

INDEX

Sr. No.	Practical	SIGN
1	<p>Write the following programs for Blockchain in Python:</p> <ul style="list-style-type: none"> i) A simple client class that generates the private and public keys by using the built-in Python RSA algorithm and test it. ii) A transaction class to send and receive money and test it. iii) Create multiple transactions and display them. iv) Create a blockchain, a genesis block and execute it. v) Create a mining function and test it. vi) Add blocks to the miner and dump the blockchain. 	
2	<p>Implement and demonstrate the use of the following in Solidity:</p> <ul style="list-style-type: none"> i) Variable, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs, Mappings, Conversions, Ether Units, Special Variables. ii) Functions, Function Modifiers, View functions, Pure Functions, Fallback Function, Function Overloading, Mathematical functions, Cryptographic functions. 	
3	<p>Implement and demonstrate the use of the following in Solidity:</p> <ul style="list-style-type: none"> i) Withdrawal Pattern, Restricted Access. ii) Contracts, Inheritance, Constructors, Abstract Contracts, Interfaces. iii) Libraries, Assembly, Events, Error handling. 	

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:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the "BLOCKCHAIN" folder, including `pract1a.py`, `pract1b.py`, `pract1c.py`, `pract1d.py`, `pract1e.py`, and `pract1f.py`.
- Code Editor:** The `pract1a.py` file contains the following Python code:

```
pract1a.py > ...
1 # following imports are required by PKI
2 from Crypto.PublicKey import RSA
3
4 key = RSA.generate(2048)
5 p_key = key.public_key().export_key("PEM")
6 priv_key = key.export_key("PEM")
7 print("7_GovindSaini \n")
8 print(p_key)
9 print(priv_key)
10
11
```

- Terminal:** The terminal shows the command `admin@DESKTOP-3B61180 MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN` and the output of running the script.
- Status Bar:** Shows the date and time as 18-03-2022, and the Python version as 3.9.7 (venv: venv).

Output :

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the "BLOCKCHAIN" folder, including `pract1a.py`, `pract1b.py`, `pract1c.py`, `pract1d.py`, `pract1e.py`, and `pract1f.py`.
- Code Editor:** The `pract1a.py` file contains the same Python code as before.
- Terminal:** The terminal shows the command `$ py pract1a.py` and the output of the RSA key generation process, including the public and private keys.
- Status Bar:** Shows the date and time as 18-03-2022, and the Python version as 3.9.7 (venv: venv).

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

Code:

The screenshot shows the Visual Studio Code interface with the file 'pract1b.py' open. The code defines a 'Bank' class with methods for deposit, withdraw, and enquiry. It also creates an instance of the class and calls its methods. The code is as follows:

```
class Bank:
    def __init__(self):
        self.balance = 0
        print("7_GovindSaini \n")
        print ("The account is created")

    def deposit(self):
        amount = float(input("Enter the amount to be deposit: "))
        self.balance = self.balance + amount
        print ("The deposit is successful and the balance in the account is %f" % self.balance)

    def withdraw(self):
        amount = float(input("Enter the amount to withdraw: "))
        if (self.balance >= amount):
            self.balance = self.balance - amount
            print ("The withdraw is successful and the balance is %f" % self.balance)
        else:
            print ('Insufficient Balance')

    def enquiry(self):
        print ("Balance in the account is %f" % self.balance)

acc = Bank()
acc.deposit()
acc.withdraw()
acc.enquiry()
```

Output:

The screenshot shows the Visual Studio Code interface with the terminal tab active. The terminal output shows the execution of the 'pract1b.py' script. The user enters '7_GovindSaini' and the account is created. Then, the user enters '10000' for deposit, and the balance is updated to 10000.000000. Next, the user enters '5000' for withdrawal, and the balance is updated to 5000.000000. Finally, the user enters 'q' to quit.

```
7_GovindSaini
The account is created
Enter the amount to be deposit: 10000
The deposit is successful and the balance in the account is 10000.000000
Enter the amount to withdraw: 5000
The withdraw is successful and the balance is 5000.000000
Balance in the account is 5000.000000
q
```

1c) Create Multiple Transaction and display them.

Code :

A screenshot of Visual Studio Code showing a Python file named `pract1cp.py` in the editor. The code implements a simple blockchain system. It defines a `blockchain` list and a `get_last_value` function to return the last item. A `add_value` function adds a transaction to the blockchain with a sender, recipient, and amount. The `get_transaction_value` function prompts the user for sender, recipient, and amount. The `print_block` function prints the entire blockchain. The terminal below shows the command `admin@DESKTOP-3B6118D MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN`.

```
1 #defining the list/chain
2 blockchain = []
3 print("7_GovindSaini \n")
4 #getting the last value/transaction
5 def get_last_value():
6     return(blockchain[-1])
7
8 #adding transaction now we have sender, recipient and amount
9 def add_value(sender, recipient, amount=1.0):
10     transaction = {'sender': sender,
11                    'recipient': recipient,
12                    'amount': amount}
13     blockchain.append(transaction)
14
15 #getting the details of transaction by entering to the prompt
16 def get_transaction_value():
17     tx_sender = input('Enter the sender: ')
18     tx_recipient = input('Enter the recipient of the transaction: ')
19     tx_amount = float(input('Enter your transaction amount: '))
20     return tx_sender, tx_recipient, tx_amount
21
22 #printing the blockchain
23 def print_block():
24     for block in blockchain:
25         print("7_GovindSaini \n")
26         print("Here is your block")
27         print(block)
28
```

A screenshot of Visual Studio Code showing a modified Python file named `pract1cp.py`. The code has been updated to include a loop that keeps adding transactions until the user says no. The terminal below shows the command `admin@DESKTOP-3B6118D MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN`.

```
18     tx_sender = input('Enter the sender: ')
19     tx_recipient = input('Enter the recipient of the transaction: ')
20     tx_amount = float(input('Enter your transaction amount: '))
21     return tx_sender, tx_recipient, tx_amount
22
23 #printing the blockchain
24 def print_block():
25     for block in blockchain:
26         print("Here is your block")
27         print(block)
28
29 #the code will keep repeating for more transaction
30 #until the user answer is no
31 again = True
32 while again == True:
33     tx = get_transaction_value()
34     s, r, a = tx
35     add_value(s, r, a)
36     print(blockchain)
37     more = input("add more block (Y/N)? ")
38     if more.lower() == 'y':
39         again = True
40     else:
41         again = False
```

Output :

The screenshot shows a Visual Studio Code interface with multiple tabs open. The terminal tab displays the execution of a Python script named `pract1cp.py`. The script performs a series of transactions:

```
admin@DESKTOP-3B611B0 MINIGN64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN
$ py pract1cp.py
7_GovindSaini

Enter the sender: Govind
Enter the recipient of the transaction: cg
Enter your transaction amount: 12000
[{'sender': 'Govind', 'recipient': 'cg', 'amount': 12000.0}]
add more block (Y/N)? y
Enter the sender: Govind
Enter the recipient of the transaction: Siddhi
Enter your transaction amount: 15000
[{'sender': 'Govind', 'recipient': 'cg', 'amount': 12000.0}, {'sender': 'Govind', 'recipient': 'Siddhi', 'amount': 15000.0}]
add more block (Y/N)? y
Enter the sender: Govind
Enter the recipient of the transaction: Saroj
Enter your transaction amount: 20000
[{'sender': 'Govind', 'recipient': 'cg', 'amount': 12000.0}, {'sender': 'Govind', 'recipient': 'Siddhi', 'amount': 15000.0}, {'sender': 'Govind', 'recipient': 'Saroj', 'amount': 20000.0}]
add more block (Y/N)? y
Enter the sender: Govind
Enter the recipient of the transaction: Pooja
Enter your transaction amount: 24000
[{'sender': 'Govind', 'recipient': 'cg', 'amount': 12000.0}, {'sender': 'Govind', 'recipient': 'Siddhi', 'amount': 15000.0}, {'sender': 'Govind', 'recipient': 'Saroj', 'amount': 20000.0}, {'sender': 'Govind', 'recipient': 'Pooja', 'amount': 24000.0}]
add more block (Y/N)? n
```

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

Code :

The screenshot shows a Visual Studio Code interface with multiple tabs open. The terminal tab displays the execution of a Python script named `pract1dp.py`. The script defines a `genesis_block` and adds multiple transactions to it:

```
admin@DESKTOP-3B611B0 MINIGN64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN
$ []
pract1dp.py > genesis_block
  1  genesis_block = {
  2      'previous_hash': '',
  3      'index': 0,
  4      'transaction': [],
  5      'nonce': 23
  6  }
  7
  8  blockchain = [genesis_block]
  9
 10 def get_last_value():
 11     return(blockchain[-1])
 12
 13
 14 def add_value(recipient, sender, amount=1.0):
 15     transaction = {'sender': sender,
 16                    'recipient': recipient,
 17                    'amount': amount}
 18     open_transactions.append(transaction)
 19
 20
 21 def get_transaction_value():
 22     tx_sender = raw_input('Enter the sender: ')
 23     tx_recipient = raw_input('Enter the recipient of the transaction: ')
 24     tx_amount = float(input('Enter your transaction amount: '))
 25     return tx_sender, tx_recipient, tx_amount
 26
 27
 28 def print_blockchain():
 29     for i in range(len(blockchain)):
 30         print('Index: ', i)
 31         print('Previous Hash: ', blockchain[i].get('previous_hash'))
 32         print('Index: ', blockchain[i].get('index'))
 33         print('Transactions: ', blockchain[i].get('transaction'))
 34         print('Nonce: ', blockchain[i].get('nonce'))
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under "BLOCKCHAIN" with files like .venv, SolidityProject, and several .py files (pract1d.py, pract1e.py, pract1f.py, pract1g.py, pract1h.py).
- Code Editor:** Displays the content of the pract1d.py file, which contains Python code for a blockchain. The code includes functions for adding a genesis block, getting transaction values, getting user choice, and printing blocks.
- Terminal:** Shows the command \$ [] and the path admin@DESKTOP-3B611B0 MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN.
- Status Bar:** Includes information such as In 1, Col 1, Spaces: 4, UTF-8, CRLF, Python 3.9.7 (.venv: venv), Prettier, and the date/time 18-03-2022 19:51.

Output :

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the same project structure as the previous screenshot.
- Code Editor:** Displays the same content as pract1d.py in the previous screenshot.
- Terminal:** Shows the command \$ py pract1d.py followed by the output: 7_GovindSaini. It then displays the message "Here is your block" and the JSON object: {"previous_hash": "", "index": 0, "transaction": [], "nonce": 23}.
- Status Bar:** Includes information such as In 1, Col 1, Spaces: 4, UTF-8, CRLF, Python 3.9.7 (.venv: venv), Prettier, and the date/time 18-03-2022 19:51.

1e) Create a mining function and test it.

Code :

```
File Edit Selection View Go Run Terminal Help • practie.py - BLOCKCHAIN - Visual Studio Code

EXPLORER ... practie.py pract1d.py pract1f.py practa.py practb.py practc.py practd.py p.py
BLOCKCHAIN
> .venv
p.py
practa.py
practb.py
practc.py
practd.py
practie.py
pract1f.py

