

VALIA C.I. COLLEGE OF COMMERCE & VALIA LC COLLEGE OF ARTS
CES ROAD D.N NAGAR

(Affiliated to University Of Mumbai)

Mumbai-Maharashtra-400053

DEPARTMENT OF INFORMATION TECHNOLOGY



CERTIFICATE

This is to certify that the Journal entitled **BlockChain** is bonafied work of
GOVIND SAINI bearing Roll No: **07** submitted in partial fulfillment of the
requirements for the award of degree of **MASTER OF SCIENCE in**
INFORMATION TECHNOLOGY from University of Mumbai.

Date:

Internal Guide

INDEX

Sr. No.	Practical	Pg. No.
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. 	1
3	<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. 	5
4	<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. 	8

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 file `pract1a.py` open. The code generates an RSA key pair and prints the public key to the console. The terminal below shows the command `python pract1a.py` being run, and the output displays the generated public key.

```
# following imports are required by PKI
from Crypto.PublicKey import RSA

key = RSA.generate(2048)
p_key = key.publickey().export_key("PEM")
priv_key = key.export_key("PEM")
print("7_GovindSaini \n")
print(p_key)
print(priv_key)
```

```
admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN
$ python pract1a.py
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9wBAQEAAQ8AMIIIBGCAQEAxOr0mBTxyybJ+evxJ21\nNNoh70xmw0Dy8gzae/elzklcHeotJK5RzMvGDC+Mj0KKSI
+kDZjaFdvaAO3w\nMpIEQYQX9M02t56aae12ntrK571dkRseTTRmzPfXjUspeChL03YHPhUr/d1/4/0dpd8esrvr/Zanhsjd2s/3zD1pkgyo5ahlgZ1p7YpxoYgeRjzbwu
yDIO/nwxhunqUAubtgJ6iR6u+fmLwLCQFCEt8pLwUwcRkoowCOSg6kkTx+z13gtrVz/vuNAOpTPseumCTUDPe8dgvtWl+jMP2f+8svq831Tnsb1ng0gFy96i7ympybh\nw1IDAQ
AB\n-----END PUBLIC KEY-----
```

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAxOr0mBTxyybJ+evxJ21\nNNoh70xmw0Dy8gzae/elzklcHeotJK5RzMvGDC+Mj0KKSI+kDZjaFdvaAO3w\nMpIEQY
X8M02t56aae12ntrK571dkRseTTRmzPfXjUspeChL03YHPhUr/d1/4/0dpd8esrvr/Zanhsjd2s/3zD1pkgyo5ahlgZ1p7YpxoYgeRjzbwu
yDIO/nwxhunqUAubtgJ6iR6u+fmLwLCQFCEt8pLwUwcRkoowCOSg6kkTx+z13gtrVz/vuNAOpTPseumCTUDPe8dgvtWl+jMP2f+8svq831Tnsb1ng0gFy96i7ympybh\nw1IDAQ
AB\n-----END RSA PRIVATE KEY-----
```

Output :

The screenshot shows the Visual Studio Code interface with the file `pract1a.py` open. The terminal below shows the command `python pract1a.py` being run, and the output displays the generated public key.

```
# following imports are required by PKI
from Crypto.PublicKey import RSA

key = RSA.generate(2048)
p_key = key.publickey().export_key("PEM")
priv_key = key.export_key("PEM")
print("7_GovindSaini \n")
print(p_key)
print(priv_key)
```

```
admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN
$ python pract1a.py
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9wBAQEAAQ8AMIIIBGCAQEAxOr0mBTxyybJ+evxJ21\nNNoh70xmw0Dy8gzae/elzklcHeotJK5RzMvGDC+Mj0KKSI
+kDZjaFdvaAO3w\nMpIEQYQX9M02t56aae12ntrK571dkRseTTRmzPfXjUspeChL03YHPhUr/d1/4/0dpd8esrvr/Zanhsjd2s/3zD1pkgyo5ahlgZ1p7YpxoYgeRjzbwu
yDIO/nwxhunqUAubtgJ6iR6u+fmLwLCQFCEt8pLwUwcRkoowCOSg6kkTx+z13gtrVz/vuNAOpTPseumCTUDPe8dgvtWl+jMP2f+8svq831Tnsb1ng0gFy96i7ympybh\nw1IDAQ
AB\n-----END PUBLIC KEY-----
```

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAxOr0mBTxyybJ+evxJ21\nNNoh70xmw0Dy8gzae/elzklcHeotJK5RzMvGDC+Mj0KKSI+kDZjaFdvaAO3w\nMpIEQY
X8M02t56aae12ntrK571dkRseTTRmzPfXjUspeChL03YHPhUr/d1/4/0dpd8esrvr/Zanhsjd2s/3zD1pkgyo5ahlgZ1p7YpxoYgeRjzbwu
yDIO/nwxhunqUAubtgJ6iR6u+fmLwLCQFCEt8pLwUwcRkoowCOSg6kkTx+z13gtrVz/vuNAOpTPseumCTUDPe8dgvtWl+jMP2f+8svq831Tnsb1ng0gFy96i7ympybh\nw1IDAQ
AB\n-----END RSA PRIVATE KEY-----
```

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:

```
pract1b.py - BLOCKCHAIN - Visual Studio Code
File Edit Selection View Go Run Terminal Help
pract1d.py pract1e.py pract1f.py pract1a.py pract1b.py pract1c.py p.py
EXPLORER
BLOCKCHAIN
SolidityProject
~.govindsaini(Blockchain_Pract1).d...
7_govindsaini(Blockchain_Pract1).d...
7_govindsaini(Blockchain_Pract2).d...
Doct.docx
Doct.pdf
p.py
pract1a.py
pract1b.py
pract1c.py
pract1d.py
pract1e.py
pract1f.py
pract1b.py > Bank
1 class Bank:
2     def __init__(self):
3         self.balance = 0
4         print("7_Govindsaini \n")
5         print ("The account is created")
6
7     def deposit(self):
8         amount = float(input("Enter the amount to be deposit: "))
9         self.balance = self.balance + amount
10        print ("The deposit is successful and the balance in the account is %f" % self.balance)
11
12     def withdraw(self):
13         amount = float(input("Enter the amount to withdraw: "))
14         if (self.balance >= amount):
15             self.balance = self.balance - amount
16             print ("The withdraw is successful and the balance is %f" % self.balance)
17         else:
18             print ('Insufficient Balance')
19
20     def enquiry(self):
21         print ("Balance in the account is %f" % self.balance)
22
23
24 acc = Bank()
25 acc.deposit()
26 acc.withdraw()
27 acc.enquiry()

Ln 19 Col 5 Spaces:4 UTF-8 CRLF Python 3.9.7 (.venv: venv) Prettier 19:43 35°C Smoke 18-03-2022
```

Output:

The screenshot shows the Visual Studio Code interface with the terminal tab active. The terminal displays the execution of the 'pract1b.py' script and its output. The output shows the creation of the account, deposits, withdrawals, and balance inquiries.

```
pract1b.py - BLOCKCHAIN - Visual Studio Code
File Edit Selection View Go Run Terminal Help
pract1d.py pract1e.py pract1f.py pract1a.py pract1b.py pract1c.py p.py
EXPLORER
BLOCKCHAIN
SolidityProject
~.govindsaini(Blockchain_Pract1).d...
7_govindsaini(Blockchain_Pract1).d...
7_govindsaini(Blockchain_Pract2).d...
Doct.docx
Doct.pdf
p.py
pract1a.py
pract1b.py
pract1c.py
pract1d.py
pract1e.py
pract1f.py
pract1b.py > Bank
1 class Bank:
2     def __init__(self):
3         self.balance = 0
4         print("7_Govindsaini \n")
5         print ("The account is created")
6
7     def deposit(self):
8         amount = float(input("Enter the amount to be deposit: "))
9         self.balance = self.balance + amount
10        print ("The deposit is successful and the balance in the account is %f" % self.balance)
11
12     def withdraw(self):
13         amount = float(input("Enter the amount to withdraw: "))
14         if (self.balance >= amount):
15             self.balance = self.balance - amount
16             print ("The withdraw is successful and the balance is %f" % self.balance)
17         else:
18             print ('Insufficient Balance')
19
20     def enquiry(self):
21         print ("Balance in the account is %f" % self.balance)
22
23
24 acc = Bank()
25 acc.deposit()
26 acc.withdraw()
27 acc.enquiry()

PROBLEMS 2 OUTPUT TERMINAL DEBUG CONSOLE
bash + ^ X
admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN
$ py pract1b.py
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

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

1c) Create Multiple Transaction and display them.

Code :

A screenshot of Visual Studio Code showing a Python project named "BLOCKCHAIN". The Explorer sidebar shows files like .venv, p.py, pract1.py, pract1b.py, pract1cp.py, pract1d.py, pract1e.py, and pract1f.py. The main editor tab is open to pract1cp.py, displaying code for a blockchain. The terminal tab shows the command "admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN" and a prompt "\$". The status bar at the bottom right shows the date and time as 19-03-2022.

