

## **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 (Left):** Shows a project structure under "BLOCKCHAIN". The "pract1" folder contains several files: pract1.py, pract1e.py, pract1f.py, pract1t.py, pract1b.py, pract1c.py, pract1d.py, and pract1e.py. A ".venv" folder and a "SolvayProject" folder are also visible.
- Code Editor (Top Right):** Displays the content of pract1.py, which generates RSA keys and exports them as PEM files.
- Terminal (Bottom):** Shows the command "python pract1.py" being run in a terminal window titled "bash". The output indicates the script is executing successfully.
- Status Bar (Bottom):** Provides information about the current file (Line 11, Col 1), code style (Spaces: 4, Tab Size: 8), encoding (UTF-8), and file type (Python). It also shows the date and time (14-03-2022).

## **Output :**

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

## Code:

## Output:

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

- File Explorer:** Shows a project structure under "BLOCKCHAIN". The file "practib.py" is the active editor.
- Code Editor:** Displays the code for "practib.py". The code defines a class "Bank" with methods for depositing and withdrawing funds.
- Terminal:** Shows the command-line output of running the script with Python 3.6.0. It prints the account creation message, prompts for deposit, shows the successful deposit message, and then prompts for withdrawal. Finally, it shows the successful withdrawal message and the updated balance.
- Status Bar:** Shows the current file is "practib.py", line 19, column 5, with 4 spaces, using UTF-8 encoding, and CRLF line endings.

**1c ) Create Multiple Transaction and display them.**

### **Code :**

```
File Edit Selection View Go Run Terminal Help pract1cp - BLOCKCHAIN - Visual Studio Code
pract1d.py pract1e.py pract1f.py pract1b.py pract1cp p.py
pract1cp > print_block
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

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN
$ 
Ln 27, Col 20 Spaces:3 UTF-8 CRLF Python 3.9.7 (.venv:venv) ⚡ Prettier 🖼 19:44
File Edit Selection View Go Run Terminal Help pract1cp - BLOCKCHAIN - Visual Studio Code
pract1d.py pract1e.py pract1f.py pract1b.py pract1cp p.py
pract1cp > ...
pract1cp > ...
1 #defining the list/chain
2 blockchain = []
3 print("7.govindsaini \n")
4 #getting the 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_transaction(sender, recipient, amount=1.0):
10     transaction = {'sender': sender,
11                    'recipient': recipient,
12                    'amount': amount}
13     blockchain.append(transaction)

14 #getting the details of transaction by entering to the prompt
15 def get_transaction_value():
16     tx_sender = input('Enter the sender: ')
17     tx_recipient = input('Enter the recipient of the transaction: ')
18     tx_amount = float(input('Enter your transaction amount: '))
19     return tx_sender, tx_recipient, tx_amount
20
21 #printing the blockchain
22 def print_block():
23     for block in blockchain:
24         print("GovindSaini \n")
25         print("Here is your block")
26         print(block)

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN
$ 
Ln 3, Col 12 Spaces:3 UTF-8 CRLF Python 3.9.7 (.venv:venv) ⚡ Prettier 🖼 19:44
28°C Haze ⚡ 90% 19-03-2022 22:46
```

## Output :

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

- File Explorer:** Shows a folder named "BLOCKCHAIN" containing files like "pract1.py", "pract1.py", "pract1f.py", "pract1.py", "pract1b.py", and "pract1c.py".
- Terminal:** Displays the command line output of the program. It starts with the code defining a blockchain list and printing the first transaction. Then it enters a loop where users can add more blocks by entering sender, recipient, and amount. The terminal shows several transactions being added, such as one from Govind to cg for 12000, and another from Govind to Siddhi for 15000.
- Status Bar:** Shows the file path "admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN", the line number "Ln 3, Col 12", and the date/time "19-03-2022 22:48".

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

### Code :

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

- File Explorer:** Shows a folder named "BLOCKCHAIN" containing files like "pract1d.py", "pract1.py", "pract1f.py", "pract1.py", "pract1b.py", and "pract1c.py".
- Terminal:** Displays the command line output of the program. It starts with the code defining a genesis block (an empty dictionary) and initializing a blockchain list with this block. It then defines functions to add transactions (adding them to a list of open transactions) and get the last value (returning the last block). Finally, it gets user input for a transaction and adds it to the blockchain.
- Status Bar:** Shows the file path "admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN", the line number "Ln 1, Col 1", and the date/time "18-03-2022 19:51".

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

- File Explorer:** On the left, it lists files and folders related to a "BLOCKCHAIN" project, including `.venv`, `SolidityProject`, and several `.py` files like `genesis.py`, `pract1.py`, `pract1b.py`, `pract1c.py`, and `pract1d.py`.
- Code Editor:** The main area displays the content of `pract1d.py`. The code implements a blockchain system with functions for adding transactions, getting transaction values, getting user choice, and printing blocks.
- Terminal:** At the bottom, the terminal window shows the command `admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN` and the status `$ [ ]`.
- Status Bar:** The bottom bar includes icons for file operations, a search bar, and status indicators for bash, terminal, output, and debug console.

## **Output :**

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

- File Explorer:** Shows a project structure under "BLOCKCHAIN" containing files like .venv, SolicityProject, and several .py files (pract1d.py, pract1e.py, pract1f.py, pract1g.py, pract1h.py).
- Code Editor:** Displays the content of pract1d.py, which defines functions for adding transactions to a blockchain.
- Terminal:** Shows command-line output for running pract1d.py, creating a genesis block, and displaying the first block's details.
- Status Bar:** Shows file paths, line numbers, and system information (Windows 10, 35°C, 19:51, 18-03-2022).

**1e ) Create a mining function and test it.**

## **Code :**

A screenshot of Visual Studio Code showing a Python file named `practie.py`. The code implements a blockchain mining algorithm. It defines a function `mine` that takes `block_number`, `transactions`, `previous_hash`, and `prefix_zeros` as parameters. It uses a loop to find a nonce that results in a SHA256 hash starting with the specified prefix zeros. If found, it prints a success message and returns the new hash. If not found after `MAX_NONCE` attempts, it raises an exception. The script then tries to mine a block with transactions from `Govind`, `Sidduh1`, and `Saroj`. It imports `time` and sets the difficulty to 4. The code is run in a terminal window.

```
from hashlib import sha256
MAX_NONCE = 100000000000
print("7_GovindSaini \n")
def SHA256(text):
    return sha256(text.encode("ascii")).hexdigest()

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

if __name__ == '__main__':
    transactions = ''
    Govind->Sidduh1->20,
    Saroj->Pooja->45
    ...
    # try changing this to higher number and you will see it will take more time for mining as difficulty increases
    import time
    difficulty = 4
    start = time.time()
    print("start mining")
    new_hash = mine(
        5, transactions, '0000000xa03d94c20568d0cff17edbe038f81208fecf9a66be9a2b8321c6ec7', difficulty)
    total_time = str(time.time() - start)
    print(f"end mining. Mining took: {total_time} seconds")
    print(f"the new hash value : {new_hash}
```

A screenshot of Visual Studio Code showing the same `practie.py` file with modifications. The `mine` function now includes a `difficulty` parameter and a `start` variable to track the start time of the mining process. The `print("start mining")` and `print("end mining. Mining took: " + str(total_time) + " seconds")` statements have been added. The code is run in a terminal window.

```
print(f"Yay! Successfully mined bitcoins with nonce value:{nonce}")
return new_hash
raise BaseException(f"Couldn't find correct has after trying {MAX_NONCE} times")
if __name__ == '__main__':
    transactions = ''
    Govind->Sidduh1->20,
    Saroj->Pooja->45
    ...
    # try changing this to higher number and you will see it will take more time for mining as difficulty increases
    import time
    difficulty = 4
    start = time.time()
    print("start mining")
    new_hash = mine(
        5, transactions, '0000000xa03d94c20568d0cff17edbe038f81208fecf9a66be9a2b8321c6ec7', difficulty)
    total_time = str(time.time() - start)
    print(f"end mining. Mining took: {total_time} seconds")
    print(f"the new hash value : {new_hash}")
```

## Output :

A screenshot of Visual Studio Code showing the terminal output of the mining process. The terminal shows the execution of `$ py practie.py`, the initial state `7_GovindSaini`, the start of mining, the successful mining of a block with nonce 27405, and the final printed hash value.

```
admin@DESKTOP-3B61180 MINGw64 /c/govindworking/SEM 4 MSC IT/BLOCKCHAIN
$ py practie.py
7_GovindSaini
start mining
Yay! Successfully mined bitcoins with nonce value:27405
end mining. Mining took: 0.2350523086816406 seconds
the new hash value : 0000605c84f941938f28a1c1c0050a71dbfb692a22eadcaa0c2ff6af4dd9027e7
```

### 1f ) Add blocks to the miner and dump the blockchain Code :

The screenshot shows two instances of Visual Studio Code running side-by-side on a Windows desktop. Both instances have the title "practif.py - BLOCKCHAIN - Visual Studio Code".