1 from hashlib import sha256
2 MAX_NONCE = 10000000000
3
4 print("7_GovindSaini \n")
5 def SHA256(text):
6     return sha256(text.encode("ascii")).hexdigest()
7
8 def mine(block_number, transactions, previous_hash, prefix_zeros):
9     prefix_str = '0'*prefix_zeros
10    for nonce in range(MAX_NONCE):
11        text = str(block_number) + transactions + previous_hash + str(nonce)
12        new_hash = SHA256(text)
13        if new_hash.startswith(prefix_str):
14            print(f"Yay! Successfully mined bitcoins with nonce value:{nonce}")
15            return new_hash
16
17    raise BaseException(f"Couldn't find correct has after trying {MAX_NONCE} times")
18
19 if __name__ == '__main__':
20     transactions = ''
21     Govind->Siddhi->20,
22     Saroj->Pooja->45
23     ...
24     # try changing this to higher number and you will see it will take more time for mining as difficulty increases
25     import time
26     difficulty = 4
27
28
29
30
31
32
33
34
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN

\$

Ln 18, Col 1 Spaces: 4 UTF-8 CRLF Python 3.9.7 (.venv\venv) ⚡ Prettier ⚡ 29°C Smoke ⚡ 19:56 19-03-2022

```
File Edit Selection View Go Run Terminal Help • practie.py - BLOCKCHAIN - Visual Studio Code

EXPLORER ... practie.py pract1d.py pract1f.py practa.py practb.py practc.py practd.py p.py
BLOCKCHAIN
> .venv
p.py
practa.py
practb.py
practc.py
practd.py
practie.py
pract1f.py

14     print(f"Yay! Successfully mined bitcoins with nonce value:{nonce}")
15     return new_hash
16
17 raise BaseException(f"Couldn't find correct has after trying {MAX_NONCE} times")
18
19 if __name__ == '__main__':
20     transactions = ''
21     Govind->Siddhi->20,
22     Saroj->Pooja->45
23     ...
24     # try changing this to higher number and you will see it will take more time for mining as difficulty increases
25     import time
26     difficulty = 4
27     start = time.time()
28     print("start mining")
29     new_hash = mine(
30         5, transactions, '000000xa036944e29568d0cff17edbe038f81208fecf9a66be9a2b8321c6ec7', difficulty)
31     total_time = str((time.time() - start))
32     print(f"end mining. Mining took: {total_time} seconds")
33     print("the new hash value : ",new_hash)
34
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN

\$

Ln 18, Col 1 Spaces: 4 UTF-8 CRLF Python 3.9.7 (.venv\venv) ⚡ Prettier ⚡ 29°C Smoke ⚡ 19:56 19-03-2022

Output :

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "BLOCKCHAIN" containing files: .venv, p.py, pract1a.py, pract1b.py, pract1c.py, pract1d.py, pract1e.py, and pract1f.py.
- Code Editor:** The active file is "pract1e.py". The code implements a blockchain mining process, including a mining loop, nonce generation, and difficulty adjustment.
- Terminal:** The terminal window shows the execution of the script and its output, indicating successful mining of bitcoins.
- Status Bar:** Displays the current line (Ln 18, Col 1), spaces (Spaces: 4), encoding (UTF-8), Python version (3.9.7), and system information (29°C, 19:57, 19-03-2022).

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

Code :

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "BLOCKCHAIN" containing files: .venv, SolidityProject, ~\$govindsaini(Blockchain_Pract1).d..., 7_govindsaini(Blockchain_Pract1).d..., Doc1.docx, Doc1.pdf, p.py, pract1a.py, pract1b.py, pract1c.py, pract1d.py, pract1e.py, and pract1f.py.
- Code Editor:** The active file is "pract1f.py". It contains a class definition for "Block" which includes methods for initialization, hashing, and string representation.
- Terminal:** The terminal window shows the execution of the script and its output, indicating successful mining of bitcoins.
- Status Bar:** Displays the current line (Ln 66, Col 43), spaces (Spaces: 4), encoding (UTF-8), Python version (3.9.7), and system information (35°C, 19:52, 18-03-2022).

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project folder named "BLOCKCHAIN" containing files like .venv, SolidityProject, and several Python files (pract1a.py, pract1b.py, pract1c.py, pract1d.py, pract1f.py).
- Code Editor:** The main editor window displays the "blockchain.py" file. The code defines a "Blockchain" class with methods for adding blocks, mining blocks, and printing the chain.
- Terminal:** The terminal at the bottom shows the command "py pract1f.py" being run.
- Status Bar:** The status bar indicates the file is 66 lines long, 43 columns wide, and uses UTF-8 encoding. It also shows the date and time as 18-03-2022 19:53.

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the same project structure as the first screenshot.
- Code Editor:** The main editor window displays the "blockchain.py" file with additional code added. It includes a loop that creates 10 blocks and prints them.
- Terminal:** The terminal at the bottom shows the command "py pract1f.py" being run.
- Status Bar:** The status bar indicates the file is 66 lines long, 43 columns wide, and uses UTF-8 encoding. It also shows the date and time as 18-03-2022 19:53.

Output :

pract1f.py - BLOCKCHAIN - Visual Studio Code

EXPLORER

- BLOCKCHAIN
 - .venv
 - SolidityProject
 - ~\$govindsaini(Blockchain_Pract1).d...
 - 7_govindsaini(Blockchain_Pract1).d...
 - 7_govindsaini(Blockchain_Pract2).d...
 - Doc1.docx
 - Doc1.pdf
 - p.py
 - pract1a.py
 - pract1b.py
 - pract1c.py
 - pract1d.py
 - pract1e.py
 - pract1f.py

PROBLEMS **OUTPUT** **TERMINAL** **DEBUG CONSOLE**

```

pract1fpy > Block
1 import datetime
2 import hashlib
3
4 print("7_GovindSaini \n")

admin@DESKTOP-38611B0 MINGW64 /c/govIndworking/SEM 4 MSC IT/BLOCKCHAIN
$ py pract1f.py
7_GovindSaini

Block Hash: 2da69f580363b47569cff724f7ffe96993dc92d96f3086ccc2b25482fb583
BlockNo: 1
Block Data: Block 1
Hashes: 11641
-----
Block Hash: 2cd5120ab93b2c7136384fb57b08b1afe571940e6c1561f01d5991aabfe1c4
BlockNo: 2
Block Data: Block 2
Hashes: 1732856
-----
Block Hash: c815958bc3663c736ab5266862e600207a5964d73eafaddb1ac2b2592e28810b
BlockNo: 3
Block Data: Block 3
Hashes: 98750
-----
Block Hash: 85fd87fa35bc3ff8f586c4a1da6540d8c61838552b57a8b36b486bb2885e772
BlockNo: 4
Block Data: Block 4
Hashes: 407313
-----
Block Hash: 1df3b068aa2036bd6bcc89daf857195e6b4e609ea1312de6b869f29db5d9dd3
BlockNo: 5
Block Data: Block 5
Hashes: 2562101

```

Ln 9, Col 16 **Spaces: 4** **UTF-8** **CRLF** **Python 3.9.7 (.venv/.venv)** **Prettier**

35°C Smoke 18-03-2022

pract1f.py - BLOCKCHAIN - Visual Studio Code

EXPLORER

- BLOCKCHAIN
 - .venv
 - SolidityProject
 - ~\$govindsaini(Blockchain_Pract1).d...
 - 7_govindsaini(Blockchain_Pract1).d...
 - 7_govindsaini(Blockchain_Pract2).d...
 - Doc1.docx
 - Doc1.pdf
 - p.py
 - pract1a.py
 - pract1b.py
 - pract1c.py
 - pract1d.py
 - pract1e.py
 - pract1f.py

PROBLEMS **OUTPUT** **TERMINAL** **DEBUG CONSOLE**

```

pract1fpy > Block
1 import datetime
2 import hashlib
3
4 print("7_GovindSaini \n")

Block Hash: 42e3ba3d954744af6ddd34fa476d8e213a661ee81fe8b96fdb4ecace1dc0d0
BlockNo: 6
Block Data: Block 6
Hashes: 288879
-----
Block Hash: d2ad82f546889bb1bfcf13d906b3acfad47af84182e8577905239e9a537ced2a
BlockNo: 7
Block Data: Block 7
Hashes: 5204675
-----
Block Hash: f01fd4a85de5a39646b6d16e5623c1946101cc9f7940eef56d79966e986eb12c
BlockNo: 8
Block Data: Block 8
Hashes: 299732
-----
Block Hash: 2e50b68a0bb3022e4b35b1c96b3dfbefdd9c47270d5bec6b390f72fe345a19b
BlockNo: 9
Block Data: Block 9
Hashes: 1378275
-----
Block Hash: c6c4a0f2c5e5b683c66c11eec2b585588597ce5e86f999cd67e2dc79cd48529c
BlockNo: 10
Block Data: Block 10
Hashes: 2953554
-----

admin@DESKTOP-38611B0 MINGW64 /c/govIndworking/SEM 4 MSC IT/BLOCKCHAIN
$ 
```

Ln 9, Col 16 **Spaces: 4** **UTF-8** **CRLF** **Python 3.9.7 (.venv/.venv)** **Prettier**

35°C Smoke 18-03-2022

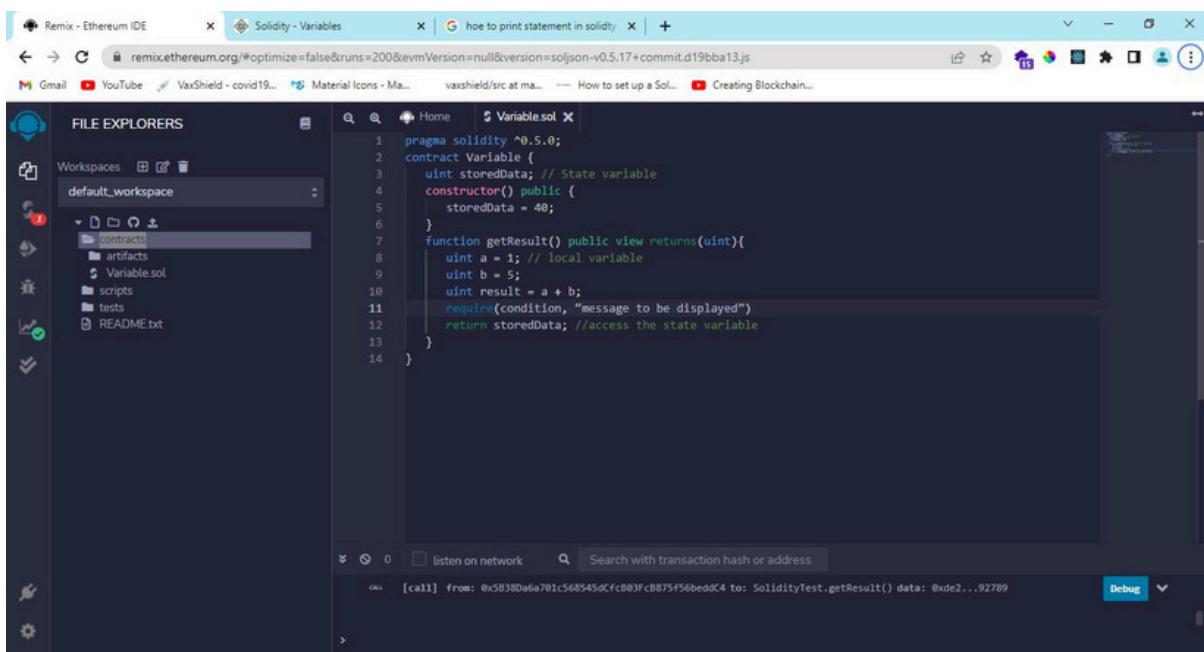
Practical No : 3

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

3a) Variable, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs, Mappings, Conversions, Ether Units, Special Variables.

Variable

Code :



