploy & Run Transactions panel, the "Deploy" button is highlighted. Below it, there are fields for "At Address" and "Load contract from Address". The "Transactions recorded" section shows a deployed contract named `EVENTS AT 0XFBE...9FBEB (MEMO)`. A transaction is being prepared to call the `getValue` function with parameters `_a: 500` and `_b: 300`. The "value" field shows the result of the transaction as `0: uint256: 800`.

Error Handling

Code:



The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORER" and lists several Solidity files, with "ErrorHandling.sol" selected. The main editor area displays the following Solidity code:

```
pragma solidity ^0.5.0;

contract ErrorHandling {

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

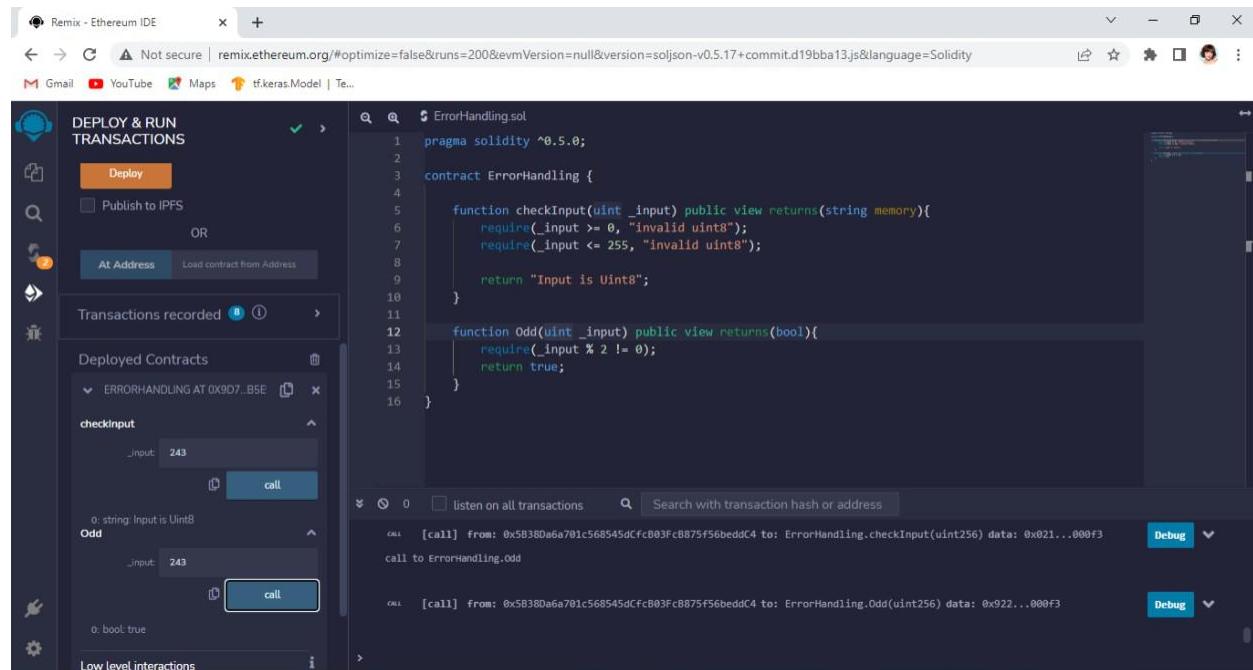
        return "Input is Uint8";
    }

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

The bottom pane shows transaction logs. One log entry is highlighted in green:

```
[vm] From: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: ErrorHandling.(constructor) value: 0 wei data: 0x60...10032 logs: 0 hash: 0x258...7958a
```

Output:



The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" sidebar open. It displays the "Deploy" button and the "At Address" field containing the address "ERRORHANDLING AT 0x907..B5E". Below this, the "Transactions recorded" section shows two interactions:

- A call to the "checkInput" function with input "243". The output is "0: string: Input is Uint8".
- A call to the "Odd" function with input "243". The output is "0: bool: true".

The main editor area shows the same Solidity code as the previous screenshot. The bottom pane shows transaction logs, with the second log entry highlighted in green:

```
[call] From: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: ErrorHandling.checkInput(uint256) data: 0x021...000F3
```

Practical No: 4

Aim: Install hyperledger fabric and composer. Deploy and execute the application.

Program Steps:

The following are prerequisites for installing the required development tools:

- Operating Systems: Ubuntu Linux 14.04 / 16.04 LTS (both 64-bit), or Mac OS 10.12
- Docker Engine: Version 17.03 or higher
- Docker-Compose: Version 1.8 or higher
- Node: 8.9 or higher (Note: version 9 is not supported)
- npm: v5.x
- git: 2.9.x or higher
- Python: 2.7.x
- A code editor of your choice, we recommend VSCode.

If installing Hyperledger Composer using Linux, the following points need to be kept in mind:

- Login as a normal user, rather than root.
- Do not use sudo su to root.
- When installing prerequisites, use curl, then unzip using sudo.
- Run prereqs-ubuntu.sh as a normal user. It may prompt for root password as some of its actions are required to be run as root.
- Do not use npm with sudo or su to root to use it.
- Avoid installing node globally as root.

Prerequisites

To download prerequisites for Hyperledger Fabric development, run following command –

```
| curl -O https://hyperledger.github.io/composer/latest/prereqs-ubuntu.sh
```

This command will download and install -

```
| docker-compose , docker-engine, npm, git, python etc
```

- Give permissions — chmod u+x prereqs-ubuntu.sh
- Run Script — ./prereqs-ubuntu.sh (restart system after it)
- Essential CLI tools — npm install -g [composer-cli@0.20](#)

Steps

1. Create a directory — mkdir multichain_network

```
cd multichain_network

curl -sSL http://bit.ly/2ysbOFE | bash -s 1.2.0

export PATH=<path to download location>/multichain_network/fabric-
samples/first-network/bin:$PATH
```

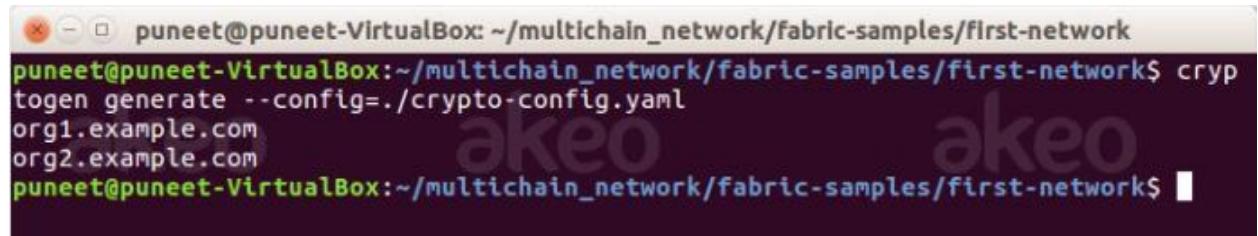
2. Generate Certificates

```
cd first-network

export FABRIC_CFG_PATH=$PWD

cryptogen generate --config=./crypto-config.yaml
```

This will create all certificates for orderers and peers in crypto-config folder.



```
puneet@puneet-VirtualBox:~/multichain_network/fabric-samples/first-network
puneet@puneet-VirtualBox:~/multichain_network/fabric-samples/first-network$ cryp
togen generate --config=./crypto-config.yaml
org1.example.com
org2.example.com
puneet@puneet-VirtualBox:~/multichain_network/fabric-samples/first-network$
```

Copy certificates for all peers and orderer to temporary folder.

```
In first-network folder run this command - mkdir -p tmp/composer/org1

awk 'NF {sub(/\r/, ""); printf "%s\\n",$0;}' ./crypto-
config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com
/tls/ca.crt > ./tmp/composer/org1/ca-org1.txt

awk 'NF {sub(/\r/, ""); printf "%s\\n",$0;}' ./crypto-
config/ordererOrganizations/example.com/orderers/orderer.example.com/t
ls/ca.crt > ./tmp/composer/ca-orderer.txt

export ORG1=./crypto-
config/peerOrganizations/org1.example.com/users/Admin@org1.example.com
/msp

cp -p $ORG1/signcerts/A*.pem ./tmp/composer/org1

cp -p $ORG1/keystore/*_sk ./tmp/composer/org1
```

3. Create genesis block and channeltx