```
#defining the list/chain
blockchain = []
print("7_GovindSaini \n")
#getting the last value/transaction
def get_last_value():
    return(blockchain[-1])

#adding transaction now we have sender, recipient and amount
def add_value(sender, recipient, amount=1.0):
    transaction = {'sender': sender,
                   'recipient': recipient,
                   'amount': amount}
    blockchain.append(transaction)

#getting the details of transaction by entering to the prompt
def get_transaction_value():
    tx_sender = input('Enter the sender: ')
    tx_recipient = input('Enter the recipient of the transaction: ')
    tx_amount = float(input('Enter your transaction amount: '))
    return tx_sender, tx_recipient, tx_amount

#printing the blockchain
def print_block():
    for block in blockchain:
        print("7_GovindSaini \n")
        print("Here is your block")
        print(block)
```

A screenshot of Visual Studio Code showing a modified Python project. The Explorer sidebar includes a "SolidityProject" folder containing ".~\$govindSaini(Blockchain_Pract1).d..." and ".7_govindSaini(Blockchain_Pract2).d...". The main editor tab is open to pract1cp.py, showing a modified version of the blockchain code. The terminal tab shows the same command as before. The status bar at the bottom right shows the date and time as 18-03-2022.

```
tx_sender = input('Enter the sender: ')
tx_recipient = input('Enter the recipient of the transaction: ')
tx_amount = float(input('Enter your transaction amount: '))
return tx_sender, tx_recipient, tx_amount

#printing the blockchain
def print_block():
    for block in blockchain:
        print("Here is your block")
        print(block)

#the code will keep repeating for more transaction
#until the user answer is no
again = True
while again == True:
    tx = get_transaction_value()
    s, r, a = tx
    add_value(s, r, a)
    print(blockchain)
    more = input("add more block (Y/N)? ")
    if more.lower() == 'y':
        again = True
    else:
        again = False
```

Output :

A screenshot of Visual Studio Code showing a terminal window running a Python script named `pract1c.py`. The script simulates a blockchain transaction between multiple users. The terminal output shows the following interaction:

```
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
```

The status bar at the bottom indicates the terminal is running on `admin@DESKTOP-3B61180 MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN`.

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

Code :

A screenshot of Visual Studio Code showing a terminal window running a Python script named `pract1d.py`. The script defines a blockchain structure with a genesis block and adds transactions. The terminal output shows the following code and execution:

```
genesis_block = {
    'previous_hash': '',
    'index': 0,
    'transaction': [],
    'nonce': 23
}

blockchain = [genesis_block]

def get_last_value():
    return(blockchain[-1])

def add_value(recipient, sender, amount=1.0):
    transaction = {'sender': sender,
                  'recipient': recipient,
                  'amount': amount}
    open_transactions.append(transaction)

def get_transaction_value():
    tx_sender = raw_input('Enter the sender: ')
    tx_recipient = raw_input('Enter the recipient of the transaction: ')
    tx_amount = float(input('Enter your transaction amount: '))
    return tx_sender, tx_recipient, tx_amount

def get_user_choice():
    pass
```

The status bar at the bottom indicates the terminal is running on `admin@DESKTOP-3B61180 MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN`.

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

- File Explorer:** Shows a project structure under "BLOCKCHAIN" containing files like .venv, SolidityProject, Doc1.docx, p.py, pract1a.py, pract1b.py, pract1c.py, and pract1d.py.
- Code Editor:** Displays the content of pract1d.py. The code defines a blockchain with a genesis block, transaction addition, value retrieval, user choice, and printing of the block.
- Terminal:** Shows the command \$ py pract1d.py followed by the output "7_Govindsaini".
- Status Bar:** Provides information such as the file path (admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN), terminal type (bash), and system status (35°C Smoke).

Output :

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

- File Explorer:** Shows a project structure under "BLOCKCHAIN" containing files like .venv, SolidityProject, Doc1.docx, p.py, pract1a.py, pract1b.py, pract1c.py, and pract1d.py.
- Code Editor:** Displays the content of pract1d.py. The code defines a blockchain with a genesis block, transaction addition, value retrieval, user choice, and printing of the block.
- Terminal:** Shows the command \$ py pract1d.py followed by the output "7_Govindsaini". It then prints "Here is your block" and shows the block structure: {'previous_hash': '', 'index': 0, 'transaction': [], 'nonce': 23}.
- Status Bar:** Provides information such as the file path (admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN), terminal type (bash), and system status (35°C Smoke).

1e) Create a mining function and test it.

Code :

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

- File Explorer:** Shows a folder named "BLOCKCHAIN" containing several Python files: .venv, p.py, pract1a.py, pract1b.py, pract1c.py, pract1d.py, and pract1f.py.
- Code Editor:** Displays a Python script named "pract1e.py". The code implements a basic blockchain mining system. It includes a SHA256 hashing function, a mining function that generates a new hash based on the previous hash and a nonce, and a main loop that tries to find a valid nonce within a range of MAX_NONCE values. If successful, it prints a message and returns the new hash. If not, it raises a BaseException.
- Terminal:** Shows the command "admin@DESKTOP-3B61180 MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN" followed by a prompt "\$".
- Status Bar:** Provides information about the file (Ln 18, Col 1), code style (Spaces: 4, UTF-8, CRLF, Python 3.9.7), and system status (29°C, 19:56, 19-03-2022).

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

- File Explorer:** Shows a folder named "BLOCKCHAIN" containing several Python files: .venv, p.py, pract1a.py, pract1b.py, pract1c.py, pract1d.py, and pract1f.py.
- Code Editor:** Displays a Python script named "pract1e.py". This version of the code includes additional print statements. It prints the current time before starting mining, the difficulty level, and the total time taken for mining. It also prints the new hash value once mining is complete.
- Terminal:** Shows the command "admin@DESKTOP-3B61180 MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN" followed by a prompt "\$".
- Status Bar:** Provides information about the file (Ln 18, Col 1), code style (Spaces: 4, UTF-8, CRLF, Python 3.9.7), and system status (29°C, 19:56, 19-03-2022).

Output :

```

File Edit Selection View Go Run Terminal Help
pract1d.py pract1e.py pract1f.py pract1a.py pract1b.py pract1c.py p.py
EXPLORER ... pract1e.py > ...
BLOCKCHAIN .venv
p.py
pract1b.py
pract1c.py
pract1d.py
pract1e.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->Siddu->0,
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
30     admin@DESKTOP-3B61180 MINING64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN
31     $ py pract1e.py
32     7_GovindSaini
33
34     start mining
35     Yay! Successfully mined bitcoins with nonce value:27405
36     end mining. Mining took: 0.23560523986816406 seconds
37     the new hash value : 0000605c84ff941938f28ae1c0050a71dbfb692a22eadcaa0c2ff6af4dd9d927e7
38
39     admin@DESKTOP-3B61180 MINING64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN
40     $ 

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Ln 18, Col 1 Spaces: 4 UTF-8 CRLF Python 3.9.7 (.venv: venv) ⚡ Prettier 19:57 29°C Smoke 19-03-2022

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

Code :

```

File Edit Selection View Go Run Terminal Help
pract1d.py 3 pract1e.py 1 pract1f.py pract1a.py pract1b.py pract1c.py p.py
EXPLORER ... pract1f.py > ...
BLOCKCHAIN .venv
SolidityProject
~$govindSaini(Blockchain_Prac1).d...
7_govindSaini(Blockchain_Prac2).d...
Doc1.docx
Doc1.pdf
p.py
pract1a.py
pract1b.py
pract1c.py
pract1d.py
pract1e.py
pract1f.py
1
2     import datetime
3     import hashlib
4
5     class Block:
6         blockNo = 0
7         data = None
8         next = None
9         hash = None
10        nonce = 0
11        previous_hash = 0x0
12        timestamp = datetime.datetime.now()
13
14        def __init__(self, data):
15            self.data = data
16
17        def hash(self):
18            h = hashlib.sha256()
19            h.update(
20                str(self.nonce).encode('utf-8') +
21                str(self.data).encode('utf-8') +
22                str(self.previous_hash).encode('utf-8') +
23                str(self.timestamp).encode('utf-8') +
24                str(self.blockNo).encode('utf-8')
25            )
26            return h.hexdigest()
27
28        def __str__(self):
29            return "Block Hash: " + self.hash() + "\nBlockNo: " + str(self.blockNo) + "\nBlock Data: " + str(self.data)
30
31
32     admin@DESKTOP-3B61180 MINING64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN
33     $ py pract1f.py
34

```

PROBLEMS 5 OUTPUT TERMINAL DEBUG CONSOLE

Ln 66, Col 43 Spaces: 4 UTF-8 CRLF Python 3.9.7 (.venv: venv) ⚡ Prettier 19:52 35°C Smoke 18-03-2022

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

- File Explorer:** Shows a project named "BLOCKCHAIN" containing files: .venv, SolidityProject, ~\$govindsaini(Blockchain_Pract1).d..., ~\$govindsaini(Blockchain_Pract2).d..., Doc1.docx, Doc1.pdf, p.py, pract1a.py, pract1b.py, pract1c.py, pract1d.py, pract1e.py, and pract1f.py.
- Code Editor:** The file "pract1f.py" is open, displaying Python code for a blockchain. The code defines a class "Blockchain" with methods for adding blocks and mining. It includes a __str__ method for printing the blockchain structure.
- Terminal:** The terminal shows the command \$ py pract1f.py being run.
- Status Bar:** Shows the current file is pract1f.py, the line number is Ln 66, column 43, and the file size is 3.97 MB.