```
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;
        require(condition, "message to be displayed")
        return storedData; //access the state variable
    }
}
```

Output :

```

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

```

The screenshot shows the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is open, showing a deployed contract named 'VARIABLE AT 0XD91...39138 (MEMO)'. Below it, there's a 'getResult' button and a 'Low level interactions' section with a 'Transact' button. The main central area displays the Solidity code for 'Variable.sol'. At the bottom, the status bar shows the date and time as 18-03-2022 13:10.

Operators :

Arthmetic Operator

Code:

```

1 pragma solidity ^0.5.0;
2
3 contract ArithmeticOperators {
4     uint16 public a = 50;
5     uint16 public b = 20;
6
7     uint public sum = a + b;
8     uint public diff = a - b;
9     uint public mul = a * b;
10
11    uint public div = a / b;
12    uint public mod = a % b;
13
14    uint public dec = --b;
15    uint public inc = ++a;
16
17}

```

The screenshot shows the Remix Ethereum IDE interface. On the left, the 'FILE EXPLORERS' sidebar shows a workspace named 'default_workspace' containing several Solidity files like 'Variable.sol', 'Arrays.sol', 'Enums.sol', etc., and the current file 'ArithmeticOperators.sol'. The main central area displays the Solidity code for 'ArithmeticOperators.sol'. At the bottom, the status bar shows the date and time as 18-03-2022 20:58.

Output:

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar titled "DEPLOY & RUN TRANSACTIONS" with various operation buttons like add, sub, mul, div, etc. The main central area displays the Solidity code for a contract named "ArithmeticOperators". The code defines two public uint16 variables, a and b, and three public functions: sum, diff, and mul. The "ContractDefinition" section shows the constructor and the three defined functions. Below the code, there are logs from the VM and a "Debug" button.

```
pragma solidity ^0.5.0;
contract ArithmeticOperators {
    uint16 public a = 50;
    uint16 public b = 20;

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

This screenshot shows the same Remix interface but with the "DEBUGGER" tab selected. The left sidebar now includes "Function Stack", "Solidity Locals", and "Solidity State". The "Solidity State" section lists local variables and their values: a: 51, b: 19, sum: 70, diff: 30, mul: 1000, div: 2, mod: 10, dec: 19, inc: 51. The bottom part of the screen shows the assembly code generated for the contract, starting with 348 JUMPDEST.

```
348 JUMPDEST
349 PUSH2 0164
352 PUSH2 0115
355 JUMP
356 JUMPDEST
357 PUSH1 40
```

Relational Operator

Code:

The screenshot shows the Remix Ethereum IDE interface. The top bar has tabs for "Remix - Ethereum IDE" and "Solidity - Operators - GeeksforGeeks". The main area displays the Solidity code for a contract named `RelationalOperator`. The code defines two uint16 variables `a` and `b`, both initialized to 10. It then defines six boolean functions: `eq`, `noteq`, `gtr`, `les`, `gtreq`, and `leseq`, each comparing `a` and `b`. The code is annotated with comments explaining the creation of the contract and the definition of relational operators.

```
pragma solidity ^0.5.0;

// Creating a contract
contract RelationalOperator {

    uint16 public a = 10;
    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;
}
```

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 each function: `a`, `b`, `eq`, `gtr`, `gtreq`, `les`, `lesseq`, and `noteq`. Each button has a value of 0 and a type of bool. The main pane shows the same Solidity code as the previous screenshot. At the bottom, transaction logs are displayed. For the `les` function, a log entry shows a call from address 0x5B380a6a701c568545dCfcB03FcB875f56beddC4 to the contract, with data 0xb61...2aa97. For the `noteq` function, another log entry shows a call from the same address to the contract, with data 0x6a8...a6ab5. There are also "Debug" buttons next to each log entry.

The screenshot shows the Remix Ethereum IDE interface. The top navigation bar has tabs for "Remix - Ethereum IDE" and "Solidity - Operators - GeeksforGeeks". The main area displays a Solidity code editor with two files: "ArithmeticOperators.sol" and "RelationalOperator.sol". The "RelationalOperator.sol" file contains the following code:

```
pragma solidity ^0.5.0;

// Creating a contract
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;
}
```

The "DEBUGGER" panel on the left shows the state of variables: a: 70 uint16, b: 10 uint16, eq: false bool, noteq: true bool, gtr: true bool, les: false bool, gtneg: true bool, lessq: false bool. A message indicates that the call has reverted, and the transaction hash is 0xa0d...7afb7.

Logical Operator

Code :

The screenshot shows the Remix Ethereum IDE interface. The top navigation bar has tabs for "Remix - Ethereum IDE" and "Solidity - Operators - GeeksforGeeks". The main area displays a Solidity code editor with three files: "ArithmeticOperators.sol", "RelationalOperator.sol", and "LogicalOperator.sol". The "LogicalOperator.sol" file contains the following code:

```
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);
    }
}
```

The "FILE EXPLORERS" panel on the left shows the project structure with workspaces and various Solidity files like Variable.sol, Arrays.sol, Enums.sol, Struct.sol, MappingVariable.sol, whileLoop.sol, doWhileLoop.sol, functionModifier.sol, ViewFunction.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, Function.sol, string.sol, ArithmeticOperators.sol, RelationalOperator.sol, and LogicalOperator.sol.

Output:

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar titled "DEPLOY & RUN TRANSACTIONS" with options like "Deploy", "Publish to IPFS", and "At Address". Below it, "Transactions recorded" and "Deployed Contracts" sections are shown, with "LOGICALOPERATOR AT 0x358...DSEE" expanded. In the main area, the Solidity code for the `LogicalOperator` contract is displayed:

```
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;
    }
}
```

On the right, the "Logs" section shows two entries:

- [vm] from: 0x583...eddC4 to: logicalOperator.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x7d5...7ae9a Debug
- [call] from: 0x583B0a6a701c568545cfcB03Fc8875f56bedd4 to: logicalOperator.Logic(bool,bool) data: 0xc94...00000 Debug

The status bar at the bottom indicates "32°C Smoke" and the date "18-03-2022".

Bitwise Operator

Code:

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a "FILE EXPLORERS" sidebar with a "Workspaces" section showing "default_workspace" containing various Solidity files like `Variable.sol`, `Struct.sol`, etc., and a "Contracts" section where `BitwiseOperator.sol` is selected. In the main area, the Solidity code for the `BitwiseOperator` contract is displayed:

```
pragma solidity ^0.5.0;
// Creating a contract
contract BitwiseOperator {
    // Declaring variables
    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 ;
}
```

The status bar at the bottom indicates "32°C Smoke" and the date "18-03-2022".

Output :

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar titled "DEPLOY & RUN TRANSACTIONS" containing a list of operators: and, or, xor, leftshift, rightshift, not, and uint16 values (20, 16, 50, 54, 38). The main code editor displays the following Solidity code:

```
pragma solidity ^0.5.0;
// Creating a contract
contract BitwiseOperator {
    // Declaring variables
    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;
}
```

The "ContractDefinition BitwiseOperator" section shows 1 reference(s). The "Logs" tab displays two entries:

- [vm] from: 0x5B3...eddC4 to: logicalOperator.(constructor) value: 0 wei data: 0x600...10032 logs: 0 hash: 0x428...4889c
- [call] from: 0x5B380a6a701c568545dCfcB03FcB875f56beddC4 to: logicalOperator.Logic(bool,bool) data: 0xc94...00000

The status bar at the bottom indicates "32°C Smoke" and the date "18-03-2022".

This screenshot shows the same Remix IDE interface but with the "DEBUGGER" tab selected. The left sidebar now includes a "Function Stack" and "Solidity Locals" section. The "Solidity State" section shows variable values: a: 20 uint16, b: 50 uint16, and: 16 uint16, or: 54 uint16, xor: 38 uint16, leftshift: 0 uint16, rightshift: 0 uint16, not: 65515 uint16. The assembly code pane at the bottom shows:

```
0463 INVALID
0464 PUSH1 60
0465 PUSH1 40
0466 MSTORE
0469 CALLVALUE
0470 DUP1
0471 ISZERO
```

The logs section shows the same deployment events as the previous screenshot.

Assignment Operator

Code :

The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file tree under 'default_workspace' containing various Solidity files like Variable.sol, Arrays.sol, Enums.sol, Struct.sol, Mapping.sol, SpecialVariable.sol, whileLoop.sol, doWhileLoop.sol, functionModifier.sol, ViewFunction.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, Function.sol, string.sol, and ArithmeticOperators.sol. The main editor window shows the Solidity code for 'ArithmeticOperators.sol'. The code defines a contract with public variables a and b, and functions sum, diff, mul, div, mod, dec, and inc.

```
pragma solidity ^0.5.0;

contract ArithmeticOperators {
    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;
}
```

Ouput :

The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file tree under 'ASSIGNMENTOPERATOR AT 0xD91...' containing functions getResult, assign_div, assign_mod, assign_mul, assign_sub, assignment, and assignment_a_. The main editor window shows the Solidity code for 'AssignmentOperator.sol'. The code creates a contract with variables assignment, assignment_add, assignment_sub, assignment_mul, assignment_div, and assignment_mod. It includes a function getResult that performs arithmetic operations on these variables.

```
pragma solidity ^0.5.0;

// Creating a contract
contract AssignmentOperator {

    // Declaring variables
    uint16 public assignment = 20;
    uint public assignment_add = 50;
    uint public assign_sub = 50;
    uint public assign_mul = 10;
    uint public assign_div = 50;
    uint public assign_mod = 32;

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

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar with tabs for DEBUGGER, Function Stack, Solidity Locals, and Solidity State. The Solidity State tab displays variables: assignment: 20 uint16, assignment_add: 60 uint256, assign_sub: 30 uint256, assign_mul: 100 uint256, assign_div: 5 uint256, and assign_mod: 12 uint256. The main editor area contains the AssignmentOperator.sol code. Below the code, a ContractDefinition panel shows 'AssignmentOperator' with 1 reference(s). The bottom status bar indicates the transaction hash: 0xa80...297c8.