**Top Window:**

```
practif.py > ...
1 import datetime
2 import hashlib
3
4 class Block:
5     blockNo = 0
6     data = None
7     next = None
8     hash = None
9     nonce = 0
10    previous_hash = 0x0
11    timestamp = datetime.datetime.now()
12
13    def __init__(self, data):
14        self.data = data
15
16    def hash(self):
17        h = hashlib.sha256()
18        h.update(
19            str(self.nonce).encode('utf-8') +
20            str(self.data).encode('utf-8') +
21            str(self.previous_hash).encode('utf-8') +
22            str(self.timestamp).encode('utf-8') +
23            str(self.blockNo).encode('utf-8')
24        )
25        return h.hexdigest()
26
27    def __str__(self):
28        return "Block Hash: " + str(self.hash()) + "\nBlockNo: " + str(self.blockNo) + "\nBlock Data: " + str(self.data)
```

**Bottom Window:**

```
practif.py > ...
27    def __str__(self):
28        return "Block Hash: " + str(self.hash()) + "\nBlockNo: " + str(self.blockNo) + "\nBlock Data: " + str(self.data)
29
30    class Blockchain:
31
32        diff = 20
33        maxNonce = 2**32
34        target = 2 ** (256-diff)
35
36        block = Block("Genesis")
37        dummy = head = block
38
39        def add(self, block):
40
41            block.previous_hash = self.block.hash()
42            block.blockNo = self.block.blockNo + 1
43
44            self.block.next = block
45            self.block = self.block.next
46
47        def mine(self, block):
48            for n in range(self.maxNonce):
49                if int(block.hash(), 16) <= self.target:
50                    self.add(block)
51                    print(block)
52                    break
53                else:
```

## **Output :**

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

- File Explorer:** Shows a project structure under "BLOCKCHAIN".
  - pract1.py
  - pract1e.py
  - pract1f.py
  - pract1f.py (3)
  - pract1f.py (4)
  - pract1f.py (5)
  - pract1f.py (6)
  - pract1f.py (7)
  - pract1f.py (8)
  - pract1f.py (9)
  - pract1f.py (10)
  - pract1f.py (11)
  - pract1f.py (12)
  - pract1f.py (13)
  - pract1f.py (14)
  - pract1f.py (15)
  - pract1f.py (16)
  - pract1f.py (17)
  - pract1f.py (18)
  - pract1f.py (19)
  - pract1f.py (20)
  - pract1f.py (21)
  - pract1f.py (22)
  - pract1f.py (23)
  - pract1f.py (24)
  - pract1f.py (25)
  - pract1f.py (26)
  - pract1f.py (27)
  - pract1f.py (28)
  - pract1f.py (29)
  - pract1f.py (30)
  - pract1f.py (31)
  - pract1f.py (32)
  - pract1f.py (33)
  - pract1f.py (34)
  - pract1f.py (35)
  - pract1f.py (36)
  - pract1f.py (37)
  - pract1f.py (38)
  - pract1f.py (39)
  - pract1f.py (40)
  - pract1f.py (41)
  - pract1f.py (42)
  - pract1f.py (43)
  - pract1f.py (44)
  - pract1f.py (45)
  - pract1f.py (46)
  - pract1f.py (47)
  - pract1f.py (48)
  - pract1f.py (49)
  - pract1f.py (50)
  - pract1f.py (51)
  - pract1f.py (52)
  - pract1f.py (53)
  - pract1f.py (54)
  - pract1f.py (55)
  - pract1f.py (56)
  - pract1f.py (57)
  - pract1f.py (58)
  - pract1f.py (59)
  - pract1f.py (60)
  - pract1f.py (61)
  - pract1f.py (62)
  - pract1f.py (63)
  - pract1f.py (64)
  - pract1f.py (65)
  - pract1f.py (66)
  - pract1f.py (67)
  - pract1f.py (68)
  - pract1f.py (69)
  - pract1f.py (70)
  - pract1f.py (71)
  - pract1f.py (72)
  - pract1f.py (73)
  - pract1f.py (74)
  - pract1f.py (75)
  - pract1f.py (76)
  - pract1f.py (77)
  - pract1f.py (78)
  - pract1f.py (79)
  - pract1f.py (80)
  - pract1f.py (81)
  - pract1f.py (82)
  - pract1f.py (83)
  - pract1f.py (84)
  - pract1f.py (85)
  - pract1f.py (86)
  - pract1f.py (87)
  - pract1f.py (88)
  - pract1f.py (89)
  - pract1f.py (90)
  - pract1f.py (91)
  - pract1f.py (92)
  - pract1f.py (93)
  - pract1f.py (94)
  - pract1f.py (95)
  - pract1f.py (96)
  - pract1f.py (97)
  - pract1f.py (98)
  - pract1f.py (99)
  - pract1f.py (100)
- Terminal:** Displays the command "pract1f.py - BLOCKCHAIN - Visual Studio Code" and the output of the blockchain's hash calculation process.
- Status Bar:** Shows file paths like "C:\Users\admine\PycharmProjects\BLOCKCHAIN\pract1f.py", line numbers (e.g., 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100), and the date "18-03-2022".

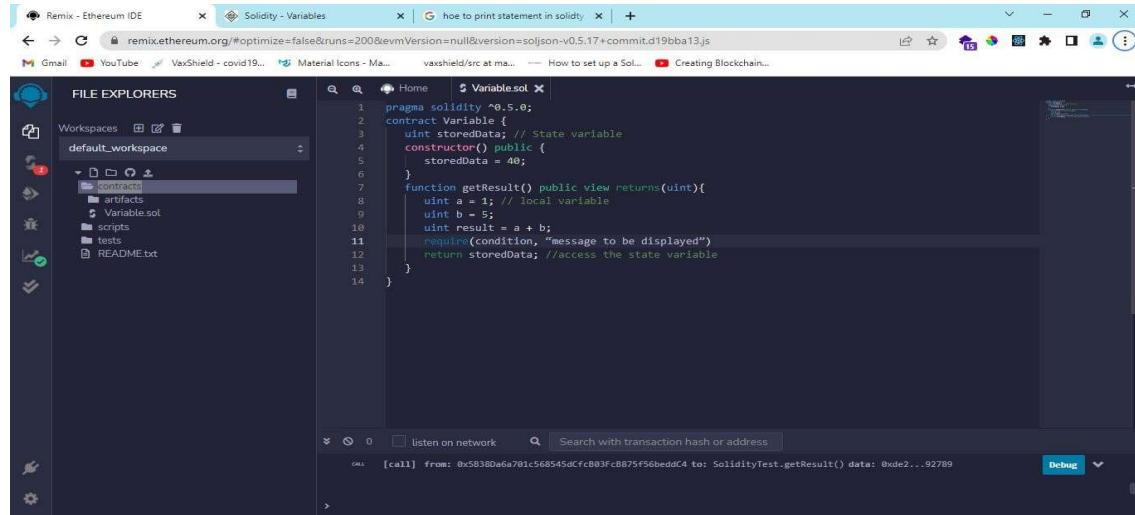
The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a tree view of a "BLOCKCHAIN" folder containing subfolders like ".venv", "SolidityProject", and several "govindSaini" files.
- Code Editor:** Displays multiple tabs of Python code, including "pract1.py", "pract1f.py", and "pract1f.py".
- Terminal:** Shows command-line output for "pract1f.py" running on a Windows machine (MINGW64). The output includes block hashes, block numbers, and block data for various blocks.
- Status Bar:** Shows file paths, line numbers (Ln 9, Col 16), and other system information.

# Practical No : 3

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

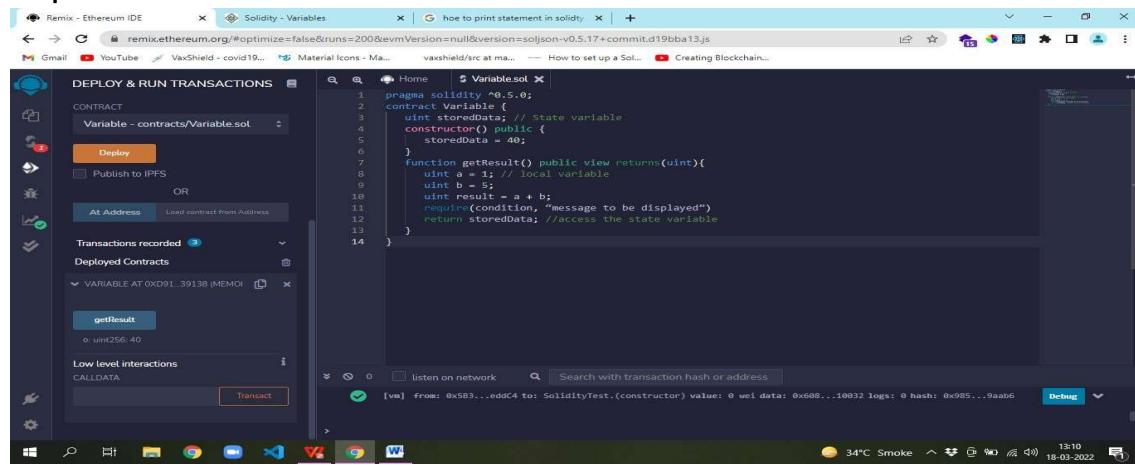
3a ) Variable, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs, Mappings, Conversions, Ether Units, Special Variables. Variable Code :



The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file explorer with a workspace named 'default\_workspace' containing contracts, artifacts, scripts, tests, and a README.txt file. The main central area shows the Solidity code for a contract named 'Variable.sol'. The code defines a state variable 'storedData' and a constructor that initializes it to 40. It includes a function 'getResult()' that adds two local variables 'a' and 'b' (both initialized to 5) and returns the result. A requirement is placed on the condition that the message to be displayed is true. The bottom status bar shows a transaction hash and a debug button.

```
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 :**



The screenshot shows the Remix Ethereum IDE interface with the 'DEPLOY & RUN TRANSACTIONS' tab selected. It displays the deployment of the 'Variable' contract from the 'Variable - contracts/Variable.sol' file. The contract has been deployed at address 0x0D91...39138. The 'Transactions recorded' section shows a call to the 'getResult' function, which returns the value 40. The 'Low level interactions' section shows a call to the 'CALLDATA' function with the same result. The bottom status bar shows a transaction hash and a debug button.