```
configtxgen -profile TwoOrgsOrdererGenesis -outputBlock ./composer-genesis.block
```

```
puneet@puneet-VirtualBox: ~/multichain_network/fabric-samples/first-network
puneet@puneet-VirtualBox:~/multichain_network/fabric-samples/first-network$ configtxgen -profile TwoOrgsOrdererGenesis -outputBlock ./composer-genesis.block
2019-03-18 14:50:56.908 IST [common/tools/configtxgen] main -> INFO 001 Loading configuration
2019-03-18 14:50:56.924 IST [msp] getMspConfig -> INFO 002 Loading NodeOUs
2019-03-18 14:50:56.924 IST [msp] getMspConfig -> INFO 003 Loading NodeOUs
2019-03-18 14:50:56.924 IST [common/tools/configtxgen] doOutputBlock -> INFO 004 Generating genesis block
2019-03-18 14:50:56.925 IST [common/tools/configtxgen] doOutputBlock -> INFO 005 Writing genesis block
puneet@puneet-VirtualBox:~/multichain_network/fabric-samples/first-network$ █
```

```
configtxgen -profile TwoOrgsChannel -outputCreateChannelTx ./composer-channel.tx -channelID composerchannel
```

```
puneet@puneet-VirtualBox: ~/multichain_network/fabric-samples/first-network
puneet@puneet-VirtualBox:~/multichain_network/fabric-samples/first-network$ configtxgen -profile TwoOrgsChannel -outputCreateChannelTx ./composer-channel.tx -channelID composerchannel
2019-03-18 15:17:36.592 IST [common/tools/configtxgen] main -> INFO 001 Loading configuration
2019-03-18 15:17:36.612 IST [common/tools/configtxgen] doOutputChannelCreateTx -> INFO 002 Generating new channel configtx
2019-03-18 15:17:36.614 IST [msp] getMspConfig -> INFO 003 Loading NodeOUs
2019-03-18 15:17:36.616 IST [msp] getMspConfig -> INFO 004 Loading NodeOUs
2019-03-18 15:17:36.656 IST [common/tools/configtxgen] doOutputChannelCreateTx -> INFO 005 Writing new channel tx
puneet@puneet-VirtualBox:~/multichain_network/fabric-samples/first-network$ █
```

4. Update CA keys in docker composer file

Open project in code editor.

We have to change CA keys in “docker-compose-e2e-template.yaml” file, therefore navigate to this file in vscode.

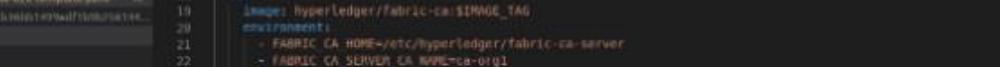
Under services section we have two certificate authorities named “ca0” and “ca1”.

For ca0 — go to command section under ca0 and find CA1_PRIVATE_KEY and replace it with private key –

c7d82435d76cb36bb1499edf1b9b256144753a458a8467ed1ff67607cef09179 sk

This key can be found in –

```
"first-network/crypto-  
config/peerOrganizations/org1.example.com/ca/c7d82435d76cb36bb1499edf1  
b9b256144753a458a8467ed1ff67607cef09179_sk"
```



```
docker compose -f docker-compose-e2e-template.yaml up -d
```

For ca1 — goto command section under ca1 and find CA2_PRIVATE_KEY and replace it with private key –

B069c3b1761b013447f7da8f1c266b26146daa0eb43d04a531f73675403b4d61 sk

This key can be found in –

```
"first-network/crypto-  
config/peerOrganizations/org1.example.com/ca/b069c3b1761b013447f7da8f1  
c266b26146daa0eb43d04a531f73675403b4d61_sk
```

Steps of Starting Hyperledger Fabric

1. Open docker-compose-base.yaml file which is present in the bin folder and introduce following changes.