```
pragma solidity ^0.5.0;

// Creating a contract
contract AssignmentOperator {

    // Declaring variables
    uint16 public assignment = 20;
    uint public assignment_add = 60;
    uint public assign_sub = 30;
    uint public assign_mul = 100;
    uint public assign_div = 5;
    uint public assign_mod = 12;

    function getResult() public{
        assignment_add += 10;
        assign_sub -= 20;
    }
}
```

Loops

whileLoop

code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar has a FILE EXPLORERS tab with a workspace named 'default_workspace'. Inside the workspace, there are several contracts: artifacts, Variable.sol, Arrays.sol, Enums.sol, Struct.sol, Mapping.sol, SpecialVariable.sol, whileLoop.sol (which is selected), doWhileLoop.sol, functionModifier.sol, ViewFunction.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, Function.sol, string.sol, ArithmeticOperators.sol, RelationalOperator.sol, LogicalOperator.sol, BitwiseOperator.sol, and AssignmentOperator.sol. The main editor area contains the whileLoop.sol code. The bottom status bar indicates the transaction hash: 0xa80...297c8.

```
pragma solidity ^0.5.0;

// Creating a contract
contract whileLoop {

    // Declaring a dynamic array
    uint[] data;

    // Declaring state variable
    uint8 j = 0;

    function loop()
        public
        returns(uint[] memory)
    {
        while(j < 16) {
            j++;
            data.push(j);
        }
        return data;
    }
}
```

Output

```
pragma solidity ^0.5.0;
// Creating a contract
contract whiteLoop {
    // Declaring a dynamic array
    uint[] data;
    // Declaring state variable
    uint8 j = 0;
    function loop()
        public returns(uint[] memory){
        while(j < 16) {
            j++;
        }
    }
}
```

```
pragma solidity ^0.5.0;
// Creating a contract
contract whiteLoop {
    // Declaring a dynamic array
    uint[] data;
    // Declaring state variable
    uint8 j = 5;
    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 displays a file explorer with a workspace named 'default_workspace' containing various Solidity files. The main editor window shows the following Solidity code:

```
pragma solidity ^0.5.0;

// Creating a contract
contract doWhileLoop{

    // Declaring a dynamic array
    uint[] data;

    // Declaring state variable
    uint8 j = 0;

    // Defining function to demonstrate
    // 'Do-While loop'
    function loop()
    public returns(uint[] memory){
        do{
            j++;
            data.push(j);
        }while(j < 8);
        return data;
    }
}
```

Output :

The screenshot shows the Remix Ethereum IDE with the debugger open. The left sidebar shows the 'DEBUGGER' tab with sections for 'Function Stack' (containing 'do.loop()'), 'Solidity Locals', and 'Solidity State'. The 'Solidity State' section shows the variable 'data' with a length of 8, containing values from 0 to 7. The main editor window shows the same Solidity code as before. The bottom status bar indicates the transaction is pending.

Forloop

Code :

```
pragma solidity ^0.5.0;
// Creating a contract
contract forLoop {
    // Declaring a dynamic array
    uint[] data;
    // Defining a function
    // to demonstrate 'For loop'
    function loop() public returns(uint[] memory){
        for(uint i=0; i<4; i++){
            data.push(i);
        }
        return data;
    }
}
```

Output :

```
pragma solidity ^0.5.0;
// Creating a contract
contract forLoop {
    // Declaring a dynamic array
    uint[] data;
    // Defining a function
    // to demonstrate 'For loop'
    function loop() public returns(uint[] memory){
        for(uint i=0; i<4; i++){
            data.push(i);
        }
        return data;
    }
}
```

Decision making

If statement

Code:

```
pragma solidity ^0.5.0;
// Creating a contract
contract ifStatement {
    // Declaring state variable
    uint i = 20;

    function decision_making()
        public
        returns(bool)
    {
        if(i<20){
            return true;
        }
    }
}
```

Code :

```
pragma solidity ^0.5.0;
// Creating a contract
contract ifStatement {
    // Declaring state variable
    uint i = 20;

    function decision_making()
        public
        returns(bool)
    {
        if(i<20){
            return true;
        }
    }
}
```

If...else statement

Code :

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORERS panel lists various Solidity files. In the center, the code editor displays the following Solidity code:

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

    // Defining function to
    // demonstrate the use of
    // 'if...else statement'
    function decision_making()
        public payable returns(bool){
        if(i%2 == 0){
            even = true;
        }
        else{
            even = false;
        }
        return even;
    }
}
```

The code defines a contract named `IfElseStatement` with a state variable `i` set to 20 and a boolean variable `even`. It contains a function `decision_making` that returns the value of `even` based on whether `i` is even or odd.

Output :

The screenshot shows the Remix Ethereum IDE with the DEBUGGER tab selected. The code editor on the right shows the same Solidity code as before. The left sidebar includes a DEBUGGER CONFIGURATION section with a checkbox for "Use generated sources (from Solidity v0.7.2)" and a "Stop debugging" button. The Function Stack panel shows a single entry: `b: decision_making()`. The Solidity Locals panel shows the variable `i: 20 uint256` and `even: false bool`. The Solidity State panel shows the initial state of the contract. At the bottom, the EVM Stack panel displays assembly code: `0x561fca1e0f53c36928167d66386f...`, followed by `009 DUP1`, `009 PUSH1 02`, `073 PUSH1 00`, and `073 SLOAD`. The status bar at the bottom indicates it's 21:56 on 18-03-2022.

If...else if...else statement

Code :

The screenshot shows the Remix Ethereum IDE interface. The left sidebar lists various Solidity files. The main code editor displays the following Solidity code:

```
pragma solidity ^0.5.0;
// Creating a contract
contract ifElseStatement {
    // Declaring state variables
    uint i = 12;
    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 open. The left sidebar shows the debugger configuration and stack trace. The main code editor shows the same Solidity code as before. The bottom pane displays the debugger logs:

```
creation of ifElseStatement pending...
[vm] from: 0x5B3...edDC4 to: ifElseStatement.(constructor) value: 0 wei data: 0x608...10032 logs: 0
hash: 0x418...b024
transact to ifElseStatement.decision_making pending ...
[vm] from: 0x5B3...edDC4 to: ifElseStatement.decision_making() 0x93F...C96CC value: 0 wei data: 0x887...3eF24 logs: 0
hash: 0xad6...a28fd
```

String

Code:

```

3 contract string {
4     string[] public row;
5
6     function getRow() public view returns (string[] memory) {
7         return row;
8     }
9
10    function pushToRow(string memory newValue) public {
11        row.push(newValue);
12    }
13 }

```

The screenshot shows the Remix Ethereum IDE interface. On the left, the file explorer displays a workspace named 'default_workspace' containing several Solidity files such as 'Variable.sol', 'Arrays.sol', 'Enums.sol', 'Struct.sol', 'Mapping.sol', 'SpecialVariable.sol', 'whileLoop.sol', 'doWhileLoop.sol', 'Operators.sol', 'functionModifier.sol', 'ViewFunction.sol', 'PureFunction.sol', 'FallbackFunction.sol', 'FunctionOverloading.sol', 'MathematicalFunction.sol', 'CryptographicFunction.sol', 'Function.sol', and 'string.sol'. The 'string.sol' file is currently selected. The main editor area shows the provided Solidity code. At the bottom, the terminal window shows a transaction call to the 'getRow' function.

Output :

The screenshot shows the Remix Ethereum IDE interface with the 'DEPLOY & RUN TRANSACTIONS' sidebar open. The sidebar shows the contract 'String - contracts/string.sol' selected, with a 'Deploy' button highlighted. Below it, there are options to 'Publish to IPFS' or 'At Address' (the latter being selected). The main editor area shows the same Solidity code as the previous screenshot. The terminal window at the bottom shows the deployment of the contract and subsequent calls to its functions: 'creation of String pending...', 'transact to String.pushToRow pending ...', and 'call to String.getRow'.

Array

Code :

The screenshot shows the Ethereum IDE interface. The left sidebar displays a file tree for a workspace named 'default_workspace'. The 'Contracts' folder contains 'Variable.sol' and 'Arrays.sol'. The 'scripts' folder contains 'remix-tests.sol' and 'remix_accounts.sol'. A 'README.txt' file is also present. The main editor window shows the Solidity code for 'Arrays.sol'.

```
1 pragma solidity ^0.5.0;
2
3 contract test {
4     function testArray() public pure{
5         uint len = 7;
6         //dynamic array
7         uint[] memory a = new uint[](7);
8
9         //bytes is same as byte[]
10        bytes memory b = new bytes(len);
11
12        assert(a.length == 7);
13        assert(b.length == len);
14
15        //access array variable
16        a[6] = 8;
17
18        //test array variable
19        assert(a[6] == 8);
20        //static array
21        uint[3] memory c = [uint(1), 2, 3];
22        assert(c.length == 3);
23    }
24 }
```

Output:

The screenshot shows the Ethereum IDE interface with the 'DEPLOY & RUN TRANSACTIONS' tab selected. The 'CONTRACT' dropdown shows 'test - contracts/Arrays.sol'. Below it, there are buttons for 'Deploy', 'Publish to IPFS', and 'At Address'. The 'Transactions recorded' section shows a transaction for 'TEST AT 0XD91...39138(MEMORY)'. The 'testArray' function is highlighted. The right side of the interface shows the Solidity code for 'Arrays.sol' and a list of recent transactions.

```
3 contract test {
4     function testArray() public pure{
5         uint len = 7;
6         //dynamic array
7         uint[] memory a = new uint[](7);
8
9         //bytes is same as byte[]
10        bytes memory b = new bytes(len);
11
12        assert(a.length == 7);
13        assert(b.length == len);
14
15        //access array variable
16        a[6] = 8;
17
18        //test array variable
19        assert(a[6] == 8);
20        //static array
21        uint[3] memory c = [uint(1), 2, 3];
22        assert(c.length == 3);
23    }
24 }
```

Enums

Code :

```
pragma solidity ^0.5.0;

contract Enums {
    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);
    }
}
```

Output :

```
pragma solidity ^0.5.0;

contract Enums {
    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) {

```

Struct

Code:

The screenshot shows the Ethereum IDE interface. The left sidebar displays a file explorer with a workspace named 'default_workspace' containing files like Contracts, Artifacts, Arrays.sol, Enums.sol, Structs.sol, Scripts, Tests, Deps, Remix-Tests, and Remix-Accounts.sol. The right panel shows the Solidity code for a 'Struct' contract:

```
pragma solidity ^0.5.0;

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

    function setBook() public {
        book = Book('Learn JavaScript', 'TP', 4);
    }
    function getBookId() public view returns (uint) {
        return book.book_id;
    }
}
```

Output:

The screenshot shows the Ethereum IDE interface after deploying the 'Struct' contract. The left sidebar now includes a 'DEPLOY & RUN TRANSACTIONS' section. It shows two deployed contracts: 'TEST AT 0xD91_39138 (MEMORY)' and 'ENUMS AT 0XD8B_33FAB (MEMORY)'. The right panel shows the same Solidity code as before. Below the code, the transaction history is displayed:

- [vm] From: 0x583...eddC4 To: Struct.(constructor) Value: 0 Wei Data: 0x600...10032 Logs: 0 Hash: 0x329...ea9d6
- [vm] From: 0x583...eddC4 To: Struct.setBook() 0xf8e...9f8e8 Value: 0 Wei Data: 0xd62...9546c Logs: 0 Hash: 0xe1...a2936
- [call] From: 0x583B0a6a701c568545cFcB03Fc8875f56beddC4 To: Struct.getBookId() Data: 0x731...0d33c