```
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
    }
}
```

Operators : Arithmetic

**Operator Code:**

**Remix - Ethereum IDE**

**Solidity - Operators - GeeksforGeeks**

File Explorers

Workspaces default\_workspace

ArithmeticOperators.sol

```
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;
}
```

ContractDefinition ArithmeticOperators → 1 reference(s) ▾

Search with transaction hash or address

32°C Smoke 20:58 18-03-2022

## Output:

**Remix - Ethereum IDE**

**Solidity - Operators - GeeksforGeeks**

Debugger

Function Stack

Solidity Locals

sum: 20 uint256

Solidity State

a: 51 uint16  
b: 19 uint16  
sum: 70 uint256  
diff: 30 uint256  
mul: 1000 uint256  
div: 2 uint256  
mod: 10 uint256  
dec: 19 uint256  
inc: 51 uint256

346 JUMPDEST  
349 PUSH1 0164  
352 PUSH1 01f5  
355 JUMP  
356 JUMPDEST  
357 PUSH1 40

```
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;
}
```

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

[vm] from: 0x583...eddC4 to: ArithmeticOperators.(constructor) value: 0 wei data: 0x608...10032 logs: 0  
hash: 0xd6f...3db68  
call to ArithmeticOperators.dec

[call] from: 0x5830a6a701c568545dCfcB03FcB875f56beddC4 to: ArithmeticOperators.dec() data: 0xb3b...cf82  
call to ArithmeticOperators.diff

32°C Smoke 20:59 18-03-2022

**Remix - Ethereum IDE**

**Solidity - Operators - GeeksforGeeks**

Deploy & Run Transactions

a  
o: uint16: 51

b  
o: uint16: 19

dec  
o: uint256: 19

diff  
o: uint256: 30

div  
o: uint256: 2

inc  
o: uint256: 51

mod  
o: uint256: 10

mul  
o: uint256: 1000

sum  
o: uint256: 70

```
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;
}
```

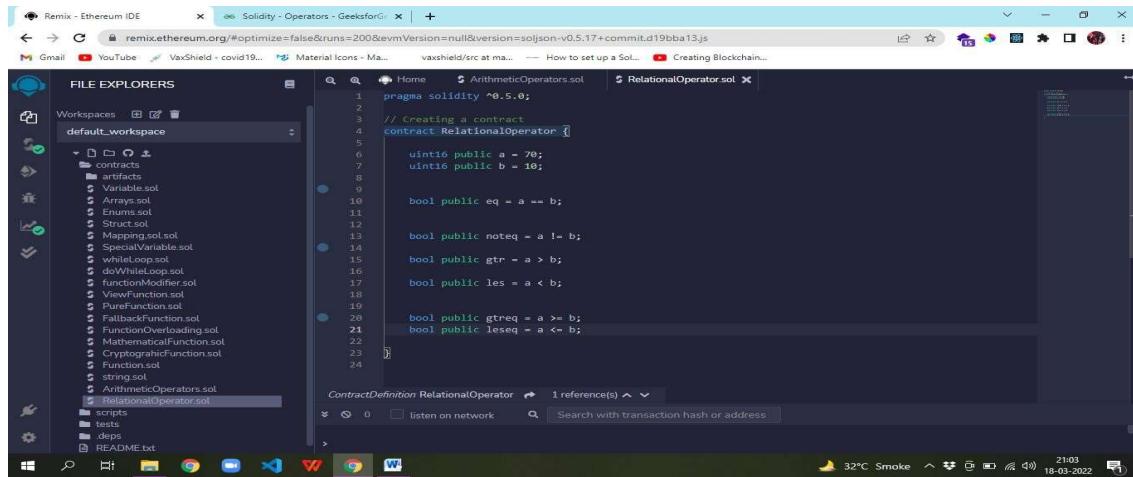
Type the library name to see available commands.  
creation of ArithmeticOperators pending...

[vm] from: 0x583...eddC4 to: ArithmeticOperators.(constructor) value: 0 wei data: 0x608...10032 logs: 0  
hash: 0xd6f...3db68  
call to ArithmeticOperators.dec

[call] from: 0x5830a6a701c568545dCfcB03FcB875f56beddC4 to: ArithmeticOperators.dec() data: 0xb3b...cf82  
call to ArithmeticOperators.diff

32°C Smoke 20:59 18-03-2022

## Relational Operator Code:



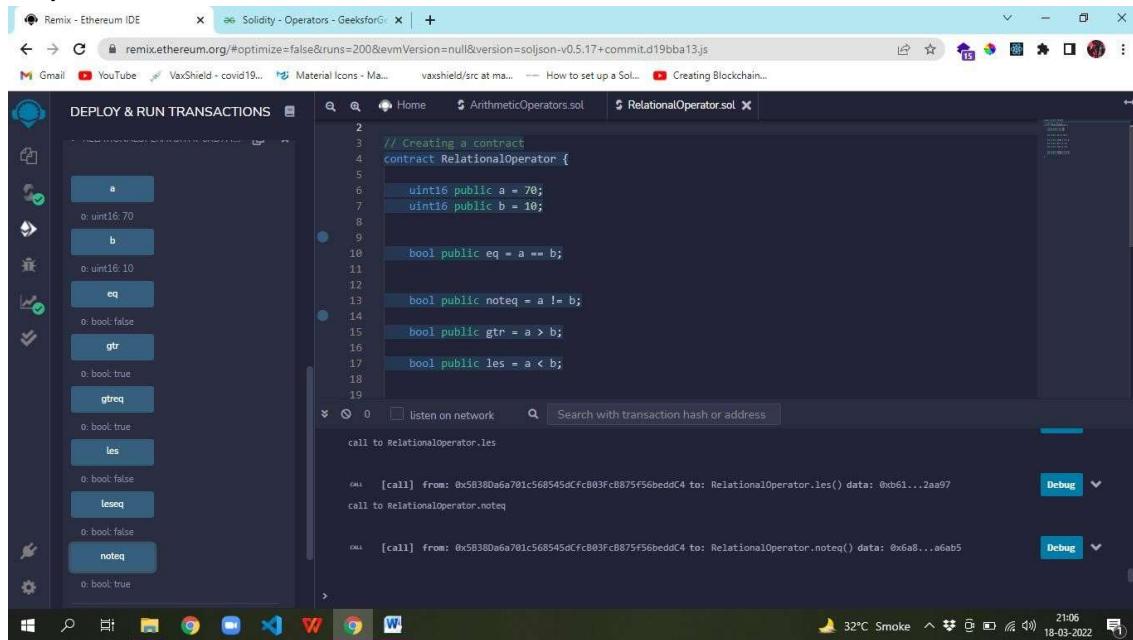
The screenshot shows the Remix Ethereum IDE interface. The left sidebar displays a file tree under 'FILE EXPLORERS' for a workspace named 'default\_workspace'. The tree includes contracts like 'ArithmeticOperators.sol' and 'RelationalOperator.sol', as well as artifacts, scripts, tests, and deps. The 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 shows system information: 32°C Smoke, 21:03, 18-03-2022.

## Output :



The screenshot shows the Remix Ethereum IDE interface with the 'DEPLOY & RUN TRANSACTIONS' sidebar open. This sidebar lists the variables and functions defined in the contract: 'a' (uint16: 70), 'b' (uint16: 10), 'eq' (bool: false), 'gtr' (bool: true), 'gtreq' (bool: true), 'les' (bool: false), 'leseq' (bool: false), and 'noteq' (bool: true). The main editor window shows the same Solidity code as the previous screenshot. The status bar at the bottom shows system information: 32°C Smoke, 21:06, 18-03-2022.

The screenshot shows the Remix Ethereum IDE interface. The left sidebar contains a 'FILE EXPLORERS' section with a 'default\_workspace' folder containing several Solidity files like ArithmeticOperators.sol, RelationalOperator.sol, and LogicalOperator.sol. The main workspace displays the 'RelationalOperator.sol' file with 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 a message: "This call has reverted. state changes made during the call will be reverted. Click here to jump where the call reverted." Below this, it lists variables: a: 70 uint16, b: 10 uint16, eq: false bool, noteq: true bool, gtr: true bool, les: false bool, gtreq: true bool, lessq: false bool. The bottom of the debugger panel shows assembly code: 141 JUMPDEST, 142 PUSH2 0095, 143 PUSH2 0145, 148 JMP, 149 JUMPDEST.

The status bar at the bottom right indicates: 32°C Smoke 21:04 18-03-2022

### Logical Operator Code :

The screenshot shows the Remix Ethereum IDE interface. The left sidebar contains a 'FILE EXPLORERS' section with a 'default\_workspace' folder containing several Solidity files like ArithmeticOperators.sol, RelationalOperator.sol, and LogicalOperator.sol. The main workspace displays the 'LogicalOperator.sol' file with 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 status bar at the bottom right indicates: 32°C Smoke 21:07 18-03-2022

### Output:

The screenshot shows the Remix Ethereum IDE interface. On the left, the sidebar includes sections for 'DEPLOY & RUN TRANSACTIONS' (with a 'Deploy' button), 'Transactions recorded' (listing 'LOGICALOPERATOR AT 0x353...', 'Deployed Contracts'), and 'Low level interactions'. The main code editor displays the Solidity code for 'LogicalOperator.sol':