```
Change orderer volume binding to -
```

```
.../composer-
genesis.block:/var/hyperledger/orderer/orderer.genesis.block
```

As shown in the figure below –

```
10  orderer.example.com:
11    container_name: orderer.example.com
12    image: hyperledger/fabric-orderer:$IMAGE_TAG
13    environment:
14      - ORDERER_GENERAL_LOGLEVEL=INFO
15      - ORDERER_GENERAL_LISTENADDRESS=0.0.0.0
16      - ORDERER_GENERAL_GENESISMETHOD=file
17      - ORDERER_GENERAL_GENESISFILE=/var/hyperledger/orderer/orderer.genesis.block
18      - ORDERER_GENERAL_LOCALMSPID=OrdererMSP
19      - ORDERER_GENERAL_LOCALMSPDIR=/var/hyperledger/orderer/msp
20    # enabled TLS
21    - ORDERER_GENERAL_TLS_ENABLED=true
22    - ORDERER_GENERAL_TLS_PRIVATEKEY=/var/hyperledger/orderer/tls/server.key
23    - ORDERER_GENERAL_TLS_CERTIFICATE=/var/hyperledger/orderer/tls/server.crt
24    - ORDERER_GENERAL_TLS_ROOTCAS=[/var/hyperledger/orderer/tls/ca.crt]
25    working_dir: /opt/gopath/src/github.com/hyperledger/fabric
26    command: orderer
27    volumes:
28      - .../composer-genesis.block:/var/hyperledger/orderer/orderer.genesis.block
29      - .../crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/msp:/var/hyperledger/orderer/msp
30      - .../crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/tls:/var/hyperledger/orderer/tls
31    ports:
32      - 7050:7050
```

Change peer volume binding to (for all 4 peers)-

```
- ...:/etc/configtx
```

```
35  peer0.org1.example.com:
36    extends:
37      file: peer-base.yaml
38      service: peer-base
39    environment:
40      - CORE_PEER_ID=peer0.org1.example.com
41      - CORE_PEER_ADDRESS=peer0.org1.example.com:7051
42      - CORE_PEER_GOSSIP_BOOTSTRAP=peer1.org1.example.com:7051
43      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org1.example.com:7051
44      - CORE_PEER_LOCALMSPID=Org1MSP
45    volumes:
46      - ...:/etc/configtx
47      - /var/run/:/host/var/run/
48      - .../crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp:/etc/hyperledger/fabric/msp
49      - .../crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls:/etc/hyperledger/fabric/tls
50    ports:
51      - 7051:7051
52      - 7053:7053
```

- To start fabric, run the following command –

```
docker-compose -f docker-compose-cli.yaml -f docker-compose-couch.yaml
up -d 2>&1
```

The output of the above command is shown below –

```
puneet@puneet-VirtualBox:~/multichain_network/fabric-samples/first-network$ docker-compose -f docker-compose-cli.yaml -f docker-compose-couch.yaml up -d
Creating orderer.example.com ...
Creating couchdb1 ...
Creating couchdb3 ...
Creating couchdb2 ...
Creating orderer.example.com
Creating couchdb0 ...
Creating couchdb2
Creating couchdb1
Creating couchdb3
Creating couchdb0 ... done
Creating peer0.org1.example.com ...
Creating couchdb3 ... done
Creating peer0.org2.example.com ...
Creating peer1.org1.example.com ...
Creating peer1.org2.example.com ...
Creating peer0.org1.example.com
Creating peer0.org2.example.com
Creating peer1.org2.example.com ... done
Creating cli ...
Creating cli ... done
```

- After completing all these steps, you can run command –

```
docker ps -a
```

This will list all running containers regarding our network setup, in our case it will list 10 containers.

CONTAINER ID PORTS	IMAGE	NAMES	COMMAND	CREATED	STATUS
badfe5ea8a92	hyperledger/fabric-peer:latest	peer1.org1.example.com	"peer node start"	4 hours ago	Up 3 hours
50d9fa8f5dfb	hyperledger/fabric-peer:latest	peer0.org2.example.com	"peer node start"	4 hours ago	Up 3 hours
ee4b501b7d00	hyperledger/fabric-peer:latest	peer0.org1.example.com	"peer node start"	4 hours ago	Up 3 hours
32f0f4426a22	hyperledger/fabric-peer:latest	peer1.org2.example.com	"peer node start"	4 hours ago	Up 3 hours
ec8f9df17258	hyperledger/fabric-couchdb	couchdb3	"tini -- /docker-ent..."	4 hours ago	Up 4 hours
2b4be64efcfe	hyperledger/fabric-couchdb	couchdb2	"tini -- /docker-ent..."	4 hours ago	Up 4 hours
f922625a027e	hyperledger/fabric-couchdb	couchdb0	"tini -- /docker-ent..."	4 hours ago	Up 4 hours
8de729e174b6	hyperledger/fabric-couchdb	couchdb1	"tini -- /docker-ent..."	4 hours ago	Up 4 hours
18830d7ccf5e	hyperledger/fabric-tools:latest	clt	"/bin/bash"	11 hours ago	Up 11 hours
c7ef15868cd5	hyperledger/fabric-orderer:latest	orderer.example.com	"orderer"	11 hours ago	Up 11 hours
			0.0.0.0:7050->7050/tcp		

Proposed network setup is complete, our network have -

- One orderer
- Two Organizations
- Four peers (two peers on each organization)

- Couchdb for all peers

4. Creating channel