Mapping

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file explorer for the workspace named "default_workspace". The workspace contains several files: contracts (Variable.sol, Arrays.sol, Enums.sol, Struct.sol, Mapping.sol.sol), artifacts, scripts (deploy_web3.js, deploy_etherjs), tests, deps, remix-tests, and README.txt. The central editor area shows the Solidity code for a "Mapping" contract. The code defines a struct "student" with fields name, subject, and marks. It then creates a mapping from address to student. A function "adding_values()" adds values to the mapping. The code is annotated with comments explaining the structure definition and mapping creation.

```
3 contract Mapping {
4     //Defining structure
5     struct student {
6         //Declaring different
7         // structure elements
8         string name;
9         string subject;
10        uint8 marks;
11    }
12
13    // Creating mapping
14    mapping (
15        address => student) result;
16    address[] public student_result;
17
18    // Function adding values to the mapping
19    function adding_values() public {
20        var student
21        = result[0xDE7796E89C82C36BAdd1375076f39D69FafE252];
22
23        student.name = "John";
24        student.subject = "Chemistry";
25        student.marks = 88;
26        student_result.push(
27            0xE7796E89C82C36BAdd1375076f39D69FafE252) -1;
28    }
29 }
```

Output :

The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" tab selected. The left sidebar shows a deployment history with one entry: "MAPPING_EXAMPLE AT 0x540...c751". The main editor area shows the same Solidity code for the "Mapping" contract. Below the code, the "Transactions recorded" section indicates that two transactions have been recorded. The first transaction is labeled "adding_values" and shows a pending call to "mapping_example.adding_values()". The second transaction is also labeled "adding_values" and shows a pending call to "mapping_example.adding_values()". The status bar at the bottom right shows the date and time as 18-03-2022 18:23.

```

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

    mapping (address => student) result;
}

function addingValues() public {
}

```

Special Variable

Code:

```

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

```
pragma solidity ^0.6.6;

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

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

    // Defining a function to
    // return the roll no.
}
```

ContractDefinition SpecialVariable 1 reference(s) ▾

Low level interactions

CALL [call] from: 0x58380a6a701c568545d0fc803fc8875f56beddC4 to: SpecialVariable.whatIsMyRollNumber() data: 0x3ee...b5710 Debug

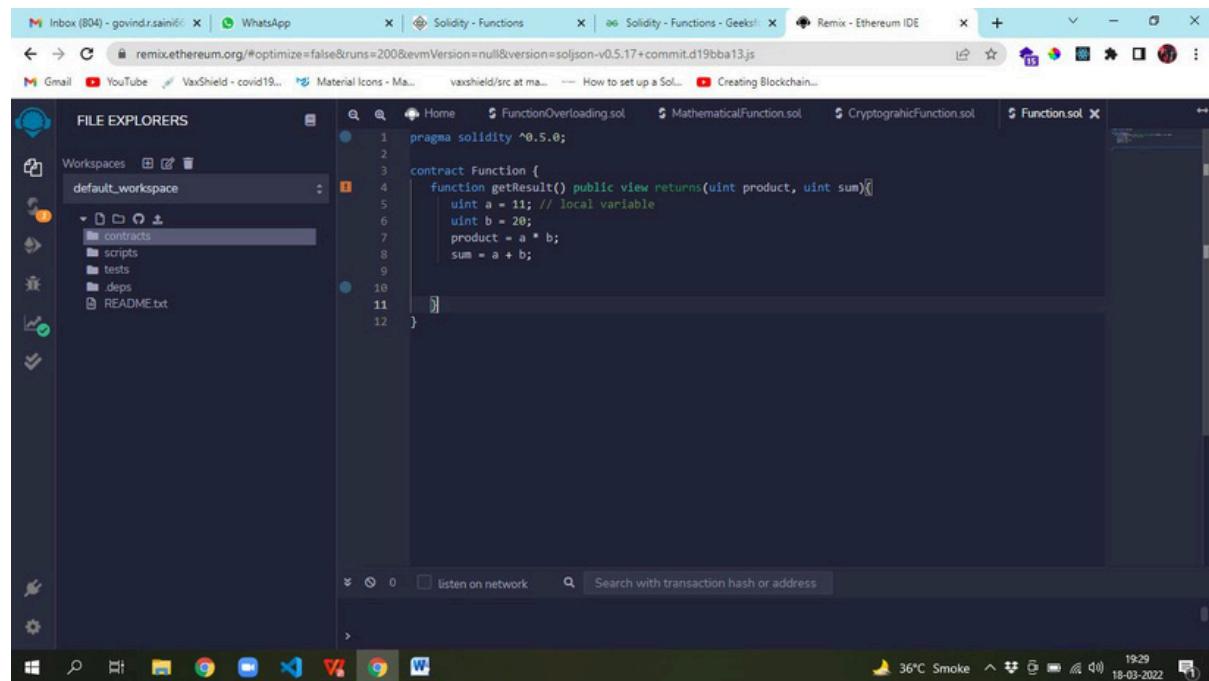
CALL [call] from: 0x58380a6a701c568545d0fc803fc8875f56beddC4 to: SpecialVariable.setRollNO(uint256) 0xC38...aaECA value: 0 wei data: 0x15e...00007 logs: 0 Debug

CALL [call] from: 0x58380a6a701c568545d0fc803fc8875f56beddC4 to: SpecialVariable.whatIsMyRollNumber() data: 0x3ee...b5710 Debug

3b) Functions, Function Modifiers, View functions, Pure Functions, Fallback Function, Function Overloading, Mathematical functions, Cryptographic functions.

Function

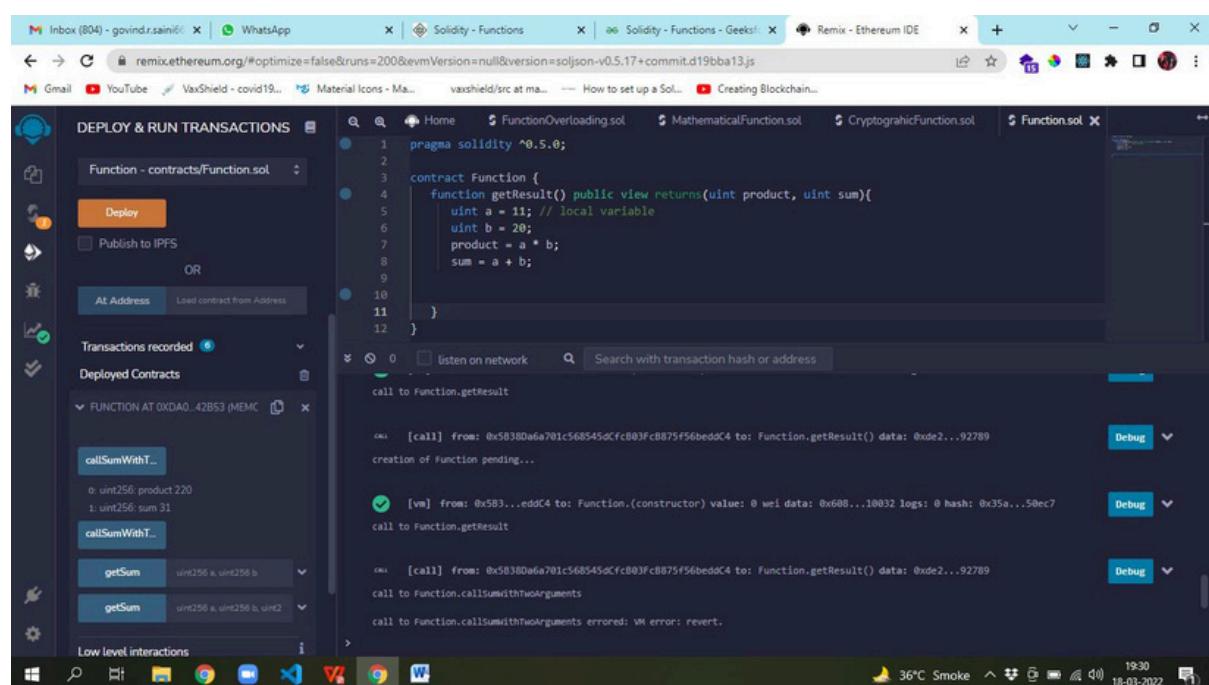
Code:



```
pragma solidity ^0.5.0;