```
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;
    }
}
```

Below the code, the 'Logic' section shows a call to the 'Logic' function with inputs 'true' and 'false', resulting in outputs: 0: bool: false, 1: bool: true, 2: bool: false. The bottom status bar shows the date and time: 21:10 18-03-2022.

### Bitwise Operator Code:

The screenshot shows the Remix Ethereum IDE interface. On the left, the sidebar includes sections for 'FILE EXPLORERS' (showing 'default\_workspace' with files like 'Variable.sol', 'Arrays.sol', etc.) and 'Deploy & Run Transactions' (with a 'Deploy' button). The main code editor displays the Solidity code for 'BitwiseOperator.sol':

```
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 bottom status bar shows the date and time: 21:13 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, not, leftshift, rightshift, and xor. The main workspace displays the Solidity code for the `BitwiseOperator` contract:

```

1 pragma solidity ^0.5.0;
2
3 // Creating a contract
4 contract BitwiseOperator {
5
6     // Declaring variables
7     uint16 public a = 28;
8     uint16 public b = 58;
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}

```

Below the code, the "ContractDefinition BitwiseOperator" section shows "1 reference(s)". The deployment logs indicate a successful constructor call and a revert error for the `leftshift` function:

- [vm] from: 0x5b3...edd4 to: logicalOperator.(constructor) value: 0 wei data: 0x600...10032 logs: 0 hash: 0x428...489c Debug
- [call] from: 0x5b380ada701c568545dCfcB83FcB875f56bedd4 to: logicalOperator.Logic(bool,bool) data: 0xc94...00000 Debug
- call to logicalOperator.b
- call to logicalOperator.b errored: VM error: revert.

This screenshot shows the same Remix IDE setup but with the "DEBUGGER" tab selected. The sidebar now includes "Function Stack" and "Solidity Locals". The "Solidity State" section shows the state of variables: `a: 28 uint16`, `b: 58 uint16`, `and: 16 uint16`, `or: 34 uint16`, `xor: 38 uint16`, `leftshift: 0 uint16`, `rightshift: 0 uint16`, and `not: 65515 int16`. The deployment logs are identical to the previous screenshot.

### Assignment Operator Code :

The screenshot shows the Remix IDE with the "FILE EXPLORERS" tab selected. The "Workspaces" section shows the "default\_workspace" with a "contracts" folder containing several files like `Variable.sol`, `Arrays.sol`, `Enums.sol`, etc. The main workspace displays the Solidity code for the `ArithmeticOperators` contract:

```

1 pragma solidity ^0.5.0;
2
3 contract ArithmeticOperators {
4
5     uint16 public a = 58;
6     uint16 public b = 28;
7
8     uint public sum = a + b;
9     uint public diff = a - b;
10    uint public mul = a * b;
11
12    uint public div = a / b;
13    uint public mod = a % b;
14
15    uint public dec = --b;
16    uint public inc = ++a;
17
18 }

```

Below the code, the "ContractDefinition ArithmeticOperators" section shows "1 reference(s)". The deployment logs are as follows:

- [vm] from: 0x5b3...edd4 to: ArithmeticOperators.(constructor) value: 0 wei data: 0x600...10032 logs: 0 hash: 0x602...24e3 Debug
- creation of ArithmeticOperators pending..
- [vm] from: 0x5b3...edd4 to: ArithmeticOperators.(constructor) value: 0 wei data: 0x600...10032 logs: 0 hash: 0x07c...dfc8e Debug
- call to ArithmeticOperators.a

## Ouput :

The screenshot shows the Remix Ethereum IDE interface. At the top, there are two tabs: "Solidity - Operators - GeeksforGeeks" and "Remix - Ethereum IDE". The main area displays the Solidity code for the `AssignmentOperator` contract. The code defines variables for assignment, addition, multiplication, division, and modulus operations, along with a `getResult` function. Below the code editor is a sidebar titled "Low level interactions" containing various operators like `assign_div`, `assign_mod`, etc. At the bottom, there's a "CALLDATA" section and a browser-like interface showing the transaction hash and status.

Below this, another instance of the Remix IDE is shown, specifically the "DEBUGGER" tab. It displays the state of the deployed contract at address `0xc0044e0ff011c3a62a7392d0ea3f7...`. The debugger interface includes a "Function Stack" showing the `getResult` call, and a "Solidity State" table listing variables like `assignment`, `assignment_add`, `assignment_sub`, `assignment_mul`, `assignment_div`, and `assignment_mod`.

## Loops whileLoop code:

This screenshot shows the Remix Ethereum IDE again, this time with the `whileLoop` contract. The code defines a dynamic array `data` and a state variable `j`. The `loop` function uses a `while` loop to push values into `data` until `j > 10`. The deployment interface at the bottom shows the transaction hash `0xa80...267c8` and the status "pending".

## Output

The screenshot shows the Remix IDE interface with the Solidity State debugger open. The code editor displays a Solidity contract named `whileLoop`:

```
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 < 10) {
            j++;
            data.push(j);
        }
        return data;
    }
}
```

The Solidity State pane shows the state of the `data` array, which has a length of 16 and contains values from 0 to 9. The transaction pane shows two successful transactions:

- From: 0x583...edd4 to: whileLoop.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x432...67d58
- From: 0x583...edd4 to: whileLoop.loop() value: 0 wei data: 0xe92...100cb logs: 0 hash: 0x54f...07ec0

At the bottom, the status bar indicates "31°C Smoke" and the date "18-03-2022".

The screenshot shows the Remix IDE interface with the Solidity State debugger open. The code editor displays a Solidity contract named `doWhileLoop`:

```
pragma solidity ^0.5.0;
// Creating a contract
contract doWhileLoop {
    // Declaring a dynamic array
    uint[] data;
    // Declaring state variable
    uint8 j = 0;

    function loop()
        public returns(uint[] memory){
        do{
            j++;
            data.push(j);
        }while(j < 8);
        return data;
    }
}
```

The Solidity State pane shows the state of the `data` array, which has a length of 5 and contains values from 0 to 4. The transaction pane shows two successful transactions:

- From: 0x583...edd4 to: whileLoop.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x10c...5dd46
- From: 0x583...edd4 to: whileLoop.loop() value: 0 wei data: 0xe92...100cb logs: 0 hash: 0xe02...eaf3c

At the bottom, the status bar indicates "36°C Smoke" and the date "18-03-2022".

## Dowhile loop Code:

The screenshot shows the Remix IDE interface with the Solidity State debugger open. The code editor displays a Solidity contract named `doWhileLoop`:

```
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;
    }
}
```

The Solidity State pane shows the state of the `data` array, which has a length of 5 and contains values from 0 to 4. The transaction pane shows two successful transactions:

- From: 0x583...edd4 to: whileLoop.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x10c...5dd46
- From: 0x583...edd4 to: whileLoop.loop() value: 0 wei data: 0xe92...100cb logs: 0 hash: 0xe02...eaf3c

At the bottom, the status bar indicates "31°C Smoke" and the date "18-03-2022".

## Output :

The screenshot shows the Remix Ethereum IDE interface. The left sidebar contains a debugger with a function stack showing '0-loop()' and solidity locals. The main code editor displays 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++;
        } while(j < 8);
        return data;
    }
}
```

The bottom status bar shows the date and time as 18-03-2022 21:42.

## Forloop Code :

The screenshot shows the Remix Ethereum IDE interface. The left sidebar contains a file explorer with a workspace named 'default\_workspace' containing various Solidity files like AssignmentOperator.sol, whiteLoop.sol, doWhileLoop.sol, and forLoop.sol. The main code editor displays the following Solidity 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;
    }
}
```

The bottom status bar shows the date and time as 18-03-2022 21:43.

## Output :

The screenshot shows the Remix Ethereum IDE interface. The left sidebar contains a 'DEBUGGER' section with tabs for 'Function Stack' (showing a stack trace for 'o: loop') and 'Solidity Locals' (showing local variables). Below these are sections for 'Solidity State' (showing memory dump data) and 'Step details' (showing VM trace steps). The main code editor area displays the following Solidity 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;
    }
}
```

The status bar at the bottom indicates a VM trace step of 519 and a temperature of 31°C.

## Decision making

### If statement Code:

The screenshot shows the Remix Ethereum IDE interface. The left sidebar contains a 'FILE EXPLORERS' section listing various Solidity files. The main code editor area displays the following Solidity 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;
        }
    }
}
```

The status bar at the bottom indicates a VM trace step of 519 and a temperature of 30°C.

### Code :

The screenshot shows the Remix Ethereum IDE interface. The top bar has tabs for "Solidity - Operators - GeeksforGeeks" and "Remix - Ethereum IDE". Below the tabs, there's a toolbar with icons for back, forward, search, and file operations. The main area is divided into several panes:

- DEBUGGER** pane on the left: Shows a transaction hash (0x17689840bb32e24d84f49d8a9a71...) and a "Stop debugging" button.
- Function Stack**: Shows a stack trace starting with "o.\_decision\_making()".
- Solidity Locals**: Shows variables i and val.
- Solidity State**: Shows memory dump details like JUMPDEST, PUSH1, and SLOAD.
- Code Editor**: Displays the Solidity code for the `ifStatement` contract, which includes a state variable `i` and a function `decision making`.
- Logs**: Shows the output of the function call, indicating a value of `false`.
- Bottom Status Bar**: Shows system information like temperature (30°C), battery level (Smoke), and date/time (18-03-2022 21:50).