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 file "pract1f.py" is open, displaying Python code for a blockchain. The code now includes a loop that mines 10 blocks and prints the blockchain's head after each block is added.
- Terminal:** The terminal shows the command \$ py pract1f.py being run.
- Status Bar:** Shows the current file is pract1f.py, the line number is Ln 66, column 43, and the file size is 3.97 MB.

Output :

File Edit Selection View Go Run Terminal Help

pract1f.py - BLOCKCHAIN - Visual Studio Code

EXPLORER ... pract1d.py 3 pract1e.py 1 pract1f.py x pract1a.py 1 pract1b.py pract1c.py p.py

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

pract1f.py > Block

```
1 import datetime
2 import hashlib
3
4 print("7_GovindSaini \n")
```

PROBLEMS 5 OUTPUT TERMINAL DEBUG CONSOLE

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

```
$ py pract1f.py
7_GovindSaini
```

Block Hash: 23da69f580363b47569cff724f7ffe96993dc92d96f3086cccc2b254022fb583

BlockNo: 1
Block Data: Block 1
Hashes: 11641

Block Hash: 2cbd5120ab93b2c7136384fb57b08b1afec571940e6c1561f01d5991aabfe1c4

BlockNo: 2
Block Data: Block 2
Hashes: 1732856

Block Hash: c815950bc3663c736ab5266862e600207a5964d73eafaddb1ac2b2592e28810b

BlockNo: 3
Block Data: Block 3
Hashes: 98750

Block Hash: 85ffdb7fa35bc3ff8f586c4a1da6540d8c61838552b57a8b36b486bb2885e772

BlockNo: 4
Block Data: Block 4
Hashes: 407313

Block Hash: 1df3b0b0aa2036bd6bcc89daf857195eb4e609e0a1312de6b869f29db5d9dd

BlockNo: 5
Block Data: Block 5
Hashes: 2562101

Ln 9, Col 16 Spaces: 4 UTF-8 CRLF Python 3.9.7 (.venv: venv) Prettier 2001 35°C Smoke 18-03-2022

File Edit Selection View Go Run Terminal Help

pract1f.py - BLOCKCHAIN - Visual Studio Code

EXPLORER ... pract1d.py 3 pract1e.py 1 pract1f.py x pract1a.py 1 pract1b.py pract1c.py p.py

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

pract1f.py > Block

```
1 import datetime
2 import hashlib
3
4 print("7_GovindSaini \n")
```

PROBLEMS 5 OUTPUT TERMINAL DEBUG CONSOLE

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

```
$
```

Block Hash: 42e3ba3d954744af6ddd34fa476d8e213a661ee81fe8bb96fdb4ecace1cd0d

BlockNo: 6
Block Data: Block 6
Hashes: 280879

Block Hash: d2ad82f546889bb1fcf13d96b3acfad47af84182e8577905239e9a537ced2a

BlockNo: 7
Block Data: Block 7
Hashes: 5204675

Block Hash: f01fd4a85de5a39646b6d16e5623c1946101cc9f7940eef56d79966e986eb12c

BlockNo: 8
Block Data: Block 8
Hashes: 299732

Block Hash: 2e50b68a0b03022e4b35b1c96b3dfbefdd9c47276d5bec6b390f72fe345a19b

BlockNo: 9
Block Data: Block 9
Hashes: 1378275

Block Hash: c5c4a0f2c5e5b683c66c11eec2b585588597ce5e86f999cd672dc79cd40529c

BlockNo: 10
Block Data: Block 10
Hashes: 2953554

Ln 9, Col 16 Spaces: 4 UTF-8 CRLF Python 3.9.7 (.venv: venv) Prettier 2002 35°C Smoke 18-03-2022

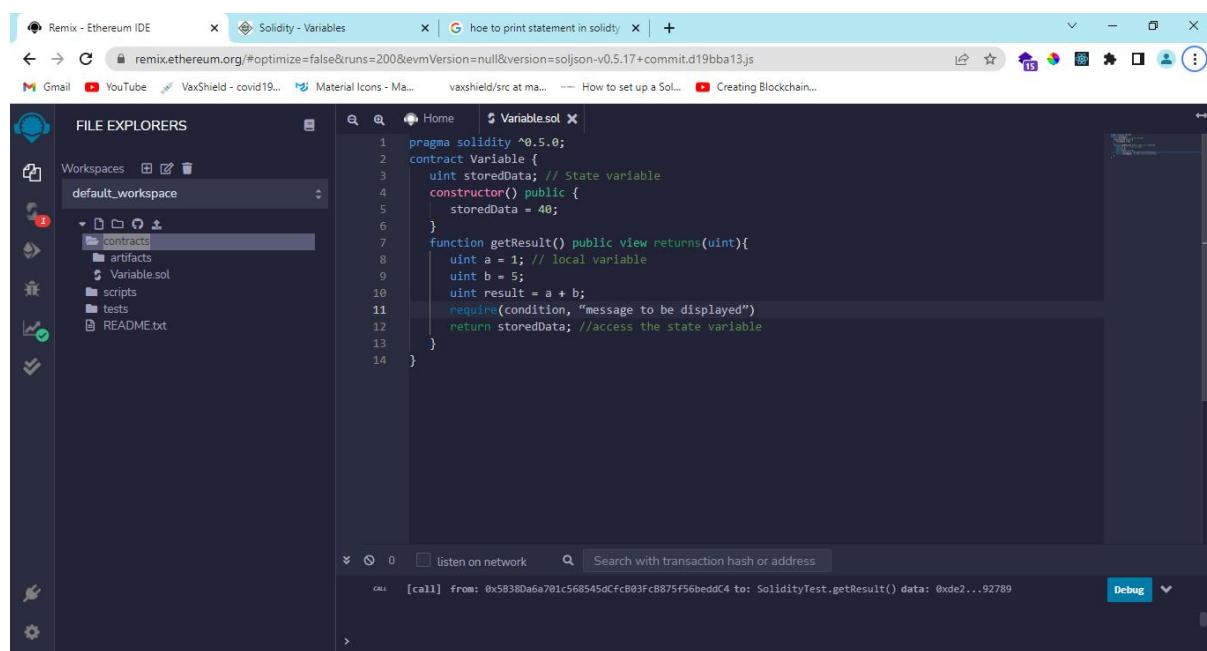
Practical No : 3

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 :

```

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
    }
}

```

The screenshot shows the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is visible, showing a deployed contract named 'Variable' at address 0xD91...39138. Below it, the 'Transactions recorded' section shows a transaction for 'getResult' with a value of 40. The main central area displays the Solidity code for 'Variable.sol'. At the bottom, the status bar indicates a temperature of 34°C, a battery level of 32%, and the date/time 18-03-2022 13:10.

Operators :

Arthmetic Operator

Code:

```

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;
}

```

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 a file named 'ArithmeticOperators.sol' which is currently selected. The main central area displays the Solidity code for 'ArithmeticOperators.sol'. At the bottom, the status bar indicates a temperature of 32°C, a battery level of 32%, and the date/time 18-03-2022 20:58.

Output:

```

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;
}

```

```

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;
}

```

Relational Operator

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file explorer with various Solidity files like ArithmeticOperators.sol and RelationalOperator.sol. The main editor window contains the following Solidity 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;