```
docker exec peer0.org1.example.com peer channel create -o
orderer.example.com:7050 -c composerchannel -f /etc/configtx/composer-
channel.tx - tls true - cafile /etc/configtx/crypto-
config/ordererOrganizations/example.com/orderers/orderer.example.com/m
sp/tlscacerts/tlsca.example.com-cert.pem
```

5. Joining first peer where channel is created —

```
docker exec -e

"CORE_PEER_MSPCONFIGPATH=/etc/configtx/tmp/composer/org1/Admin@org1.ex
ample.com/msp" peer0.org1.example.com peer channel join -b composer-
genesis.block
```

6. For other peers, we have to first fetch the config of block from orderer

For Fetching –

```
docker exec -e

"CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.example
.com/msp"

peer1.org1.example.com peer channel fetch config -o
orderer.example.com:7050 -c TwoOrgsChannel
```

For joining channel –

```
docker exec -e

"CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.example
.com/msp"

peer1.org1.example.com peer channel join -b
composerchannel_config.block
```

Install chainCode on every machine

```
composer network install -a device-network.bna -c  
PeerAdmin@multi_org1
```

Start chaincode on one machine

```
composer network start -n device-network -V 0.0.2-  
deploy.${bna_deployment_version} -A admin -S adminpw -c  
PeerAdmin@multi_org1
```

Conclusion

We learnt about installing Hyperledger Fabric development tools and Hyperledger Composer using Linux. We also successfully deployed a Hyperledger Fabric network having one orderer, two organizations and two peers in each organization.

Practical No: 5

Aim: Demonstrate the running of the blockchain node.

Refer Practical No: 1, 2 3

Practical No: 6

Aim: Demonstrate the use of Bitcoin Core API.

Code:

```
from hashlib import sha256
MAX_NONCE = 100000000000

def SHA256(text):
    return sha256(text.encode("ascii")).hexdigest()

def mine(block_number, transactions, previous_hash, prefix_zeros):
    prefix_str = '0'*prefix_zeros
    for nonce in range(MAX_NONCE):
        text = str(block_number) + transactions + previous_hash + str(nonce)
        new_hash = SHA256(text)
        if new_hash.startswith(prefix_str):
            print(f"Yay! Successfully mined bitcoins with nonce value:{nonce}")
            return new_hash
    raise BaseException(f"Couldn't find correct has after trying {MAX_NONCE} times")

if __name__=='__main__':
    transactions='''
    Dhaval->Bhavin->20,
    Mando->Cara->45
    '''
    difficulty=4 # try changing this to higher number and you will see it
    will take more time for mining as difficulty increases
    import time
    start = time.time()
    print("start mining")
```

```
new_hash = mine(5,transactions,'0000000xa036944e29568d0cff17edbe038f81  
208fecf9a66be9a2b8321c6ec7', difficulty)  
total_time = str((time.time() - start))  
print(f"end mining. Mining took: {total_time} seconds")  
print(new_hash)
```

Output:

```
start mining  
Yay! Successfully mined bitcoins with nonce value:2425  
end mining. Mining took: 0.011358022689819336 seconds  
0000de957fbfdfc77582e0d0b20c53d2d1d83d8bb8cfe3693521f672bf2a6021
```

Practical No: 7

Aim: Create your own blockchain and demonstrate its use.

Refer Practical No: 1, 2 3