### If...else statement Code :

The screenshot shows the Remix Ethereum IDE interface. The top bar has tabs for "Solidity - Operators - GeeksforGeeks" and "Remix - Ethereum IDE". Below the tabs, there's a toolbar with icons for back, forward, search, and file operations. The main area is divided into several panes:

- FILE EXPLORERS** pane on the left: Shows a file tree with various Solidity files like `Enums.sol`, `Struct.sol`, etc., and a `scripts` folder containing deployment scripts.
- Code Editor**: Displays the Solidity code for the `ifElseStatement` contract, which includes a state variable `i` and a function `decision making` that uses an `if...else` conditional statement.
- Bottom Status Bar**: Shows system information like temperature (30°C), battery level (Smoke), and date/time (18-03-2022 21:54).

### Output :

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

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

## If...else if...else statement

### Code :

```
pragma solidity ^0.5.0;
// Creating a contract
contract IfElseIfStatement {
    // 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:

```

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){
    
```

## String

### Code:

```

contract String {
    string[] public row;

    function getRow() public view returns (string[] memory) {
        return row;
    }

    function pushToRow(string memory newValue) public {
        row.push(newValue);
    }
}

```

### Output :

The screenshot shows the Remix Ethereum IDE interface. On the left, the sidebar has sections for 'DEPLOY & RUN TRANSACTIONS' and 'Deployed Contracts'. The main area displays the Solidity code for a contract named 'String'. The code defines a public array of strings and two functions: 'getRow' which returns the array, and 'pushToRow' which adds a new value to the array. Below the code, the 'Transactions recorded' section shows several pending transactions, including calls to the contract's methods like 'callSumWithT...' and 'getSum'. The status bar at the bottom indicates it's 36°C, Smoke, 19:34, and the date is 18-03-2022.

```
contract String {
    string[] public row;

    function getRow() public view returns (string[] memory) {
        return row;
    }

    function pushToRow(string memory newValue) public {
        row.push(newValue);
    }
}
```

## Array

### Code :

The screenshot shows the Remix Ethereum IDE interface. The 'FILE EXPLORERS' sidebar shows a workspace named 'default\_workspace' containing files like 'Variable.sol', 'Arrays.sol', and 'remix\_tests.sol'. The main area displays the Solidity code for a test contract named 'test'. The code includes a function 'testArray()' that initializes a dynamic array 'a' of length 7, asserts its length, and then asserts that its 6th element is 8. It also demonstrates static arrays by creating a memory array 'c' with values [1, 2, 3] and asserting its length. The status bar at the bottom indicates it's 36°C, Smoke, 16:49, and the date is 18-03-2022.

```
pragma solidity ^0.5.0;

contract test {
    function testArray() public pure{
        uint len = 7;
        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. The top bar has tabs for "Remix - Ethereum IDE" and "Solidity - Arrays". The main area displays the following Solidity code:

```
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);
    }
}
```

The left sidebar has sections for "DEPLOY & RUN TRANSACTIONS" (Deploy, Publish to IPFS, At Address), "Transactions recorded" (TEST AT 0xD91\_39138 (MEMORY)), and "Low level interactions" (CALLDATA). The bottom status bar shows system information: 36°C, Smoke, 16:51, 18-03-2022.

## Enums

### Code :

The screenshot shows the Remix Ethereum IDE interface. The top bar has tabs for "Remix - Ethereum IDE" and "Solidity - Enums". The main area 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 has sections for "FILE EXPLORERS" (Workspaces, default\_workspace, contracts, scripts, tests, deps, remix-tests, remix\_tests.sol, remix\_accounts.sol, README.txt) and "Low level interactions" (CALLDATA). The bottom status bar shows system information: 36°C, Smoke, 16:58, 18-03-2022.

### 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) {
        return uint(FreshJuiceSize.defaultChoice);
    }
}

```

## Struct

### Code:

```

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 Remix Ethereum IDE interface. The top navigation bar has tabs for "Remix - Ethereum IDE", "Solidity - Structs", "Inbox (803) - govind.rainin666@...", and "WhatsApp". Below the tabs, there's a toolbar with icons for back, forward, search, and file operations. The main workspace is divided into several panes:

- Deploy & Run Transactions**: Shows deployed contracts and recorded transactions.
- Code Editor**: Displays the following Solidity code:
 

```

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) {
    }
  
```
- Transactions Recorded**: Lists recorded transactions like "setBook" and "getBookId".
- Low Level Interactions**: Shows CALLDATA and TRANACT buttons.
- Output**: Shows transaction logs and pending calls.

The status bar at the bottom shows system information: 36°C, Smoke, 17:20, 18-03-2022.

### Mapping Code:

The screenshot shows the Remix Ethereum IDE interface with a different tab selected: "Solidity - Mappings - GeeksforGeeks". The layout is similar to the previous screenshot, with the same top navigation bar and toolbar.

The main workspace shows the following Solidity code for a "Mapping" contract:

```

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(
            0xDEE7796E89C82C36BAdd1375076f39D69FafE252) -1;
    }
}
  
```

The status bar at the bottom shows system information: 37°C, Smoke, 18:23, 18-03-2022.

### Output :

```

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 (address => student) result;
15    address[] public student_result;
16
17    // Function adding values to the mapping
18    function adding_values() public {
19        student memory s;
20        s.name = "Govind";
21        s.subject = "Solidity";
22        s.marks = 90;
23        result[tx.origin] = s;
24    }
25
26    // Function getting values from the mapping
27    function get_marks(address student_address) public view returns (uint8) {
28        return result[student_address].marks;
29    }
30
31    // Function getting all students
32    function get_all_students() public view returns (address[]) {
33        return student_result;
34    }
35}

```

## Special Variable Code:

```

2 pragma solidity ^0.6.6;
3
4 // Creating a smart contract
5 contract SpecialVariable {
6     // Creating a mapping
7     mapping (address => uint) rollNo;
8
9     function setRollNO(uint _myNumber) public {
10
11         rollNo[msg.sender] = _myNumber;
12     }
13
14     // Defining a function to
15     // return the roll no.
16     function whatIsMyRollNumber() public view returns (uint) {
17
18         return rollNo[msg.sender];
19     }
20
21 }
22
23
24

```

## Output

```
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.
}
```

Transactions recorded: 22

Deployed Contracts: SPECIALVARIABLE AT 0x38C...24C73

Low level interactions: CALLDATA

ContractDefinition SpecialVariable → 1 reference(s) ▾

0 listen on network Search with transaction hash or address

[call] from: 0x58380a6a701c568545dcfc803fc8875f56bedd4 to: SpecialVariable.whatIsMyRollNumber() data: 0x3ee...b5710 Debug

transact to SpecialVariable.setRollNo pending ...

[vm] from: 0x58380a6a701c568545dcfc803fc8875f56bedd4 to: SpecialVariable.setRollNo(uint256) 0xC3B...aaECA value: 0 wei data: 0x15e...00007 logs: 0 Debug

hash: 0xb6a...24737

call to SpecialVariable.whatIsMyRollNumber

[call] from: 0x58380a6a701c568545dcfc803fc8875f56bedd4 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 = 28;
        product = a * b;
        sum = a + b;
    }
}
```

## Output:

```
pragma solidity ^0.5.0;

contract Function {
    function getResult() public view returns(uint product, uint sum) {
        uint a = 11; // local variable
        uint b = 28;
        product = a * b;
        sum = a + b;
    }
}
```

## Functions Modifiers 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 :

```
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; }
    }
}
```

## View function Code:

The screenshot shows the Remix IDE interface. The left sidebar displays the file explorer with contracts like Artifacts, Variable.sol, Arrays.sol, Enums.sol, Struct.sol, Mapping.sol.sol, SpecialVariable.sol, whileLoop.sol, doWhileLoop.sol, Operators.sol, functionModifier.sol, and ViewFunction.sol. The main editor window 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;
    }
}
```

The status bar at the bottom indicates a network connection, a temperature of 36°C, and a timestamp of 19:01 on 18-03-2022.