    bool public gtreq = a >= b;
    bool public leseq = a <= b;
}
```

The status bar at the bottom indicates it's 32°C, the time is 21:03, and the date is 18-03-2022.

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 different relational operators: a, b, eq, gtr, gtreq, les, leseq, and noteq. The main editor window shows the same Solidity code as above. The bottom pane displays transaction logs:

- call to RelationalOperator.les
- call to RelationalOperator.noteq
- call [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: RelationalOperator.les() data: 0xb61...2aa97
- call to RelationalOperator.noteq
- call [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: RelationalOperator.noteq() data: 0x6a8...a5ab5

The status bar at the bottom indicates it's 32°C, the time is 21:06, and the date is 18-03-2022.

The screenshot shows the Remix Ethereum IDE interface. In the center, there is a code editor window titled "RelationalOperator.sol" containing Solidity code for creating a relational operator contract. The code defines variables `a` and `b` of type `uint16`, and functions `eq`, `noteq`, `gtr`, `les`, `gtreq`, and `lesreq` that return boolean values based on the comparison of `a` and `b`. To the left of the code editor is a "DEBUGGER" panel. A message in this panel states: "This call has reverted, state changes made during the call will be reverted. Click here to jump where the call reverted." Below this message, the "Solidity State" section shows variable values: `a: 70 uint16`, `b: 10 uint16`, `eq: false bool`, `noteq: true bool`, `gtr: true bool`, `les: false bool`, `gtreq: true bool`, and `lesreq: false bool`. At the bottom of the debugger panel, it says "140 REVERT" followed by assembly code: 141 JUMPDEST, 142 PUSH2 0095, 145 PUSH2 01a5, 148 JUMP, 149 JUMPDEST. The status bar at the bottom right indicates "32°C Smoke" and the date "18-03-2022".

Logical Operator

Code :

The screenshot shows the Remix Ethereum IDE interface. On the left, there is a "FILE EXPLORERS" panel showing a workspace named "default_workspace" containing several Solidity files: Variable.sol, Arrays.sol, Enums.sol, Struct.sol, Mapping.sol.sol, SpecialVariable.sol, whileLoop.sol, doWhileLoop.sol, functionModifier.sol, ViewFunction.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, Functions.sol, string.sol, ArithmeticOperators.sol, RelationalOperator.sol, LogicalOperator.sol, scripts, tests, and .deps. The file "LogicalOperator.sol" is currently selected and shown in the main code editor window. The code defines a contract `logicalOperator` with a single function `Logic` that takes two boolean parameters `a` and `b`, and returns three boolean values: `bool and = a&b;`, `bool or = a|b;`, and `bool not = !a;`. The status bar at the bottom right 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 with options like 'Deploy' and 'Publish to IPFS'. The main area displays the Solidity code for a logical operator contract:

```

1 pragma solidity ^0.5.0;
2
3 // Creating a contract
4 contract logicalOperator{
5
6     function Logic(
7         bool a, bool b) public view returns(
8             bool, bool, bool){
9
10        // Logical AND operator
11        bool and = a&b;
12
13        // Logical OR operator
14        bool or = a||b;
15
16        // Logical NOT operator
17        bool not = !a;
18    }
19
20}

```

Below the code, there's a 'Logic' section with inputs 'true' and 'false' and a 'call' button. The status bar at the bottom indicates a transaction hash: 0x7d5...7ae0a.

Bitwise Operator

Code:

The screenshot shows the Remix Ethereum IDE interface with the BitwiseOperator.sol file selected in the workspace. The code defines a contract with various bitwise operations:

```

1 pragma solidity ^0.5.0;
2
3 // Creating a contract
4 contract BitwiseOperator {
5
6     // Declaring variables
7     uint16 public a = 20;
8     uint16 public b = 50;
9
10    uint16 public and = a & b;
11
12    uint16 public or = a | b;
13
14    uint16 public xor = a ^ b;
15
16    uint16 public leftshift = a << b;
17
18    uint16 public rightshift = a >> b;
19
20    uint16 public not = ~a ;
21
22}

```

The left sidebar shows a 'FILE EXPLORERS' section with a 'Workspaces' tree containing multiple Solidity files like 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 various operators like and, or, xor, etc., each with a corresponding icon and value. The main code editor contains 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;
}
```

The "ContractDefinition BitwiseOperator" section shows "1 reference(s)". Below the code, the "Debug" tab is active, displaying logs from the VM:

- [vm] from: 0x5B3...eddC4 to: logicalOperator.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x426...4889c Debug
- call to logicaloperator.Logic
- [vm] [call] from: 0x5B3...eddC4 to: logicalOperator.Logic(bool,bool) data: 0xc94...00000 Debug
- call to logicaloperator.b
- call to logicaloperator.b errored: VM error: revert.

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

This screenshot shows the Remix IDE with the "DEBUGGER" tab selected. The sidebar on the left displays the "Solidity State" with 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 shows the creation of the contract:

```
creation of Bitwiseoperator pending...
```

The logs pane shows the creation of the contract and the deployment of the logic function:

- [vm] from: 0x5B3...eddC4 to: BitwiseOperator.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x682...24e33 Debug
- creation of Bitwiseoperator pending...
- [vm] from: 0x5B3...eddC4 to: BitwiseOperator.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0xd7c...dfe8e Debug
- call to Bitwiseoperator.a

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

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.sol, SpecialVariable.sol, whileLoop.sol, doWhileLoop.sol, functionModifier.sol, ViewFunction.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, CryptographicFunction.sol, Function.sol, and string.sol. The central code editor window shows the 'ArithmeticOperators.sol' contract:

```
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;
}
```

The status bar at the bottom indicates the date as 18-03-2022 and the time as 20:58.

Ouput :

The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file tree under 'default_workspace' containing 'AssignmentOperator.sol'. The central code editor window shows the 'AssignmentOperator.sol' contract:

```
pragma solidity ^0.5.0;
// Creating a contract
contract AssignmentOperator {
    // Declaring variables
    uint16 public assignment = 20;
    uint public assignment_add = 50;
    uint public assignment_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 *= 10;
        assignment_div /= 10;
        assignment_mod %= 20;
        return ;
    }
}
```

The status bar at the bottom indicates the date as 18-03-2022 and the time as 21:33.

```

pragma solidity ^0.5.0;
// Creating a contract
contract AssignmentOperator {
    // Declaring variables
    uint16 public assignment = 20;
    uint public assignment_add = 50;
    uint public assignment_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;
    }
}

```

Loops

whileLoop

code:

```

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 whileLoop {
    // 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 whileLoop {
    // Declaring a dynamic array
    uint[] data;
    // Declaring state variable
    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 displays a file explorer with several Solidity files listed under 'default_workspace'. The main editor area contains 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' tab selected. The left sidebar shows the 'Function Stack' with a single entry '0: loop()'. The main editor area shows the same Solidity code as before. The bottom right corner of the editor shows a status bar indicating 'creation of doWhileLoop pending...'. The bottom right of the screen shows a system tray with a temperature of 31°C, a battery icon, and the date/time 18-03-2022 21:42.

Forloop

Code :

```

1 pragma solidity ^0.5.0;
2
3 // Creating a contract
4 contract forLoop {
5
6     // Declaring a dynamic array
7     uint[] data;
8
9     // Defining a function
10    // to demonstrate 'For loop'
11    function loop()
12        public returns(uint[] memory){
13        for(uint i=0; i<4; i++){
14            data.push(i);
15        }
16        return data;
17    }
18
19 }

```

Output :

Function Stack

- 0: loop()

Solidity State

data: uint256[]
[length: 4]
0: 0 uint256
1: 1 uint256
2: 2 uint256
3: 3 uint256

Assembly

```

140 JUMPDEST
144 PUSH1 60
146 PUSH1 00
148 DUP1
149 SWAP1
150 POP
151 JUMPDEST

```

Decision making

If statement

Code:

The screenshot shows the Remix Ethereum IDE interface. The top navigation bar has tabs for "Solidity - Operators - GeeksforGeeks", "Remix - Ethereum IDE", and "Solidity - Decision Making Statement". The main area displays the Solidity code for an "ifStatement" contract:

```
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;
        }
    }
}
```

The left sidebar shows a file explorer with various Solidity files like "artifacts", "Variable.sol", "Arrays.sol", etc. The bottom status bar shows system information: 30°C, Smoke, 21:48, 18-03-2022.

Code :

The screenshot shows the Remix Ethereum IDE with the "DEBUGGER" tab selected. The code for the "ifStatement" contract is visible in the editor. The debugger interface includes a "FUNCTION STACK" section showing "o_decision_making()", a "SOLIDITY LOCALS" section, and a "SOLIDITY STATE" section showing "i: uint256". The bottom pane displays the assembly code:

```
0x17689840bb32e24d84f49d8a9a71...  
Stop debugging  
transaction cost 23386 gas 0  
execution cost 23386 gas 0  
input 0x887...3ef24 0  
decoded input {} 0  
decoded output [{}]  
{"0": "bool: false"}  
logs [] 0  
val 0 wei 0
```

The bottom status bar shows system information: 30°C, Smoke, 21:50, 18-03-2022.

If...else statement

Code :

The screenshot shows the Remix Ethereum IDE interface. The left sidebar lists various Solidity files. The main editor window 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;
    }
}
```

Output :

The screenshot shows the Remix Ethereum IDE with the debugger open. The left sidebar shows the debugger configuration and stack trace. The main editor window shows the same Solidity code as before. The bottom status bar indicates the transaction hash and a pending status.

Debugger Configuration:

- Use generated sources (from Solidity v0.7.2)

Stack Trace:

- 0x36c1fca1e6f53c36928167d66386f...