contract Function {
    function getResult() public view returns(uint product, uint sum){
        uint a = 11; // local variable
        uint b = 20;
        product = a * b;
        sum = a + b;
    }
}
```

Output:

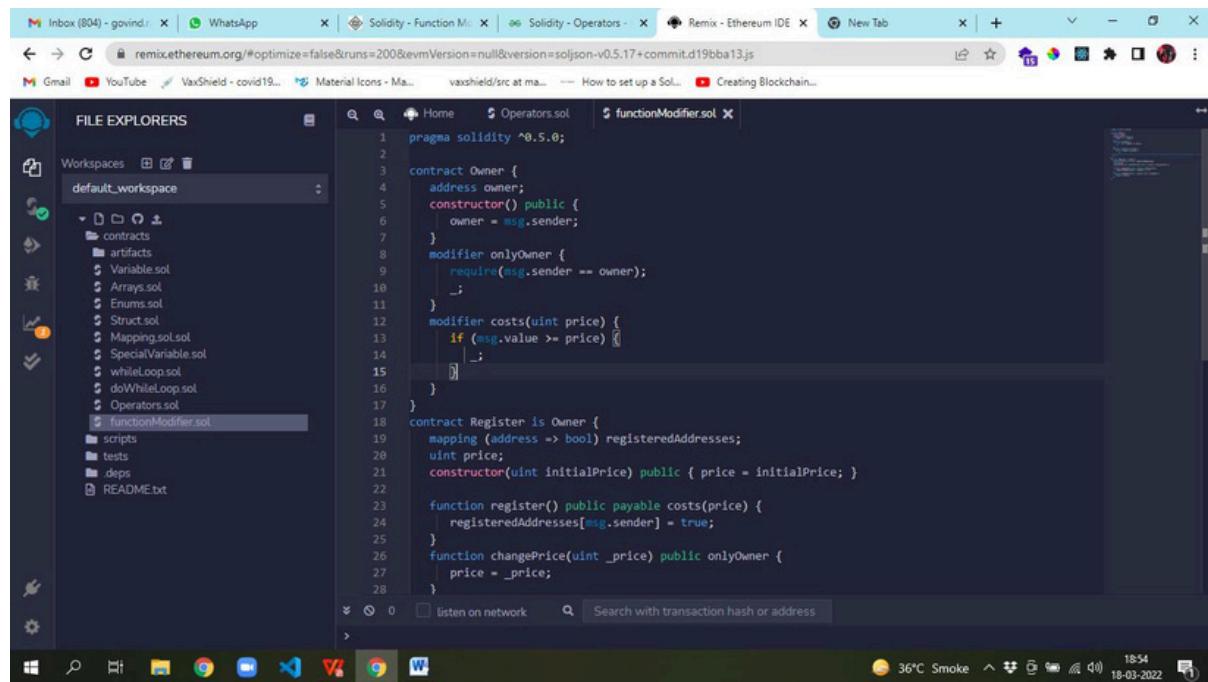


The screenshot shows the Remix Ethereum IDE interface. On the left, the "FILE EXPLORERS" panel displays a workspace named "default_workspace" containing contracts, scripts, tests, and dependencies. The main code editor window shows the same Solidity code as above. On the right, the "DEPLOY & RUN TRANSACTIONS" panel is active, showing the deployed contract "FUNCTION AT 0XDAO...42B53 (MEMC)". It lists several transaction logs and interactions:

- A call to `Function.getResult()` from address 0x58380a6a701c568545dCfcB803FcB875f56beddC4 to Function getResult() data: 0xde2...92789 creation of Function pending...
- A constructor call from address 0x583...eddc4 to Function.(constructor) value: 0 wei data: 0x600...10032 logs: 0 hash: 0x35a...50ec7
- A call to `Function.getResult()` from address 0x58380a6a701c568545dCfcB803FcB875f56beddC4 to Function getResult() data: 0xde2...92789
- A call to `Function.callSumWithTwoArguments` from address 0x58380a6a701c568545dCfcB803FcB875f56beddC4 to Function.callSumWithTwoArguments data: 0xde2...92789
- A call to `Function.callSumWithTwoArguments` from address 0x58380a6a701c568545dCfcB803FcB875f56beddC4 to Function.callSumWithTwoArguments data: 0xde2...92789

Functions Modifiers

Code :



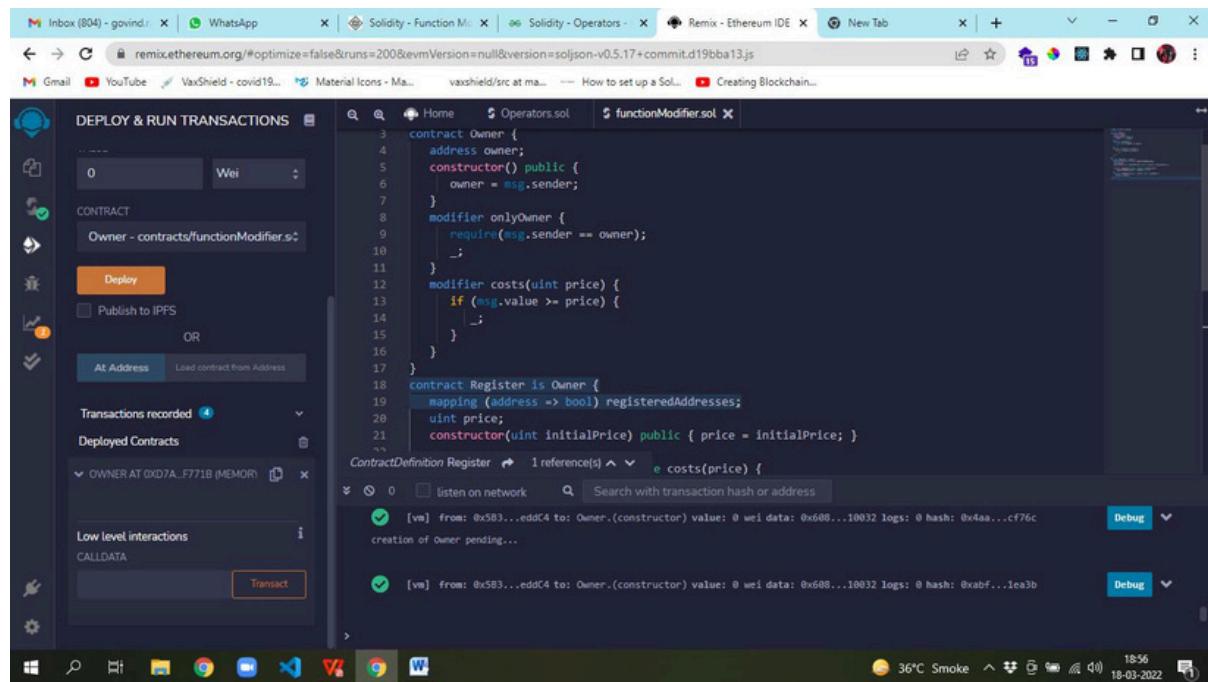
The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file tree under 'FILE EXPLORERS' for a workspace named 'default_workspace'. The tree includes contracts like Variable.sol, Arrays.sol, Enums.sol, Structs.sol, Mapping.sol.sol, SpecialVariable.sol, whileLoop.sol, doWhileLoop.sol, Operators.sol, and functionModifier.sol. The right panel shows the Solidity code for 'functionModifier.sol'.

```
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;
    }
}
```

Output :



The screenshot shows the Remix Ethereum IDE interface with the 'DEPLOY & RUN TRANSACTIONS' tab selected. It displays the deployment of the 'functionModifier.sol' contract. The 'Deploy' button is highlighted. The 'Transactions recorded' section shows two transactions:

- [vm] from: 0x583...eddC4 to: Owner.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x4aa...cf76c creation of owner pending...
- [vm] from: 0x583...eddC4 to: Owner.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0xabf...1ea3b

View function

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file explorer for the workspace, which includes contracts like Variable.sol, Enums.sol, Struct.sol, Mapping.sol, SpecialVariable.sol, whileLoop.sol, doWhileLoop.sol, Operators.sol, functionModifier.sol, and ViewFunction.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;
    }
}

```

Below the code, the deployment status indicates that the contract has been deployed to address 0x5B3...edc4, and a transaction is pending for the getResult() function. The Remix interface also shows the Ethereum network status (36°C, Smoke) and the date (18-03-2022).

Output :

This screenshot shows the Remix IDE after the contract has been deployed. The left sidebar now lists the deployed contracts: VIEWFUNCTION AT 0X7EF_8CB47 and VIEWFUNCTION AT 0XDAO_42B53. The main editor area displays the same Solidity code as before. The deployment status shows the contract has been deployed to address 0x7EF_8CB47, and a transaction is pending for the getResult() function. The Remix interface also shows the Ethereum network status (36°C, Smoke) and the date (18-03-2022).

Pure function

Code :

```

4 // Defining a contract
5 contract PureFunction {
6     // Defining pure function to
7     // calculate product and sum
8     // of 2 numbers
9     function getResult(
10    ) public pure returns(
11        uint product, uint sum){
12        uint num1 = 2;
13        uint num2 = 10;
14        product = num1 * num2;
15        sum = num1 + num2;
16    }
17 }
18 }
19 }
20 }

```

Output:

```

1 pragma solidity ^0.5.0;
2
3 contract PureFunction {
4
5     function getResult(
6    ) public pure returns(
7        uint product, uint sum){
8        uint num1 = 2;
9        uint num2 = 10;
10       product = num1 * num2;
11       sum = num1 + num2;
12    }
13 }
14 }
15

```

Deployed Contracts

PUREFUNCTION AT 0xD2A...FD005

getResult

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

Fallback function

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays the file structure of the workspace, with 'FallbackFunction.sol' selected. The main editor area contains the following Solidity code:

```
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 after deploying the contracts. The left sidebar shows the deployed contracts: 'Caller - contracts/FallbackFunction.sol' and 'PUREFUNCTION AT 0xD2A...FD005'. The main editor area shows the same Solidity code as above. The bottom status bar indicates the transaction was successful.

ContractDefinition Sink → 2 reference(s) ↗ interact,

transact to caller.callTest errored: Error encoding arguments: Error: invalid address (argument="address", value="", code=INVALID_ARGUMENT, version=address)

creation of caller pending...

[vm] from: 0x5B3...eddC4 to: Caller.(constructor) value: 0 wei data: 0x600...10032 logs: 0 hash: 0x527...78cd5

Function Overloading

Code:

```
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 :

```
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);
    }
}
```

Type the library name to see available commands.
creation of FunctionOverloading pending...

[vm] from: 0x5B3...eddc4 to: FunctionOverloading.(constructor) value: 0 wei data: 0x608...10032 logs: 0
hash: 0x9ca...f2a2a
call to FunctionOverloading.callSumWithThreeArguments

[call] from: 0x5B3B0a6a701c568545dCfcB03FcB875f56beddC4 to: FunctionOverloading.callSumWithTwoArguments()
data: 0xd20...4110
call to FunctionOverloading.callSumWithTwoArguments

[call] from: 0x5B3B0a6a701c568545dCfcB03FcB875f56beddC4 to: FunctionOverloading.callSumWithTwoArguments()
data: 0xf20...4110
call to FunctionOverloading.callSumWithTwoArguments

Mathematical Function

Code:

```

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 IDE interface with the Solidity code for MathematicalFunction.sol. The code defines a contract with two pure functions: callAddMod() and callMulMod(). The interface also shows a reference to the fallback function.

The 'DEPLOY & RUN TRANSACTIONS' sidebar indicates the contract has been deployed at address 0x5B3... and provides links to its functions: callSumWithT..., getSum, and callMulWithT... .

Cryptographic function

Code:

The screenshot shows the Remix Ethereum IDE interface. The central area displays the Solidity code for `CryptographicFunction.sol`:

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

The left sidebar shows the file structure of the workspace, which includes contracts like `Variable.sol`, `Arrays.sol`, `Enums.sol`, `Struct.sol`, `Mapping.sol.sol`, `SpecialVariable.sol`, `whileLoop.sol`, `doWhileLoop.sol`, `Operators.sol`, `functionModifier.sol`, `ViewFunction.sol`, `PureFunction.sol`, `FallbackFunction.sol`, `FunctionOverloading.sol`, `MathematicalFunction.sol`, and `CryptographicFunction.sol`. It also lists `scripts`, `tests`, `deps`, and `README.txt`.

Output :

The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" sidebar open. The "CONTRACT" field is set to `CryptographicFunction - contracts/CryptographicFunction.sol`. The "Deploy" button is highlighted.

The central area displays the same Solidity code for `CryptographicFunction.sol` as in the previous screenshot.

The bottom section shows the transaction history under "Transactions recorded". It lists a deployment event:

```
[vm] from: 0x5B3...eddC4 to: CryptographicFunction.(constructor) value: 0 wei data: 0x60...10032 logs: 0
hash: 0x73...b9acb
```

Below this, it shows a call to the `callKeccak256()` function:

```
call to CryptographicFunction.callKeccak256() data: 0x5b4...ea3ee
```

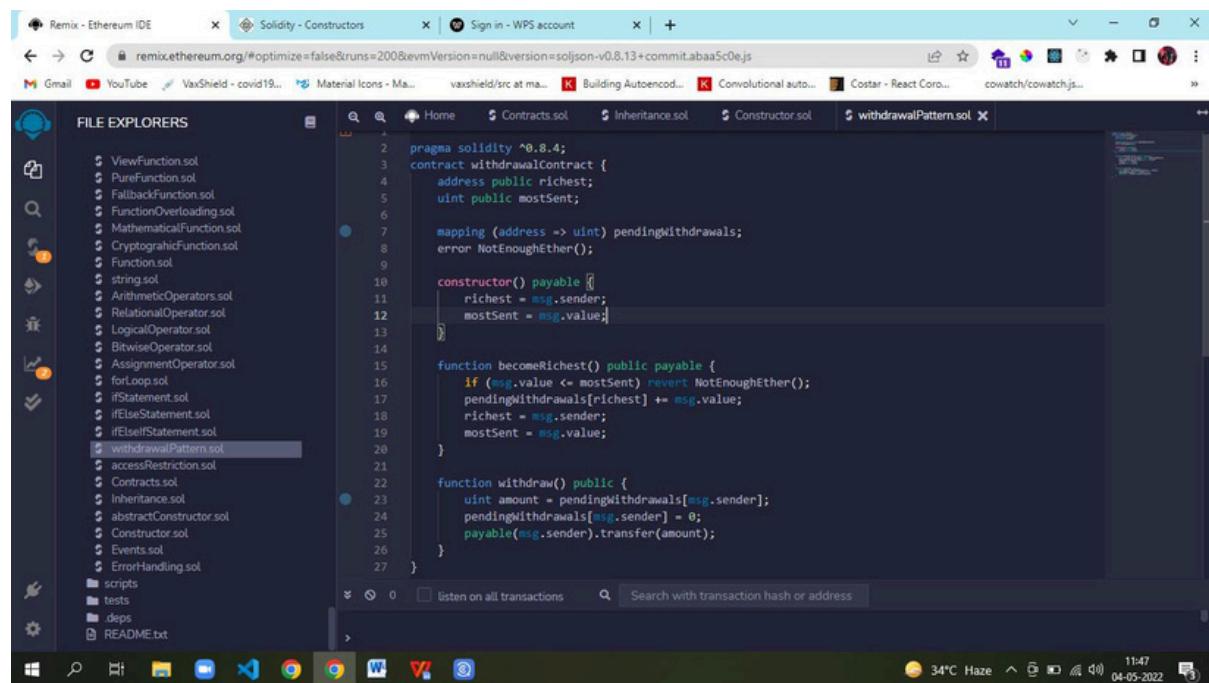
Practical No : 4

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

4a) Withdrawal Pattern, Restricted Access.

Withdrawal Pattern :

Code :



The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORERS" and lists various Solidity files. The main editor window displays the code for "withdrawalPattern.sol". The code implements the Withdrawal Pattern, which restricts access to the withdrawal function to the contract's owner (msg.sender). It also tracks the most sent amount and the address of the current richest user.

```
pragma solidity ^0.8.4;
contract withdrawalContract {
    address public richest;
    uint public mostSent;

    mapping (address => uint) pendingWithdrawals;
    error NotEnoughEther();

    constructor() payable {
        richest = msg.sender;
        mostSent = msg.value;
    }

    function becomeRichest() public payable {
        if (msg.value <= mostSent) revert NotEnoughEther();
        pendingWithdrawals[richest] += msg.value;
        richest = msg.sender;
        mostSent = msg.value;
    }

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

Output :

```

pragma solidity ^0.8.4;
contract WithdrawalContract {
    address public richest;
    uint public mostSent;

    mapping (address => uint) pendingWithdrawals;
    error NotEnoughEther();

    constructor() payable {
        richest = msg.sender;
        mostSent = msg.value;
    }

    function becomeRichest() external {
        require(msg.sender == richest);
        mostSent += pendingWithdrawals[msg.sender];
        pendingWithdrawals[msg.sender] = 0;
    }

    function withdraw() external {
        require(pendingWithdrawals[msg.sender] >= msg.value);
        msg.sender.transfer(msg.value);
        pendingWithdrawals[msg.sender] -= msg.value;
    }

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

```

Restricted Access :

Code :

```

pragma solidity ^0.4.21;

contract AccessRestriction {
    address public owner = msg.sender;
    uint public lastOwnerChange = now;

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

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

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

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

```

The screenshot shows the Remix Ethereum IDE interface. The left sidebar lists various Solidity files under 'FILE EXPLORERS'. The main editor window displays the 'accessRestriction.sol' file. The code defines a contract with a constructor, a modifier 'onlyBy', and a modifier 'onlyAfter'. It includes functions for changing ownership and buying the contract. The status bar at the bottom right shows the date as 04-05-2022 and the time as 12:02.

```
14     }
15   }
16 
17   modifier costs(uint _amount) {
18     require(msg.value >= _amount);
19   }
20   if (msg.value > _amount) {
21     msg.sender.transfer(msg.value - _amount);
22   }
23 
24   function changeOwner(address _newOwner) public onlyBy(owner) {
25     owner = _newOwner;
26   }
27 
28   function buyContract() public payable onlyAfter(lastOwnerChange + 4 weeks) costs(1 ether) {
29     owner = msg.sender;
30     lastOwnerChange = now;
31   }
32 }
```

Ouput :

The screenshot shows the Remix Ethereum IDE interface with the 'DEPLOY & RUN TRANSACTIONS' sidebar open. It displays the deployed contract 'ACCESSRESTRICTION AT 0XAE0...96E'. The 'Low level interactions' section shows the 'buyContract' button highlighted in red, indicating it is the current operation. The status bar at the bottom right shows the date as 04-05-2022 and the time as 12:04.

```
1 pragma solidity ^0.4.21;
2 
3 contract AccessRestriction {
4   address public owner = msg.sender;
5   uint public lastOwnerChange = now;
6 
7   modifier onlyBy(address _account) {
8     require(msg.sender == _account);
9   }
10 
11   modifier onlyAfter(uint _time) {
12     require(now > lastOwnerChange + _time);
13 }
```

```

1 pragma solidity ^0.4.21;
2
3 contract AccessRestriction {
4     address public owner = msg.sender;
5     uint public lastOwnerChange = now;
6
7     modifier onlyBy(address _account) {
8         require(msg.sender == _account);
9     }
10
11     modifier onlyAfter(uint _time) {
12         require(now >= _time);
13     }
14 }

```

Transactions recorded: 16

Deployed Contracts:

- ACCESSRESTRICTION AT 0xae0...96E
- ACCESSRESTRICTION AT 0x908...AS

ContractDefinition AccessRestriction 1 reference(s)

buyContract

changeOwner 0x5830a6a701c568545

lastOwnerCh...

o: uint256 1651646024

owner

o: address 0x5830a6a701c568545dCfB03F...

Low level interactions CALDATA

call to AccessRestriction.lastOwnerChange

call [call] From: 0x5830a6a701c568545dCfB03FcB875f56beddC4 to: AccessRestriction.lastOwnerChange() data: 0x5ff...711ef Debug

call to AccessRestriction.owner

call [call] From: 0x5830a6a701c568545dCfB03FcB875f56beddC4 to: AccessRestriction.owner() data: 0x8d8...5cb5b Debug

transact to AccessRestriction.changeowner pending ...

[vm] From: 0x583...eddC4 to: AccessRestriction.changeOwner(address) 0x9d8...a5692 value: 0 wei data: 0xaef...eddC4 Debug

logs: 0 hash: 0xe9...742a1

4b) Contracts, Inheritance, Constructors, Abstract Contracts, Interfaces.

Contracts:

Code :

```

1 pragma solidity ^0.5.0;
2
3 contract C {
4     //private state variable
5     uint private data;
6
7     //public state variable
8     uint public info;
9
10    //constructor
11    constructor() public {
12        info = 10;
13    }
14    //private function
15    function increment(uint a) private pure returns(uint) { return a + 1; }
16
17    //public function
18    function updateData(uint a) public { data = a; }
19    function getData() public view returns(uint) { return data; }
20    function compute(uint a, uint b) internal pure returns (uint) { return a + b; }
21
22    //External Contract
23    contract D {
24        function readData() public returns(uint) {
25            C c = new C();
26            c.updateData(7);
27            return c.getData();
28        }
29    }
}

```

Remix - Ethereum IDE Solidity - Contracts

remixethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js

Gmail YouTube VaxShield - covid19... Material Icons - Ma... vaxshield/src at ma... Building Autoencod... Convolutional auto... Costar - React Coro... cowatch/cowatch.j... 33°C Haze 11:07 04-05-2022

FILE EXPLORERS

Contracts.sol

```
//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();
    }
}

//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(); }
}
```

ContractDefinition E 1 reference(s) 33°C Haze 11:07 04-05-2022

Output :

Remix - Ethereum IDE Solidity - Contracts

remixethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js

Gmail YouTube VaxShield - covid19... Material Icons - Ma... vaxshield/src at ma... Building Autoencod... Convolutional auto... Costar - React Coro... cowatch/cowatch.j... 33°C Haze 11:08 04-05-2022

DEPLOY & RUN TRANSACTIONS

CONTRACT C - contracts/Contracts.sol Deploy Publish to IPFS OR At Address Load contract from Address

Transactions recorded 1 Deployed Contracts C AT 0xD91...39138 (MEMORY)

updateData uint256 a
getData
o: uint256 0
info
o: uint256: 10

```
//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();
    }
}

//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(); }
}
```

ContractDefinition E 1 reference(s) 33°C Haze 11:08 04-05-2022

```

25     C c = new C();
26     c.updateData(7);
27     return c.getData();
28   }
29 }
//Derived Contract
30 contract E is C {
31   uint private result;
32   C private c;
33
34   constructor() public {
35     c = new C();
36   }

```

Transactions recorded

- C AT 0xD91...39138 (MEMORY)
- updateData
- getData
- info

Low level interactions

CALL DATA

0x58380a6a701c568545dCfcB803FcB875f56beddC4 to: C.info() data: 0x370...15ea

call to C.info

0x58380a6a701c568545dCfcB803FcB875f56beddC4 to: C.updateData() data: 0x3bc...5de30

call to C.updateData

0x6c2...cd89b [vm] from: 0x58380a6a701c568545dCfcB803FcB875f56beddC4 to: C.updateData(uint256) 0xd91...39138 value: 0 wei data: 0x09e...00007 logs: 0

call to C.updateData

0x58380a6a701c568545dCfcB803FcB875f56beddC4 to: C.getData() data: 0x3bc...5de30

call to C.getData

Inheritance

Code :

```

1 pragma solidity ^0.5.0;
2
3 contract C {
4   //private state variable
5   uint private data;
6
7   //public state variable
8   uint public info;
9
10  //constructor
11  constructor() public {
12    info = 20;
13  }
14  //private function
15  function increment(uint a) private pure returns(uint) { return a + 1; }
16
17  //public function
18  function updateData(uint a) public { data = a; }
19  function getData() public view returns(uint) { return data; }
20  function compute(uint a, uint b) internal pure returns (uint) { return a + b; }
21 }
22 //Derived Contract
23 contract E is C {
24   uint private result;
25   C private c;
26   constructor() public {

```

FILE EXPLORERS

- functionModifier.sol
- ViewFunction.sol
- PureFunction.sol
- FallbackFunction.sol
- FunctionOverloading.sol
- MathematicalFunction.sol
- CryptographicFunction.sol
- Function.sol
- string.sol
- ArithmeticOperators.sol
- RelationalOperator.sol
- LogicalOperator.sol
- BitwiseOperator.sol
- AssignmentOperator.sol
- forLoop.sol
- ifStatement.sol
- ifElseStatement.sol
- ifElselfStatement.sol
- withdrawalPattern.sol
- accessRestriction.sol
- Contracts.sol
- Inheritance.sol
- abstractConstructor.sol
- Constructor.sol
- Events.sol
- ErrorHandling.sol
- scripts
- tests
- deps