## **Output :**

The screenshot shows the Remix IDE interface with the "Deploy & Run Transactions" tab selected. The left sidebar shows deployed contracts: VIEWFUNCTION AT 0X7EF...BCB47 and VIEWFUNCTION AT 0XDA0...42B53. The main editor window contains the same Solidity code as above. The status bar at the bottom indicates a network connection, a temperature of 36°C, and a timestamp of 19:03 on 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 two 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 }

```

The screenshot shows the Remix IDE interface with multiple tabs open. The left sidebar shows a file explorer with a workspace named 'default\_workspace' containing several Solidity files like Operators.sol, functionModifier.sol, ViewFunction.sol, and PureFunction.sol. The main editor area displays the Solidity code for PureFunction.sol. Below the editor, a terminal window shows a transaction log:

```

[call] from: 0x5B38D0a6a701c568545dCfcB03FcB875f56beddC4 to: ViewFunction.getResult() data: 0xde2...92789

```

## Output:

The screenshot shows the Remix IDE interface with the 'Deploy & Run Transactions' tab selected. On the left, there's a sidebar for deploying contracts with fields for 'VALUE' (0 Wei), 'CONTRACT' (PureFunction), and buttons for 'Deploy' and 'Publish to IPFS'. Below this, a list shows 'Transactions recorded' and 'Deployed Contracts' with one entry: 'PUREFUNCTION AT 0xD2A...FD005'. Under 'Low level interactions', there's a 'CALLDATA' section. The main editor area shows the same Solidity code as before. The terminal window shows the deployment and execution of the contract:

```

[call] from: 0x5B38D0a6a701c568545dCfcB03FcB875f56beddC4 to: ViewFunction.getResult() data: 0xde2...92789
creation of PureFunction pending...
[call] from: 0x5B38D0a6a701c568545dCfcB03FcB875f56beddC4 to: ViewFunction.getResult() data: 0xde2...92789

```

## Fallback function

```

1 pragma solidity ^0.5.0;
2
3 contract FallbackFunction {
4     uint public x;
5     function() external { x = 1; }
6 }
7 contract Sink {
8     function() external payable {}
9 }
10
11 function callTest(Test test) public returns (bool) {
12     (bool success,) = address(test).call(gbi.encodeWithSignature("nonExistingFunction()"));
13     require(success);
14     // test.x is now 1
15     address payable testPayable = address(uint160(address(test)));
16     // sending ether to Test contract
17     // this transaction will fail because this returns false here.
18     // (testPayable.send(2 ether));
19     return (testPayable.send(2 ether));
20 }
21
22 function callsink(Sink sink) public returns (bool) {
23     address payable sinkPayable = address(sink);
24     return (sinkPayable.send(2 ether));
25 }

```

## Output :

DEPLOY & RUN TRANSACTIONS

Caller - contracts/FallbackFunction.sol

Deploy

PUREFUNCTION AT 0x02A...F0005 (0)

CALLER AT ADDRESS 0x02A...5482D (MEMORY)

callSink address sink

callTest address test

Low level interactions CALLDATA

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...

## Function Overloading

```

1 pragma solidity ^0.5.0;
2
3 contract FunctionOverloading {
4     function getSum(uint a, uint b) public pure returns(uint){
5         return a + b;
6     }
7     function getSum(uint a, uint b, uint c) public pure returns(uint){
8         return a + b + c;
9     }
10    function callSumWithTwoArguments() public pure returns(uint){
11        return getSum(1,2);
12    }
13    function callSumWithThreeArguments() public pure returns(uint){
14        return getSum(1,2,3);
15    }
16 }

```

## 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);
    }
}

```

**Deploy & Run Transactions**

- Deploy
- Publish to IPFS
- OR
- At Address
- Load contract from Address

**Transactions recorded**

**Deployed Contracts**

**FUNCTIONOVERLOADING AT 0xD91**

- callSumWithT...
- D: uint256 6
- callSumWithT...
- D: uint256 3
- getSum
- uint256 a, uint256 b
- getSum
- uint256 a, uint256 b, uint2

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

[vm] from: 0x5B3...eddC4 to: FunctionOverloading.(constructor) value: 0 wei data: 0x600...10032 logs: 0  
hash: 0x9c4...f2a2a  
call to FunctionOverloading.callSumWithThreeArguments

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: FunctionOverloading.callSumWithThreeArguments()  
data: 0xd20...74110  
call to FunctionOverloading.callSumWithThreeArguments

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: FunctionOverloading.callSumWithTwoArguments()  
data: 0xd20...74110  
call to FunctionOverloading.callSumWithTwoArguments

Low level interactions

## Mathematical Function

```

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);
    }
}

```

**FILE EXPLORERS**

Workspaces

default\_workspace

- contracts
  - artifacts
  - 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
- scripts
- tests
- deps
- README.txt

**MathematicalFunction** MathematicalFunction → 1 reference(s)

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: MathematicalFunction.(fallback) data: 0xd20...74110

**Output :**

```

Solidity - Cryptographic Function.sol | Solidity - View and Pure Functions.sol | Remix - Ethereum IDE
pragma solidity ^0.5.0;
contract CryptographicFunction {
    function callKeccak256() public pure returns(bytes32 result){
        return keccak256("ABC");
    }
}

Solidity - Mathematical Function.sol | Solidity - View and Pure Functions.sol | Remix - Ethereum IDE
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 :

```

Solidity - Cryptographic Function.sol | Solidity - View and Pure Functions.sol | Remix - Ethereum IDE
pragma solidity ^0.5.0;
contract CryptographicFunction {
    function callKeccak256() public pure returns(bytes32 result){
        return keccak256("ABC");
    }
}

Solidity - Mathematical Function.sol | Solidity - View and Pure Functions.sol | Remix - Ethereum IDE
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);
    }
}

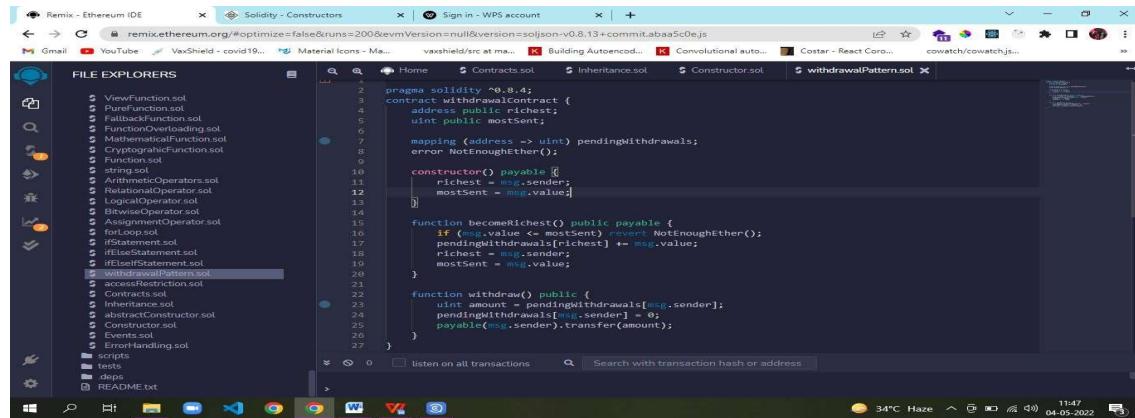
```

# Practical No : 4

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

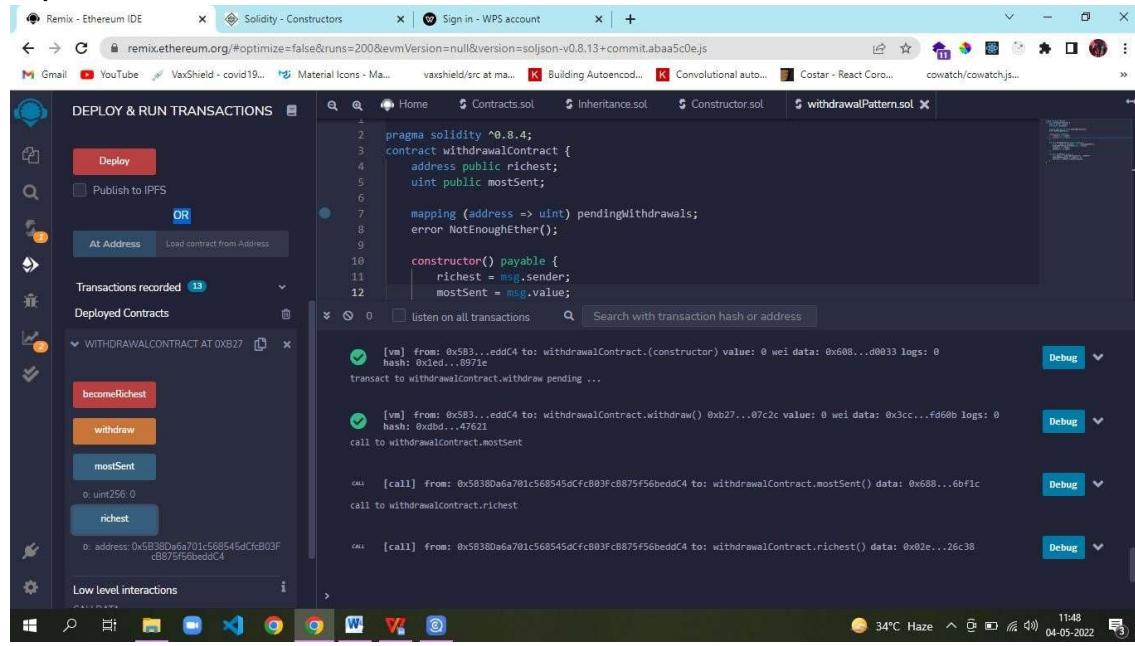
4a) Withdrawal Pattern, Restricted Access. Withdrawal Pattern :

Code :



```
pragma solidity ^0.8.4;
contract withdrawalContract {
    address public richiest;
    uint public mostSent;
    mapping (address => uint) pendingWithdrawals;
    error NotEnoughEther();
    constructor() payable {
        richiest = msg.sender;
        mostSent = msg.value;
    }
    function becomeRichest() public payable {
        if (msg.value <= mostSent) revert NotEnoughEther();
        pendingWithdrawals[richiest] += msg.value;
        richiest = msg.sender;
        mostSent = msg.value;
    }
    function withdraw() public {
        uint amount = pendingWithdrawals[msg.sender];
        pendingWithdrawals[msg.sender] = 0;
        payable(msg.sender).transfer(amount);
    }
}
```

Output :



The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" sidebar open. A red "Deploy" button is visible. Below it, there are two options: "Publish to IPFS" and "At Address". Under "Transactions recorded", there is a list with 13 items. The first item is "WITHDRAWALCONTRACT AT 0XB27". To the right of the transaction list, the Solidity code for the withdrawalPattern.sol contract is displayed. At the bottom of the screen, a terminal window shows several log entries from the VM, indicating successful deployment and calls to the contract's functions.