Function Stack:

- o: decision_making()

Solidity Locals:

- i: 20 uint256
- even: false bool

Solidity State:

- 0000 PUSH1 00
- 0001 DUP1
- 0002 PUSH1 02
- 0003 PUSH1 00
- 0004 SLOAD

Logs:

- [vm] from: 0x5B3...eddC4 to: Types.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0xc54...d4493 transact to IfElseStatement.decision_making pending ...
- [vm] from: 0x5B3...eddC4 to: Types.decision_making() 0xe28...4157A value: 0 wei data: 0x887...3af24 logs: 0 hash: 0x36c...de7fc

If...else if...else statement

Code :

The screenshot shows the Remix Ethereum IDE interface. The left sidebar contains a file explorer with various Solidity files like 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, ArithmeticOperators.sol, RelationalOperator.sol, LogicalOperator.sol, BitwiseOperator.sol, AssignmentOperator.sol, forLoop.sol, ifStatement.sol, ifElseStatement.sol, scripts, deploy_web3.js, deploy_ethers.js, tests, .deps, and README.txt. The main code editor window displays the following Solidity code:

```

1 pragma solidity ^0.5.0;
2
3 // Creating a contract
4 contract ifElseStatement {
5
6     // Declaring state variables
7     uint i = 12;
8     string result;
9
10    function decision_making()
11        public returns(string memory){
12        if(i<10){
13            result = "less than 10";
14        }
15        else if(i == 10){
16            result = "equal to 10";
17        }
18        else{
19            result = "greater than 10";
20        }
21
22        return result;
23    }
24
25 }

```

The status bar at the bottom indicates it's 30°C, Smoke, 21:58, and the date is 18-03-2022.

Output:

The screenshot shows the Remix Ethereum IDE with the debugger open. The left sidebar has a debugger configuration section with a checked checkbox for "Use generated sources (from Solidity v0.7.2)" and a transaction hash "0xb64e7d4791ff2a7d41a38718735...". The main code editor window shows the same Solidity code as before. The status bar at the bottom indicates it's 30°C, Smoke, 21:59, and the date is 18-03-2022.

The debugger interface includes a "Function Stack" showing "o: decision_making()", "Solidity Locals" showing "i: 12 uint256", and "Solidity State" showing "result: greater than 10 string". The assembly code pane shows:

```

179 JUMPDEST
180 PUSH1 60
182 PUSH1 0a
184 PUSH1 00

```

Two transactions are listed in the bottom right:

- [vm] from: 0x5B3...eddC4 to: ifElseStatement.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x418...bb224 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: 0xab6...a28fd

String

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file explorer with several Solidity files listed under the 'default_workspace' folder. The main editor window contains the following Solidity 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 status bar at the bottom right indicates the date and time as 18-03-2022 19:32.

Output :

The screenshot shows the Remix Ethereum IDE interface with the 'DEPLOY & RUN TRANSACTIONS' tab selected. The left sidebar shows the deployed contract 'String' at address 0x358...D5eE3. The main editor window shows the same Solidity code as before. The transaction history pane on the right lists several transactions:

- [vm] from: 0x5B3...eddC4 to: String.(constructor) value: 0 wei data: 0x608...d0033 logs: 0 hash: 0x6e1...28ba9 transact to String.pushToRow pending ...
- [vm] from: 0x5B3...eddC4 to: String.pushToRow(string) 0x358...D5eE3 value: 0 wei data: 0xf75...00000 logs: 0 hash: 0x34c...53160 call to String.pushToRow
- [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: String.getRow() data: 0x52d...c3caa call to String.getRow

The status bar at the bottom right indicates the date and time as 18-03-2022 19:34.

Array

Code :

The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file explorer with a workspace named 'default_workspace' containing contracts like 'Variable.sol' and 'Arrays.sol'. The main editor window shows the Solidity code for 'Arrays.sol' which defines a test contract with a function 'testArray()' that asserts the length of arrays 'a' and 'b' to be 7, and array 'c' to have a length of 3. The bottom status bar shows system information including temperature and date.

```
pragma solidity ^0.5.0;

contract test {
    function testArray() public pure{
        uint len = 7;
        //dynamic array
        uint[] memory a = new uint[](7);

        //bytes is same as byte[]
        bytes memory b = new bytes(len);

        assert(a.length == 7);
        assert(b.length == len);

        //access array variable
        a[6] = 8;

        //test array variable
        assert(a[6] == 8);
        //static array
        uint[3] memory c = [uint(1), 2, 3];
        assert(c.length == 3);
    }
}
```

Output:

The screenshot shows the Remix Ethereum IDE interface with the 'DEPLOY & RUN TRANSACTIONS' tab selected. It displays the deployment of the 'test' contract from 'Contracts/Arrays.sol'. The 'Transactions recorded' section shows a successful deployment of the 'testArray' function. The right side of the screen shows the transaction history and logs, including calls to the deployed contract's functions. The bottom status bar shows system information including temperature and date.

Enums

Code :

The screenshot shows the Remix Ethereum IDE interface. The top navigation bar has tabs for "Remix - Ethereum IDE" and "Solidity - Enums". The main workspace displays the following Solidity 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);
    }
}
```

The left sidebar shows a file tree for the workspace, including contracts, tests, and remix-test files. The bottom status bar shows network connection, temperature (36°C), and date (18-03-2022).

Output :

The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" tab selected. The workspace contains the same Solidity code as above. The left sidebar shows the deployed contract "TEST AT 0xD91...39138 (MEMORY)". The bottom section displays transaction history and low-level interactions:

- setLarge**: A button labeled "creation of Enums pending...".
- getChoice**: A button labeled "0: uint8: 0".
- getDefaultCho...**: A button labeled "0: uint256: 1".

Logs from transactions:

- [vm] from: 0x5B3...eddC4 to: Enums.(constructor) value: 0 wei data: 0x600...10032 logs: 0 hash: 0xfbb...e3a9c call to Enums.getChoice
- [call] from: 0x5B3...a6a701c568545dCfcB03FcB875f56beddC4 to: Enums.getChoice() data: 0x67c...b61b6 call to Enums.getDefaultchoice
- [call] from: 0x5B3...a6a701c568545dCfcB03FcB875f56beddC4 to: Enums.getDefaultChoice() data: 0x843...f7258 call to Enums.getDefaultChoice

The bottom status bar shows network connection, temperature (36°C), and date (18-03-2022).

Struct

Code:

The screenshot shows the Ethereum IDE interface. On the left, the File Explorer sidebar displays a workspace named 'default_workspace' containing contracts, artifacts, scripts, tests, and dependencies. The main editor window 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 Deploy & Run Transactions sidebar indicates that the contract has been deployed at address 0x091...3913B. The main editor window shows the same Solidity code as before, but now includes transaction logs for the deployed 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) {
    }
}
```

The transaction logs show the creation of the contract and the execution of the 'setBook' and 'getBookId' functions.

Mapping

Code:

The screenshot shows the Remix Ethereum IDE interface. On the left, the FILE EXPLORERS panel displays a workspace named 'default_workspace' containing several Solidity files: Contracts (Variable.sol, Arrays.sol, Enums.sol, Struct.sol, Mapping.sol.sol), Scripts (deploy_web3.js, deploy_ether.js), Tests (.deps, remix-tests, remix-tests.sol, remix_accounts.sol), and a README.txt file. The central 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 a new entry to the mapping. The code is annotated with comments explaining the structure definition and mapping creation.

```
contract Mapping {
    //Defining structure
    struct student {
        //Declaring different
        // structure elements
        string name;
        string subject;
        uint8 marks;
    }

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

    // Function adding values to the mapping
    function adding_values() public {
        var student
        = result[0xDEE7796E89C82C36BAdd1375076f39D69FafE252];

        student.name = "John";
        student.subject = "Chemistry";
        student.marks = 88;
        student_result.push(
            0xEE7796E89C82C36BAdd1375076f39D69FafE252) -1;
    }
}
```

Output :

The screenshot shows the Remix Ethereum IDE interface with the 'DEPLOY & RUN TRANSACTIONS' panel active. This panel allows users to interact with deployed contracts. In the 'Deployed Contracts' section, there is a listing for 'MAPPING_EXAMPLE AT 0x540...C756'. Below this, a transaction history shows two entries for the 'adding_values()' function. The first transaction is pending, and the second has been executed successfully. The transaction details include the from address (0x583...eddC4), to address (mapping_example), function call (adding_values()), value (0 wei), data (0x548...7729d), logs (0), and hash (0xae9...9312f). There is also a 'Debug' button next to the transaction details.

The screenshot shows the Remix Ethereum IDE interface. The top navigation bar has tabs for Remix - Ethereum IDE, Inbox (803) - govind.r.saini@..., WhatsApp, Solidity - Special Variables, Solidity - Mappings - Geeksforgeeks, and others. The main area displays Solidity code for a `Mapping` contract:

```

contract Mapping {
    struct student {
        //Declaring different
        // structure elements
        string name;
        string subject;
        uint8 marks;
    }

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

    // Function adding values to the mapping
    function adding_values() public {
        ContractDefinition Mapping 1 reference(s) ^ v
    }
}

```

The left sidebar shows the **DEBUGGER** section with **Solidity Locals** and **Solidity State**. The **Solidity State** pane lists the `result` mapping with five entries, each containing an address and a `student` struct. The bottom status bar shows the date as 18-03-2022 and the time as 18:22.