0x58380a6a701c568545dCfcB803FcB875f56beddC4 to: C.info() data: 0x370...15ea

call to C.info

Remix - Ethereum IDE Solidity - Inheritance

```
functionModifier.sol ViewFunction.sol PureFunction.sol FallbackFunction.sol FunctionOverloading.sol MathematicalFunction.sol CryptographicFunction.sol Function.sol string.sol ArithmeticOperators.sol RelationalOperator.sol LogicalOperator.sol BitwiseOperator.sol AssignmentOperator.sol forLoop.sol ifStatement.sol ifElseStatement.sol ifElseIfStatement.sol withdrawalPattern.sol accessRestriction.sol Contracts.sol Inheritance.sol abstractConstructor.sol Constructor.sol Events.sol Error-handling.sol scripts tests deps
```

```
21 }
22 //Derived Contract
23 contract E is C {
24     uint private result;
25     C private c;
26     constructor() public {
27         c = new C();
28     }
29     function getComputedResult() public {
30         result = compute(3, 5);
31     }
32     function getResult() public view returns(uint) { return result; }
33     function getData() public view returns(uint) { return c.info(); }
34 }
```

33°C Haze 11:12 04-05-2022

Output :

Remix - Ethereum IDE Solidity - Inheritance

```
C - contracts/Inheritance.sol Deploy Publish to IPFS OR At Address Load contract from Address
```

Transactions recorded 5 Deployed Contracts C AT 0X7EF...8CB47 (MEMORY)

```
updateData(uint256 a)
getData()
info()
```

Low level interactions CALDATA

```
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
```

[call] from: 0x5B380a6a701c568545dCfcB03FcB875f56beddC4 to: C.info() data: 0x370...158ea creation of C pending...

[vm] from: 0x5B380a6a701c568545dCfcB03FcB875f56beddC4 to: C.(constructor) value: 0 wei data: 0x600...10032 logs: 0 hash: 0x291...c4630 call to C.getData

[call] from: 0x5B380a6a701c568545dCfcB03FcB875f56beddC4 to: C.getData() data: 0x3b0...5de30 call to C.info

[call] from: 0x5B380a6a701c568545dCfcB03FcB875f56beddC4 to: C.info() data: 0x370...158ea

33°C Haze 11:11 04-05-2022

```

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

```

Transactions recorded:

- [call] From: 0x5B38D0a6a701c568545dCfcB803FcB875f56beddC4 to: C.getData() data: 0x3bc...5de30 call to C.info
- [call] From: 0x5B38D0a6a701c568545dCfcB803FcB875f56beddC4 to: C.info() data: 0x370...15ea transact to C.updateData pending ...
- [vm] From: 0x48c...f9607 hash: 0x48c...f9607 call to C.updateData
- [call] From: 0x5B38D0a6a701c568545dCfcB803FcB875f56beddC4 to: C.getData() data: 0x3bc...5de30 call to C.getData

Constructors :

Code :

```

pragma solidity ^0.5.0;
// Creating a contract
contract constructorExample {
    // Declaring state variable
    string str;
    // Creating a constructor
    // to set value of 'str'
    constructor() public {
        str = "This is Example of Constructor";
    }
    function getValue()
    public view returns (
        string memory)
    {
        return str;
    }
}

```

Output :

The screenshot shows the Remix Ethereum IDE interface. On the left, the sidebar has sections for 'DEPLOY & RUN TRANSACTIONS' and 'CONTRACT'. Under 'CONTRACT', there is a code editor with the following Solidity code:

```

1 pragma solidity ^0.5.0;
2 // Creating a contract
3 contract constructorExample {
4
5     // Declaring state variable
6     string str;
7
8     // Creating a constructor
9     // to set value of 'str'
10    constructor() public {
11        str = "This is Example of Constructor";
12    }
13
14    function getValue()
15        public view returns (
16        string memory) {

```

Below the code editor, the 'Deploy' button is highlighted. To the right, the transaction history shows three entries:

- [vm] from: 0x5B3...eddC4 to: AccessRestriction.changeOwner(address) 0xd8...a5692 value: 0 wei data: 0xa6f...eddc4 creation of constructorExample pending...
- [vm] from: 0x5B3...eddC4 to: constructorExample.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x9e9...89960 call to constructorExample.getValue
- [call] from: 0x5B3B0a6a701c568545d0FcB03Fc8875f56beddC4 to: constructorExample.getValue() data: 0x209...65255

The status bar at the bottom indicates it's 34°C Haze, 12:12, and the date is 04-05-2022.

Abstract Contracts :

Code :

The screenshot shows the Remix Ethereum IDE interface with the 'FILE EXPLORERS' sidebar open. It lists several Solidity files, including 'abstractConstructor.sol' which is currently selected. The code editor displays the following Solidity code:

```

1 pragma solidity ^0.5.0;
2
3 contract abstractConstructor {
4     function getResult() public view returns(uint);
5 }
6 contract Test is abstractConstructor {
7     function getResult() public view returns(uint) {
8         uint a = 10;
9         uint b = 17;
10        uint result = a + b;
11        return result;
12    }
13 }

```

The status bar at the bottom indicates it's 33°C Haze, 12:15, and the date is 04-05-2022.

Output :

```
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;
    }
}
```

Interfaces :

Code :

```
pragma solidity ^0.5.0;

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

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

Output :

```

1 pragma solidity ^0.5.0;
2
3 contract Interface {
4     function getResult() public view returns(uint);
5 }
6 contract Test is Interface {
7     function getResult() public view returns(uint) {
8         uint a = 11;
9         uint b = 67;
10        uint result = a + b;
11        return result;
12    }
13 }

```

4c) Libraries, Assembly, Events, Error handling.

Libraries :

Code :

```

1 pragma solidity ^0.5.0;
2
3 library Search {
4     function indexOf(uint[] storage self, uint value) public view returns (uint) {
5         for (uint i = 0; i < self.length; i++) if (self[i] == value) return i;
6         return uint(-1);
7     }
8 }
9 contract Library {
10     uint[] data;
11     constructor() public {
12         data.push(1);
13         data.push(2);
14         data.push(3);
15         data.push(4);
16         data.push(5);
17     }
18     function isValuePresent() external view returns(uint){
19         uint value = 4;
20         //search if value is present in the array using Library function
21         uint index = Search.indexOf(data, value);
22         return index;
23     }
24 }

```

Output :

```
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;
}
```

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar with icons for deploying contracts, publishing to IPFS, and interacting with deployed contracts. The main area displays the Solidity code for a library contract named `Library.sol`. The code contains a function `isValuePresent` that uses a library function `Search.indexOf` to find a value in an array. Below the code, the transaction history shows two calls to the `isValuePresent` function, one from a VM and one from a user, both returning the correct index values.

Assembly :

Code :

```
pragma solidity ^0.4.0;

contract Assembly {
    function add(uint a) view returns (uint b) {
        assembly {
            let c := add(a, 16)
            mstore(0x00, c)
            {
                let d := add(sload(c), 12)
                // assign the value of 'd' to 'b'
                b := d
                // 'd' is deallocated now
            }
            b := add(b, c)
        }
    }
}
```

This screenshot shows the Remix Ethereum IDE with a Solidity assembly contract named `Assembly.sol`. The code within the `add` function uses assembly language to perform arithmetic operations. It adds 16 to the input `a`, stores the result at memory location `0x00`, and then performs a nested assembly block. Inside this block, it loads the value from memory location `c` (which is 12), assigns it to `d`, and then deallocates `d`. Finally, it adds the value of `c` back to `b`.

Output :

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar titled "DEPLOY & RUN TRANSACTIONS" with options for "Deploy" and "Publish to IPFS". Below that is a list of "Deployed Contracts" under "ASSEMBLY AT 0xD16...23F01 (MEMORY)". A specific contract named "add" is selected, showing its code: `uint256 b = 45;` and a "call" button. The main panel displays the Solidity code for the "Assembly" contract:

```
pragma solidity ^0.4.0;
contract Assembly {
    function add(uint a) view returns (uint b) {
        assembly {
            let c := add(a, 16)
            mstore(0x80, c)
            {
                let d := add(sload(c), 12)
                // assign the value of 'd' to 'b'
            }
        }
    }
}
```

Logs from the transaction are shown below:

- [vm] from: 0x583...eddC4 to: Assembly.(constructor) value: 0 wei data: 0x600...30029 logs: 0 hash: 0x2b5...4839f
- [call] from: 0x5838Da6a701c568545dFc803Fc8875f56beddC4 to: Assembly.add(uint256) data: 0x100...00011

Events:

Code :

The screenshot shows the Remix Ethereum IDE interface with the "FILE EXPLORERS" sidebar open, displaying various Solidity files. The "Events.sol" file is currently selected. The main panel shows the Solidity code for the "Events" contract:

```
// 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;
    }
}
```

Output :

```

1 // creating an event
2 pragma solidity ^0.4.21;
3
4 // Creating a contract
5 contract Events {
6
7     // Declaring state variables
8     uint256 public value = 0;
9
10    // Declaring an event
11    event Increment(address owner);
12
13    // Defining a function for logging event
14    function getValue(uint _a, uint _b) public {
15        emit Increment(msg.sender);
16        value = _a + _b;
17    }

```

Transactions recorded: 45 Deployed Contracts: EVENTS at 0xf28...95BE9 (MEMORY)

Low level interactions: value

value: 500 → 800

transact

ContractDefinition Events → 1 reference(s)

value: 500 → 800

transact to Events.getValue pending ...

[vm] from: 0x583...eddC4 to: Events.getValue(uint256,uint256) 0xf28...95BE9 value: 0 wei data: 0x6a1...0012c logs: 1

call to Events.value

[call] from: 0x58380a6a701c568545dCfc803Fc8875f56beddC4 to: Events.value() data: 0x3fa...4f245

Error handling :

Code :

```

1 pragma solidity ^0.5.0;
2
3 contract ErrorHandling {
4
5     function checkInput(uint _input) public view returns(string memory) {
6         require(_input >= 0, "Invalid uint8");
7         require(_input <= 255, "invalid uint8");
8
9         return "Input is Uint8";
10    }
11
12    // Defining function to use require statement
13    function Odd(uint _input) public view returns(bool) {
14        require(_input % 2 != 0);
15        return true;
16    }
17
18 }

```

Output :

The screenshot shows the Remix Ethereum IDE interface. The left sidebar has sections for "DEPLOY & RUN TRANSACTIONS" (with options for IPFS, Address, or Load contract from Address), "Transactions recorded" (listing one entry: "ERRORHANDLING AT 0xF27...501CB"), and "Deployed Contracts" (listing "checkinput" and "Odd"). The main area displays Solidity code for "ErrorHandling.sol".

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

Below the code, there are two "call" buttons: one for "checkInput" with input "243" and one for "Odd" with input "243". The status bar at the bottom shows system information: 33°C Haze, 12:46, 04-05-2022.