```
pragma solidity ^0.8.4;
contract withdrawalContract {
    address public richiest;
    uint public mostSent;
    mapping (address => uint) pendingWithdrawals;
    error NotEnoughEther();
    constructor() payable {
        richiest = msg.sender;
        mostSent = msg.value;
    }
    function becomeRichest() public payable {
        if (msg.value <= mostSent) revert NotEnoughEther();
        pendingWithdrawals[richiest] += msg.value;
        richiest = msg.sender;
        mostSent = msg.value;
    }
    function withdraw() public {
        uint amount = pendingWithdrawals[msg.sender];
        pendingWithdrawals[msg.sender] = 0;
        payable(msg.sender).transfer(amount);
    }
}
```

## Restricted Access :

### **Code :**

The screenshot shows the Remix Ethereum IDE interface with two tabs open, both titled "Solidity - Constructors".

The top tab displays the following Solidity 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;
    }

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

The bottom tab displays the same code, slightly offset vertically.

### **Output :**

The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" tab active.

The left sidebar shows the "Deployed Contracts" section, which lists the "ACCESSRESTRICTION AT 0x908...A5" contract. Below it, the "Low level interactions" section shows several buttons:

- buyContract
- changeOwner (address \_newOwner)
- lastOwnerCh...
- owner

The right pane shows the deployed contract's source code and a log of transactions:

```
creation of Accessrestriction pending...
[vm] From: 0x5B3...addC4 to: Accessrestriction.(constructor) value: 0 wei data: 0x600...38029 logs: 0
call to Accessrestriction.lastOwnerChange
[vm] [call] # from: 0x5B3Bdada701c568545dcFcB03FcB875f56beddC4 to: Accessrestriction.lastOwnerChange() data: 0x5ff...711e#
call to Accessrestriction.owner
[vm] [call] # from: 0x5B3Bdada701c568545dcFcB03FcB875f56beddC4 to: Accessrestriction.owner() data: 0xbde...5cb5b
```

```

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 >= lastOwnerChange + _time);
        _;
    }
}

ContractDefinition AccessRestriction 1 Reference(s) ▾

call to AccessRestriction.lastOwnerChange
call to AccessRestriction.owner
call to AccessRestriction.changeOwner

[...]

```

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

### Contracts:

#### Code :

```

pragma solidity ^0.5.0;

contract C {
    //private state variable
    uint private data;
    //public state variable
    uint public info;
    //constructor
    constructor() public {
        info = 10;
    }
    //private function
    function increment(uint a) private pure returns(uint) { return a + 1; }

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

//External Contract
contract D {
    function readData() public returns(uint) {
        C c = new C();
        c.updateData(5);
        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) ▾

```

## Output :

```

Solidity - Contracts
Solidity - Deploy & Run Transactions

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
info
o: uint256: 0
o: uint256: 10

https://remix.readthedocs.io/en/latest/run.html

Solidity - Contracts
Solidity - Deploy & Run Transactions

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

Transactions recorded 2
Deployed Contracts
C AT 0xD91_39138 (MEMORY)

updateData
    7
    transaction
    [call] from: 0x5838Da6a701c568545dFcB03FcB875f56bedd4 to: C.getData() data: 0x3bc...5de30
    call to C.info

    [call] from: 0x5838Da6a701c568545dFcB03FcB875f56bedd4 to: C.info() data: 0x370...158e0
    transaction to C.updateData pending ...

    [vm] from: 0x583...edd4 to: C.updateData(uint256) 0xd91_39138 value: 0 wei data: 0x09e...000007 logs: 0
    hash: 0x6c2...cd890
    call to C.updateData

    [call] from: 0x5838Da6a701c568545dFcB03FcB875f56bedd4 to: C.getData() data: 0x3bc...5de30
    transaction to C.updateData pending ...

```

## Inheritance Code :

```

FILE EXPLORERS
Contracts.sol
Inheritance.sol

pragma solidity ^0.5.0;

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

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

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

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

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

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

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

    function getComputedResult() public {
        result = compute(c.info(), 5);
    }

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

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

```

Remix - Ethereum IDE Solidity - Inheritance

```

2.1 }
2.2 //DRAFTED Contract
2.3 contract C is E {
2.4     uint private result;
2.5     C private c;
2.6     constructor() public {
2.7         c = new C();
2.8     }
2.9     function getComputedResult() public {
3.0         result = compute(3, e);
3.1     }
3.2     function getResult() public view returns(uint) { return result; }
3.3     function getData() public view returns(uint) { return c.info(); }
3.4 }

```

FILE EXPLORERS

- functionModifier.sol
- ViewFunction.sol
- PureFunction.sol
- FallBackFunction.sol
- FunctionOverload.sol
- FunctionOverriding.sol
- CryptographicFunction.sol
- StringFunction.sol
- ArithmeticOperators.sol
- RelationalOperator.sol
- LogicalOperator.sol
- AssignmentOperator.sol
- forLoop.sol
- ifStatement.sol
- elseStatement.sol
- ifElseIfStatement.sol
- whileLoopPattern.sol
- Contracts.sol
- Inheritance.sol
- SubtractConstructor.sol
- Contract.sol
- Events.sol
- ErrorHandling.sol
- Function.sol
- tests
- deps

call to c.info

Listen on all transactions Search with transaction hash or address

33°C Haze 11:12 04-05-2022

## Output :

Remix - Ethereum IDE Solidity - Inheritance

Deploy & Run Transactions

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

Transactions recorded Deployed Contracts

C AT 0X7EF...8CB47 (MEMORY)

- updateData uint256 a
- getData
- o: uint256: 0
- info
- o: uint256: 20

Low level interactions CALLDATA

```

b ...
6 //public state variable
7 uint public info;
8
9 //constructor
10 constructor() public {
11     info = 20;
12 }
13
14 //private function
15 function increment(uint a) private pure returns(uint) { return a + 1; }
16
17 //public function

```

[call] from: 0x5B38Da6a701c568545dFcB03FcB875f56bedd4 to: C.info() data: 0x370...158ea creation of C pending...

[vm] from: 0x5B38Da6a701c568545dFcB03FcB875f56bedd4 to: C.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x291...c4638 call to C.getData

[call] from: 0x5B38Da6a701c568545dFcB03FcB875f56bedd4 to: C.getData() data: 0x3bc...5de30 call to C.info

[call] from: 0x5B38Da6a701c568545dFcB03FcB875f56bedd4 to: C.info() data: 0x370...158ea

33°C Haze 11:11 04-05-2022

Remix - Ethereum IDE Solidity - Inheritance

Deploy & Run Transactions

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

Transactions recorded Deployed Contracts

C AT 0X7EF...8CB47 (MEMORY)

- updateData 17
- getData
- o: uint256: 17
- info
- o: uint256: 20

Low level interactions CALLDATA

```

b ...
6 //public state variable
7 uint public info;
8
9 //constructor
10 constructor() public {
11     info = 20;
12 }
13
14 //private function
15 function increment(uint a) private pure returns(uint) { return a + 1; }
16
17 //public function

```

[call] from: 0x5B38Da6a701c568545dFcB03FcB875f56bedd4 to: C.getData() data: 0x3bc...5de30 call to C.info

[call] from: 0x5B38Da6a701c568545dFcB03FcB875f56bedd4 to: C.info() data: 0x370...158ea transact to C.updateData pending ...

[vm] from: 0x5B3...edd4 to: C.updateData(uint256) 0x7EF...8CB47 value: 0 wei data: 0x09e...00001 logs: 0 hash: 0x8c...f9e07 call to C.increment

[call] from: 0x5B38Da6a701c568545dFcB03FcB875f56bedd4 to: C.getData() data: 0x3bc...5de30

33°C Haze 11:11 04-05-2022

## 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;
    }
}
```

The bottom window shows the deployment of the contract and its interaction:

- Deploy**: The contract `constructorExample` has been deployed at address `0x58D...eddc4`.
- Transactions recorded**: A transaction was recorded for the `getValue` function.
- Deployed Contracts**: The deployed contract is `CONSTRUCTOREXAMPLE AT 0x58D...`.
- Low level interactions**: A transaction was sent to the `getValue` function, returning the value `0x209...65255`.

### Abstract Contracts :Code :

```
pragma solidity ^0.5.0;
contract abstractConstructor {
    function getResult() public view returns(uint);
}

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

The screenshot shows the Remix Ethereum IDE interface. On the left, the sidebar displays the 'DEPLOY & RUN TRANSACTIONS' section with a 'CONTRACT' dropdown set to 'Test - contracts/abstractConstructor.sol'. It includes options for 'Deploy', 'Publish to IPFS', and 'At Address'. Below this, 'Transactions recorded' and 'Deployed Contracts' sections are shown, with 'TEST AT 0X1C9...2B4BD (MEMORY)' expanded. The code editor on the right 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 = 18;
        uint b = 17;
        uint result = a + b;
        return result;
    }
}
```

The 'ContractDefinition Test' pane shows two transaction logs:

- [vm] from: 0x503...edd4 to: Test.(constructor) value: 0 wei data: 0x008...10032 logs: 0 hash: 0xfa...86e41 call to Test.getResult
- [call] from: 0x58380a6a701c568545dCfcB03FcB875f56bedd4 to: Test.getResult() data: 0xde2...92789

The status bar at the bottom indicates it's 12:16 on 04-05-2022.