Special Variable

Code:

The screenshot shows the Remix Ethereum IDE interface with multiple tabs open. The active tab is `Special Variables`. The code editor contains the following Solidity code for a `SpecialVariable` contract:

```

pragma solidity ^0.6.6;

// Creating a smart contract
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.
    function whatIsMyRollNumber()
    | public view returns (uint)
    {
        return rollNo[msg.sender];
    }
}

```

The left sidebar shows the **FILE EXPLORERS** section with various Solidity files listed. The bottom status bar shows the date as 18-03-2022 and the time as 22:16.

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 one contract named 'SPECIALVARIABLE AT 0xC3B...AAEC' expanded. This expanded section shows two functions: 'setRollNO' and 'whatIsMyRollNo'. The 'setRollNO' function has a parameter 'o: uint256: 7'. The 'whatIsMyRollNo' function has a return type 'o: uint256: 7'. In the main central area, a Solidity code editor displays the following code:

```
pragma solidity ^0.6.6;
// Creating a smart contract
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.
}
```

Below the code editor, the transaction history shows two calls to the contract:

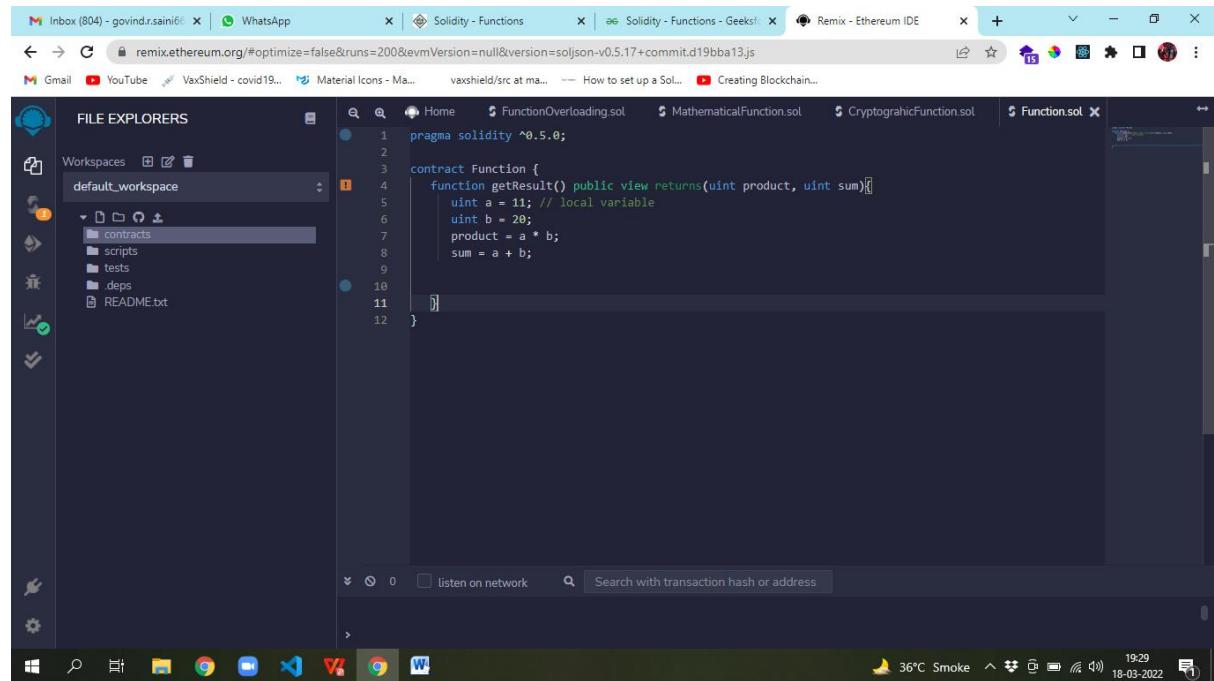
- [call] from: 0x58380a6a701c568545dCfc803FcB875f56beddC4 to: SpecialVariable.whatIsMyRollNumber() data: 0x3ee...b5710
- [call] from: 0x58380a6a701c568545dCfc803FcB875f56beddC4 to: SpecialVariable.setRollNO(uint256) 0xC3B...aaECA value: 0 wei data: 0x15e...00007 logs: 0

The bottom status bar indicates the system temperature is 30°C, the time is 22:17, and the date is 18-03-2022.

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

Function

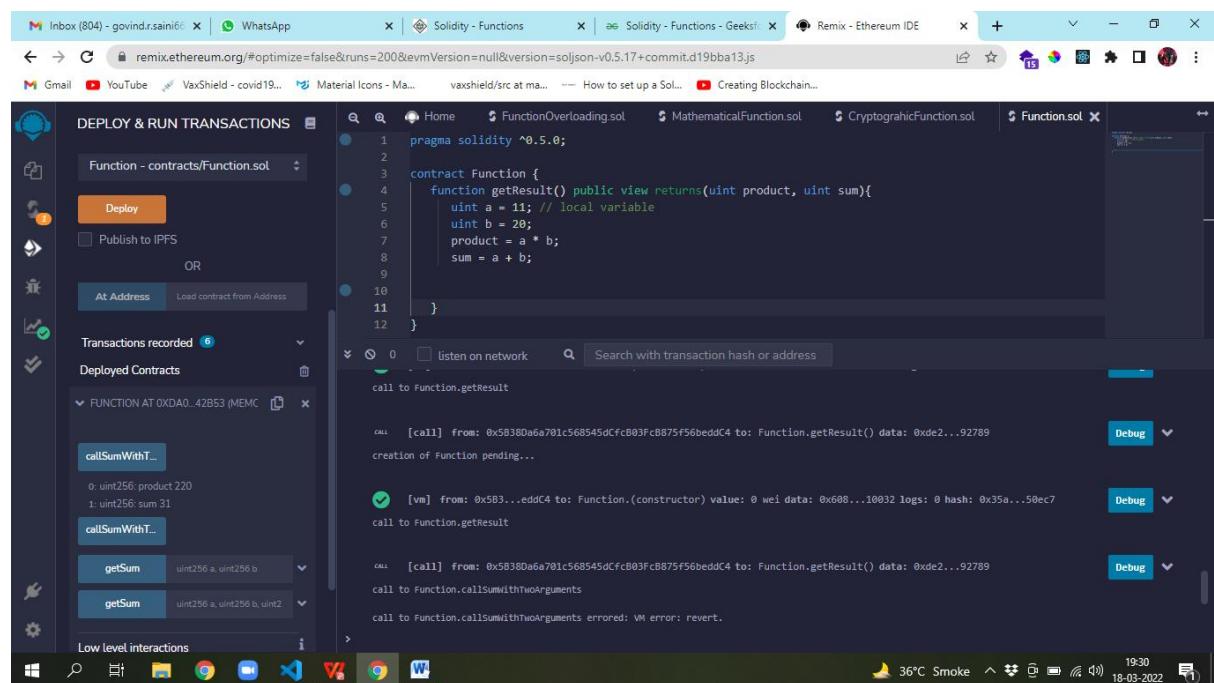
Code:



```
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 'DEPLOY & RUN TRANSACTIONS' sidebar is open, displaying the deployed contract 'Function' at address 0xDA0...42B53. It shows two transaction logs: one for the constructor call and another for a call to the 'getResult()' function. The main code editor window contains the same Solidity code as the previous screenshot. The status bar at the bottom right indicates it's 36°C, 19:29, and the date is 18-03-2022.

Functions Modifiers

Code :

The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file tree for a workspace named 'default_workspace'. Under the 'contracts' folder, several files are listed: artifacts, Variable.sol, Arrays.sol, Enums.sol, Struct.sol, Mapping.sol, whileLoop.sol, doWhileLoop.sol, Operators.sol, and functionModifier.sol, which is currently selected. The main editor area 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;
    }
}
```

Output :

The screenshot shows the Remix Ethereum IDE interface with the 'DEPLOY & RUN TRANSACTIONS' tab active. The deployment section shows the 'Wei' input field set to 0 and the 'Deploy' button highlighted. The contract dropdown shows 'Owner - contracts/functionModifier.sol'. Below it, there are options to 'Publish to IPFS' or 'At Address' (with a 'Load contract from Address' button). The 'Transactions recorded' section shows 4 entries under 'Deployed Contracts' with the address 'OWNER AT 0xD7A...F771B (MEMORY)'. The code editor area is identical to the previous screenshot. The bottom pane shows the transaction history with two successful transactions:

- [vm] from: 0x5B3...eddC4 to: Owner.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x4aa...cf76c creation of Owner pending...
- [vm] from: 0x5B3...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 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', 'Operators.sol', 'functionModifier.sol', and 'ViewFunction.sol'. The main editor area shows 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;
    }
}
```

The status bar at the bottom indicates a temperature of 36°C, a battery level of 19:01, and the date 18-03-2022.

Output :

The screenshot shows the Remix Ethereum IDE interface with the 'DEPLOY & RUN TRANSACTIONS' tab selected. The left sidebar shows deployed contracts including 'VIEWFUNCTION AT 0X7EF...8CB47' and 'VIEWFUNCTION AT 0XA0...42B53'. The main editor area shows the same Solidity code as before. The status bar at the bottom indicates a temperature of 36°C, a battery level of 19:03, and the date 18-03-2022.

Pure function

Code :

```

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

```

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

```

Fallback function

Code:

The screenshot shows the Remix IDE interface. The left sidebar displays a file tree under 'default_workspace' containing various Solidity files like Operators.sol, functionModifier.sol, ViewFunction.sol, PureFunction.sol, and FallbackFunction.sol. The main editor window shows the FallbackFunction.sol 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 IDE interface with the 'DEPLOY & RUN TRANSACTIONS' tab selected. It displays the deployment of the Caller contract, which has deployed the FallbackFunction contract at address 0x0D2A...FD005. The transaction log shows the deployment of the Caller contract and the creation of the FallbackFunction contract.