## Interfaces :

### Code

The screenshot shows the Remix Ethereum IDE interface. On the left, the sidebar displays the 'DEPLOY & RUN TRANSACTIONS' section with a 'CONTRACT' dropdown set to 'Test - contracts/Interface.sol'. It includes options for 'Deploy', 'Publish to IPFS', and 'At Address'. Below this, 'Transactions recorded' and 'Deployed Contracts' sections are shown, with 'TEST AT 0X93F...C90CC (MEMORY)' expanded. The code editor on the right 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;
    }
}
```

The 'ContractDefinition Test' pane shows two transaction logs:

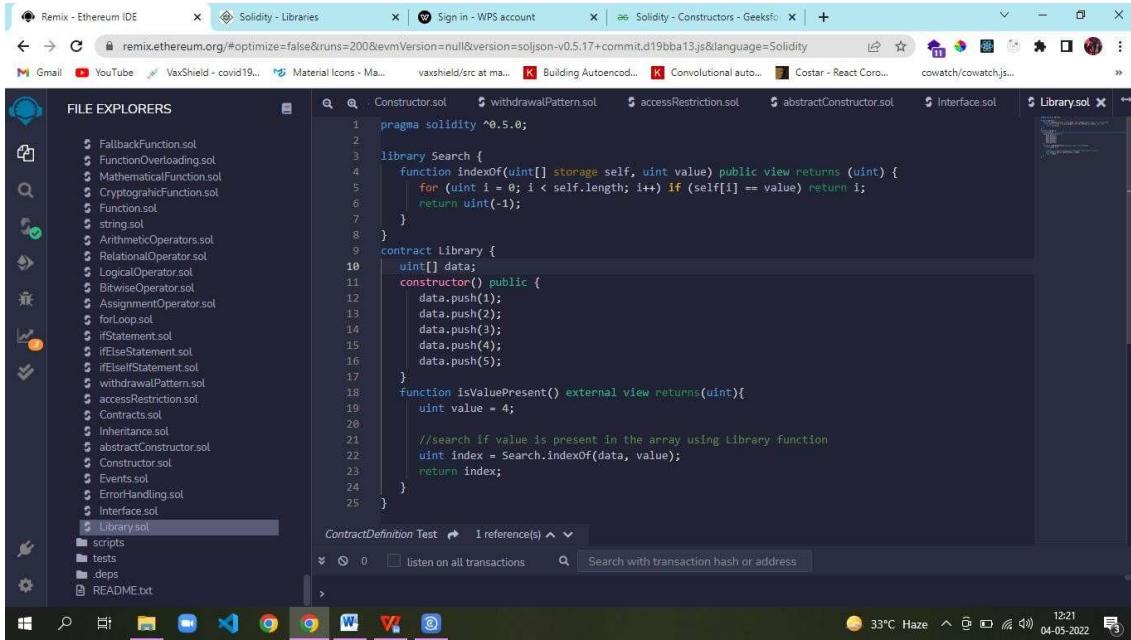
- [call] from: 0x58380a6a701c568545dCfcB03FcB875f56bedd4 to: Test.getResult() data: 0xde2...92789 creation of Test pending...
- [vm] from: 0x503...edd4 to: Test.(constructor) value: 0 wei data: 0x008...10032 logs: 0 hash: 0x776...0secf call to Test.getResult
- [call] from: 0x58380a6a701c568545dCfcB03FcB875f56bedd4 to: Test.getResult() data: 0xde2...92789

The status bar at the bottom indicates it's 12:19 on 04-05-2022.

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

### Libraries :

### Code :



The screenshot shows the Remix Ethereum IDE interface. The left sidebar is titled "FILE EXPLORERS" and lists several Solidity files: 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, Interface.sol, and Library.sol. The "Library.sol" file is currently selected. The main editor area displays the following Solidity code:

```
pragma solidity ^0.5.0;

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

contract Library {
    uint[] data;
    constructor() public {
        data.push(1);
        data.push(2);
        data.push(3);
        data.push(4);
        data.push(5);
    }

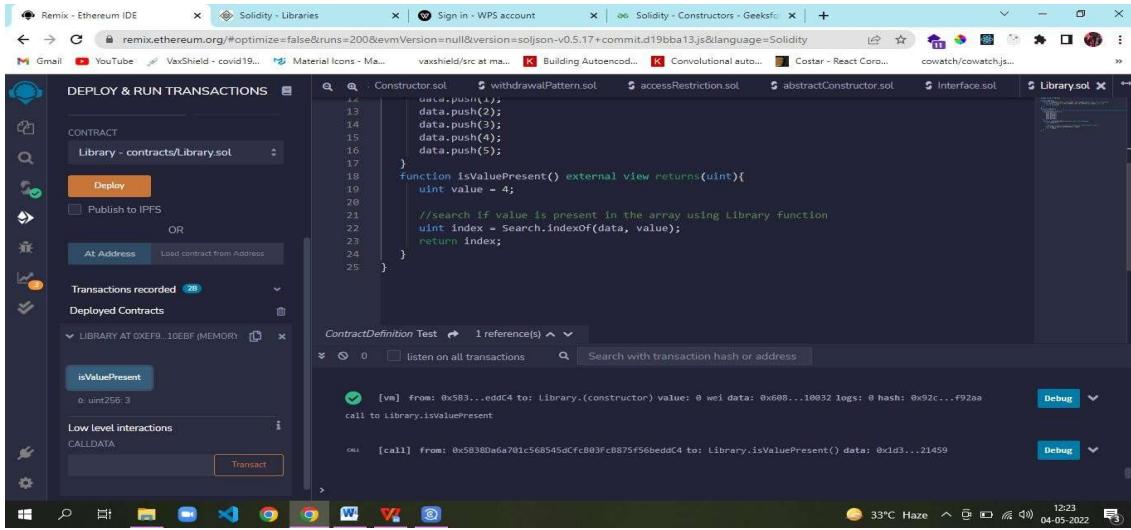
    function isValuePresent() external view returns(uint){
        uint value = 4;

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

The status bar at the bottom right indicates it's 33°C Haze, 12:21, and the date is 04-05-2022.

### Output :

### Assembly : Code :



The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" tab active. The "CONTRACT" dropdown is set to "Library - contracts/Library.sol". The "Deploy" button is highlighted. The "Transactions recorded" section shows one transaction: "isValuePresent" from address 0x5B3...eddC4 to Library.(constructor) with value 0 wei. The "Deployed Contracts" section shows the deployed contract at address 0x5B380a6a701c568545dCfc803Fc0875f56bedd4. The status bar at the bottom right indicates it's 33°C Haze, 12:23, and the date is 04-05-2022.

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar titled "FILE EXPLORERS" which lists various Solidity files. The main area displays the Solidity code for the "Assembly" contract. The code uses inline assembly to add two uint256 variables and store the result in memory.

```
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'
                b := d
                // 'd' is deallocated now
            }
            b := add(b, c)
        }
    }
}
```

The interface includes tabs for "Deploy & Run Transactions" and "FILE EXPLORERS". In the transaction tab, it shows a pending transaction for the "add" function with parameters 17 and 17. The transaction details show it originated from 0x580...edc4 and is headed to the Assembly contract at 0xd16...23f01. The status bar indicates the transaction was sent at 12:54 on 04-05-2022.

## Output :

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 contract named 'Events'. The code includes a constructor, state variables, and an event 'Increment' that logs the sum of two parameters. Below the code, the 'Transactions recorded' section shows a pending transaction for the 'getValue' function with inputs 500 and 300. The transaction details show it's from address 0x583... to Events合约地址, with a value of 0 wei and 1 log. The logs section shows the emitted event 'Increment' with the message 'value = 800'. The bottom status bar indicates the date as 04-05-2022 and the time as 12:37.

```
// 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;
    }
}
```

## Events:

### Code :

This screenshot shows the Remix Ethereum IDE with the file explorer open on the left, displaying a list of Solidity files. The 'Events.sol' file is currently selected. The main code editor area contains the same Solidity code as the previous screenshot, defining a 'Events' contract with a constructor, state variable 'value', and an 'Increment' event. The bottom status bar shows the date as 04-05-2022 and the time as 12:36.

```
// 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;
    }
}
```

## Outut :

### Error handling :

## Code :

The screenshot shows the Remix Ethereum IDE interface. The central area displays the Solidity code for a contract named `ErrorHandling`. The code includes two functions: `checkInput` and `odd`. The `checkInput` function takes a uint input and returns a string indicating if it's an even or odd number. The `odd` function takes a uint input and returns a boolean value indicating if it's odd.

```
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 :

The screenshot shows the Remix Ethereum IDE interface with the "DEPLOY & RUN TRANSACTIONS" tab selected. The left sidebar shows the deployment of the `ErrorHandling` contract at address `0x7f...501c`. Below the sidebar, three transaction calls are shown: `checkInput` with input `243` returning `"Input is Uint8"`, `Odd` with input `243` returning `true`, and another `Odd` call with the same input returning `true`.

```
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;
    }
}
```