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: 0x60B...10032 logs: 0 hash: 0x527...78cd5

Function Overloading

Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file tree for a workspace named 'default_workspace'. The central code editor window contains the following Solidity 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);
    }
}
```

The status bar at the bottom right indicates the date and time as 18-03-2022 19:15.

Output :

The screenshot shows the Remix Ethereum IDE interface, specifically the 'DEPLOY & RUN TRANSACTIONS' tab. The code editor window contains the same Solidity code as above. The left sidebar shows a deployed contract named 'FUNCTIONOVERLOADING AT 0xD91'. The bottom section displays a list of transactions recorded, with the first few being:

- [VM] from: 0x583...eddc4 to: FunctionOverloading.(constructor) value: 0 wei data: 0x608...10032 logs: 0
- call to FunctionOverloading.callSumWithThreeArguments
- [CALL] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: FunctionOverloading.callSumWithTwoArguments
- call to FunctionOverloading.callSumWithTwoArguments
- [CALL] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: FunctionOverloading.callSumWithTwoArguments

The status bar at the bottom right indicates the date and time as 18-03-2022 19:17.

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 :

```

[vm] from: 0x583...edd4 to: MathematicalFunction.(constructor) value: 0 wei data: 0x608...10032 logs: 0
hash: 0xc21...0f06d
call to MathematicalFunction.callAddMod()

[call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: MathematicalFunction.callAddMod() data: 0xf9b...41691
call to MathematicalFunction.callMulMod()

[call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: MathematicalFunction.callMulMod() data: 0xaad4...e8744

```

Cryptographic function

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, Operators.sol, functionModifier.sol, ViewFunction.sol, PureFunction.sol, FallbackFunction.sol, FunctionOverloading.sol, MathematicalFunction.sol, and CryptographicFunction.sol. The right panel shows the Solidity code for 'CryptographicFunction.sol':

```
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. It displays a transaction record for deploying the 'CryptographicFunction' contract. The transaction details are as follows:

- Value: 0 Wei
- Contract: CryptographicFunction - contracts/CryptographicFunction.sol
- Deploy button is orange.
- Publish to IPFS checkbox is unchecked.
- At Address: Load contract from Address
- Transactions recorded: 4
- Deployed Contracts: CRYPTOGRAPHICFUNCTION AT 0xD7...
- callSumWithT... (with a small note below it: 0: bytes32 result 0xe1629b9dd060bb30c790 8346f6af189c16773f148d3366701fbba35d 54f3d8)
- callSumWithT... (another entry with the same details)

The transaction record for the deployment is shown in the bottom right:

- [vm] from: 0x5B3...eddC4 to: CryptographicFunction.(constructor) value: 0 wei data: 0x608...10032 logs: 0
- hash: 0x073...b9ac
- call to CryptographicFunction.callKeccak256

The transaction record for the first function call is also shown:

- [call] from: 0x5B3D0a6a701c568545dCfcB03FcB875f56beddC4 to: CryptographicFunction.callKeccak256() data: 0x5b4...8a3ee
- hash: 0x073...b9ac
- call to CryptographicFunction.callKeccak256

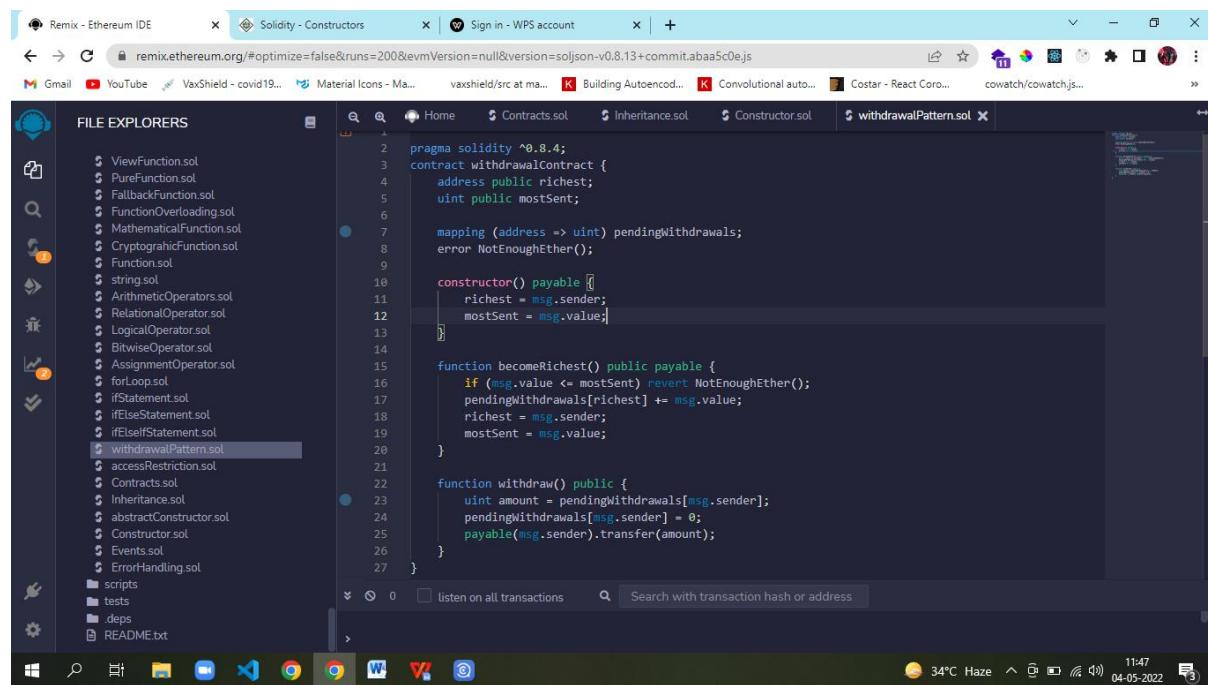
Practical No : 4

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

4a) Withdrawal Pattern, Restricted Access.

Withdrawal Pattern :

Code :



```
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);
    }
}
```

Ouput :

```

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

    function becomeRichest() public {
        if (msg.value > mostSent) {
            mostSent = msg.value;
        }
    }

    function mostSent() public 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) onlyAfter(lastOwnerChange + 4 weeks) costs(1 ether) {
        owner = _newOwner;
    }
}

```

Remix - Ethereum IDE | Solidity - Constructors | Sign in - WPS account | Access Restriction | solidity-patter...

FILE EXPLORERS

```

14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
        }
    }

    modifier costs(uint _amount) {
        require(msg.value >= _amount);
    }

    if (msg.value > _amount) {
        msg.sender.transfer(msg.value - _amount);
    }

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

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

```

ContractDefinition AccessRestriction → 1 reference(s) ▾

34°C Haze 12:02 04-05-2022

Output :

Remix - Ethereum IDE | Solidity - Constructors | Sign in - WPS account | Access Restriction | solidity-patter...

DEPLOY & RUN TRANSACTIONS

At Address Load contract from Address

Transactions recorded 15

Deployed Contracts

ACCESSRESTRICTION AT 0xae0...96E

ACCESSRESTRICTION AT 0x9db...A5

buyContract

changeOwner address _newOwner

lastOwnerCh...

owner

creation of AccessRestriction pending...

[vm] from: 0x503...eddC4 to: AccessRestriction.(constructor) value: 0 wei data: 0x608...30029 logs: 0

call to AccessRestriction.lastOwnerChange

[call] from: 0x58380da6a701c568545dCfB03Fcb03FcB875f56beddC4 to: AccessRestriction.lastOwnerChange() data: 0x5ff...71lef

call to AccessRestriction.owner

[call] from: 0x58380da6a701c568545dCfB03FcB875f56beddC4 to: AccessRestriction.owner() data: 0x8da...5cb5b

34°C Haze 12:04 04-05-2022

The screenshot shows the Remix Ethereum IDE interface. The top navigation bar has tabs for Remix - Ethereum IDE, Solidity - Constructors, Sign in - WPS account, Access Restriction | solidity-patter..., and others. The main area displays the Solidity code for the `AccessRestriction` contract:

```

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
12    modifier onlyAfter(uint _time) {
13        require(now >= _time);
14    }
15}

```

The left sidebar shows the deployment interface with sections for DEPLOY & RUN TRANSACTIONS, Transactions recorded (16), Deployed Contracts, and ACCESSRESTRICTION AT 0XAE0_96E. It includes buttons for buyContract, changeOwner, and lastOwnerChange, along with their respective addresses and values.

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

Contracts:

Code :

The screenshot shows the Remix Ethereum IDE interface with the tab Solidity - Contracts selected. The main area displays the Solidity code for the `Contracts.sol` file:

```

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 }

```

The left sidebar shows the FILE EXPLORERS section with a list of other contracts like ViewFunction.sol, PureFunction.sol, etc., and a list of files including scripts, tests, deps, and README.txt.

The screenshot shows the Remix Ethereum IDE interface. The title bar reads "Remix - Ethereum IDE" and "Solidity - Contracts". The left sidebar is titled "FILE EXPLORERS" and lists several Solidity source files: 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, ErrorHandling.sol, scripts, tests, .deps, and README.txt. The file "Contracts.sol" is currently selected. The main editor area displays the Solidity code for 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(); }
}
```

The status bar at the bottom shows system information: 33°C Haze, 11:07, 04-05-2022.

Output :

The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" tab selected. The title bar remains the same: "Remix - Ethereum IDE" and "Solidity - Contracts". The left sidebar now includes a "DEPLOY" button and a "Transactions recorded" section. The main editor area shows the same Solidity code as before. The status bar at the bottom shows system information: Very high UV, 11:08, 04-05-2022.

The screenshot shows the Remix Ethereum IDE interface. The top navigation bar has tabs for "Remix - Ethereum IDE" and "Solidity - Contracts". The main workspace displays Solidity code for a contract named `C`. The code includes a constructor that initializes a private state variable `c` and a function `updateData` that updates a public state variable `data`. A derived contract `E` is defined, which inherits from `C` and adds a new function `info`. The left sidebar shows a file tree with `Contracts.sol` selected. The bottom right corner shows system status: 33°C Haze, 11:08, 04-05-2022.

```
25     C c = new C();
26     c.updateData(7);
27     return c.getData();
28   }
29 }
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 }
```

Inheritance

Code :

The screenshot shows the Remix Ethereum IDE interface. The top navigation bar has tabs for "Remix - Ethereum IDE" and "Solidity - Inheritance". The main workspace displays Solidity code for two contracts: `C` and `E`. Contract `C` has a constructor that sets a private state variable `info` to 20. It also has a function `increment` that increments a public state variable `data` by 1. Contract `E` is a derived contract that inherits from `C` and adds a new function `compute` that adds two public state variables `a` and `b`. The left sidebar shows a file tree with multiple test files like `Function.sol`, `String.sol`, etc., and `Inheritance.sol` selected. The bottom right corner shows system status: 33°C Haze, 11:12, 04-05-2022.

```
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 
15  //private function
16  function increment(uint a) private pure returns(uint) { return a + 1; }
17 
18  //public function
19  function updateData(uint a) public { data = a; }
20  function getData() public view returns(uint) { return data; }
21  function compute(uint a, uint b) internal pure returns (uint) { return a + b; }
22 }
23 //Derived Contract
24 contract E is C {
25   uint private result;
26   C private c;
27   constructor() public {
28     c = new C();
29   }
30 }
```

The screenshot shows the Remix Ethereum IDE interface. The top bar displays the title "Solidity - Inheritance" and the URL "remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13js". The left sidebar is titled "FILE EXPLORERS" and lists various Solidity files. The main editor window contains the following Solidity code:

```
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 }
```

Output :

The screenshot shows the Remix Ethereum IDE interface with the title "Solidity - Inheritance" and the same URL as the previous screenshot. The left sidebar is titled "DEPLOY & RUN TRANSACTIONS" and shows the deployed contract "C AT 0x7EF...8CB47 (MEMORY)". The main editor window shows the same Solidity code as before. The bottom section displays the transaction history and logs:

- A successful deployment log: "[call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: C.info() data: 0x370...158ea creation of C pending..."
- A successful constructor call log: "[vm] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: C.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x291...c4630 call to c.info
- A successful getData call log: "[call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: C.getData() data: 0x3bc...5de30 call to c.info
- A successful info call log: "[call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: C.info() data: 0x370...158ea call to c.info"

```

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: 6

Deployed Contracts: C AT 0x7EF...8CB47 (MEMORY)

updateData 17

getData

o: uint256: 17

info

o: uint256: 20

Low level interactions

CALLDATA

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, there's a sidebar titled "DEPLOY & RUN TRANSACTIONS" with options like "Deploy", "Publish to IPFS", and "Transactions recorded". Below it, "Deployed Contracts" lists "CONSTRUCTOREXAMPLE AT 0x5FD...". A preview window shows the contract's code and a message: "0: string. This is Example of Constructor". On the right, the main editor window contains 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, the transaction history shows two entries:

- [vm] from: 0x5B3...eddC4 to: AccessRestriction.changeOwner(address) 0x9d8...a5692 value: 0 wei data: 0xa6f...eddc4 logs: 0 hash: 0x9e9...742a1 creation of constructorExample pending...
- [vm] from: 0x5B3...eddC4 to: constructorExample.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x9a9...98960 call to constructorExample.getValue

At the bottom, the status bar indicates "34°C Haze" and the date "04-05-2022".

Abstract Contracts :

Code :

The screenshot shows the Remix Ethereum IDE interface. On the left, the "FILE EXPLORERS" sidebar lists various Solidity files. The file "abstractConstructor.sol" is currently selected. The main editor window contains the following Solidity code:

```

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

```

At the bottom, the status bar indicates "33°C Haze" and the date "04-05-2022".

Output :

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar titled "DEPLOY & RUN TRANSACTIONS" with options like "Deploy" and "Transactions recorded". The main code editor contains the following Solidity 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;
    }
}

```

On the right, the "ContractDefinition Test" pane shows two transaction logs:

- [vm] from: 0x5B3...eddC4 to: Test.(constructor) value: 0 wei data: 0x600...10032 logs: 0 hash: 0xfaf...86e41 call to Test.getResult
- [call] from: 0x5B38D0a6a701c568545dCfcB03FcB875f56beddC4 to: Test.getResult() data: 0xde2...92789

Interfaces :

Code :

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar titled "FILE EXPLORERS" with a list of files including "Interface.sol". The main code editor contains the following Solidity 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 :

```

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;
    }
}

```

ContractDefinition Test → 1 reference(s) ▾

- 0 listen on all transactions Search with transaction hash or address
- [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: Test.getResult() data: 0xde2...92789 creation of Test pending...
- [vm] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: Test.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x776...05ecf call to Test.getResult
- [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: Test.getResult() data: 0xde2...92789

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

Libraries :

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 Library {
    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 :

```

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
21     //search if value is present in the array using Library function
22     uint index = Search.indexOf(data, value);
23
24     return index;
25 }

```

Contract Definition Test:

- [vm] from: 0x5B3...eddC4 to: Library.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x92c...f92aa call to Library.isValuePresent
- [call] from: 0x5B3D0a6a701c568545dCfcB03FcB8875f56beddC4 to: Library.isValuePresent() data: 0x1d3...21459

Assembly :

Code :

```

1 pragma solidity ^0.4.0;
2
3 contract Assembly {
4
5     function add(uint a) view returns (uint b) {
6
7         assembly {
8
9             let c := add(a, 16)
10            mstore(0x80, c)
11
12             let d := add(sload(c), 12)
13             // assign the value of 'd' to 'b'
14             b := d
15             // 'd' is deallocated now
16         }
17         b := add(b, c)
18
19     }
20 }

```

Contract Definition Assembly:

- [vm] from: 0x5B3...eddC4 to: Assembly.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x92c...f92aa
- [call] from: 0x5B3D0a6a701c568545dCfcB03FcB8875f56beddC4 to: Assembly.add() data: 0x1d3...21459

Output :

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar titled "DEPLOY & RUN TRANSACTIONS" with options like "Deploy" and "Publish to IPFS". The main area displays the Solidity code for the "Assembly" contract:

```

1 pragma solidity ^0.4.0;
2
3 contract Assembly {
4
5     function add(uint a) view returns (uint b) {
6
7         assembly {
8
9             let c := add(a, 16)
10            mstore(0x80, c)
11
12             let d := add(sload(c), 12)
13             // assign the value of 'd' to 'b'
}

```

Below the code, it says "ContractDefinition Assembly" and "1 reference(s)". The "Deployed Contracts" section shows "ASSEMBLY AT 0xD16_23F01 (MEMORY)". Under "Low level interactions", there's a "call" button next to the "add" function. The status bar at the bottom shows "33°C Smoke" and the date "04-05-2022".

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 selected. The main code editor shows the following Solidity code:

```

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    }
18
19 }

```

The status bar at the bottom shows "33°C Haze" and the date "04-05-2022".

Outut :

```
// 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;
    }
}
```

Error handling :

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";
    }

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

Output :

