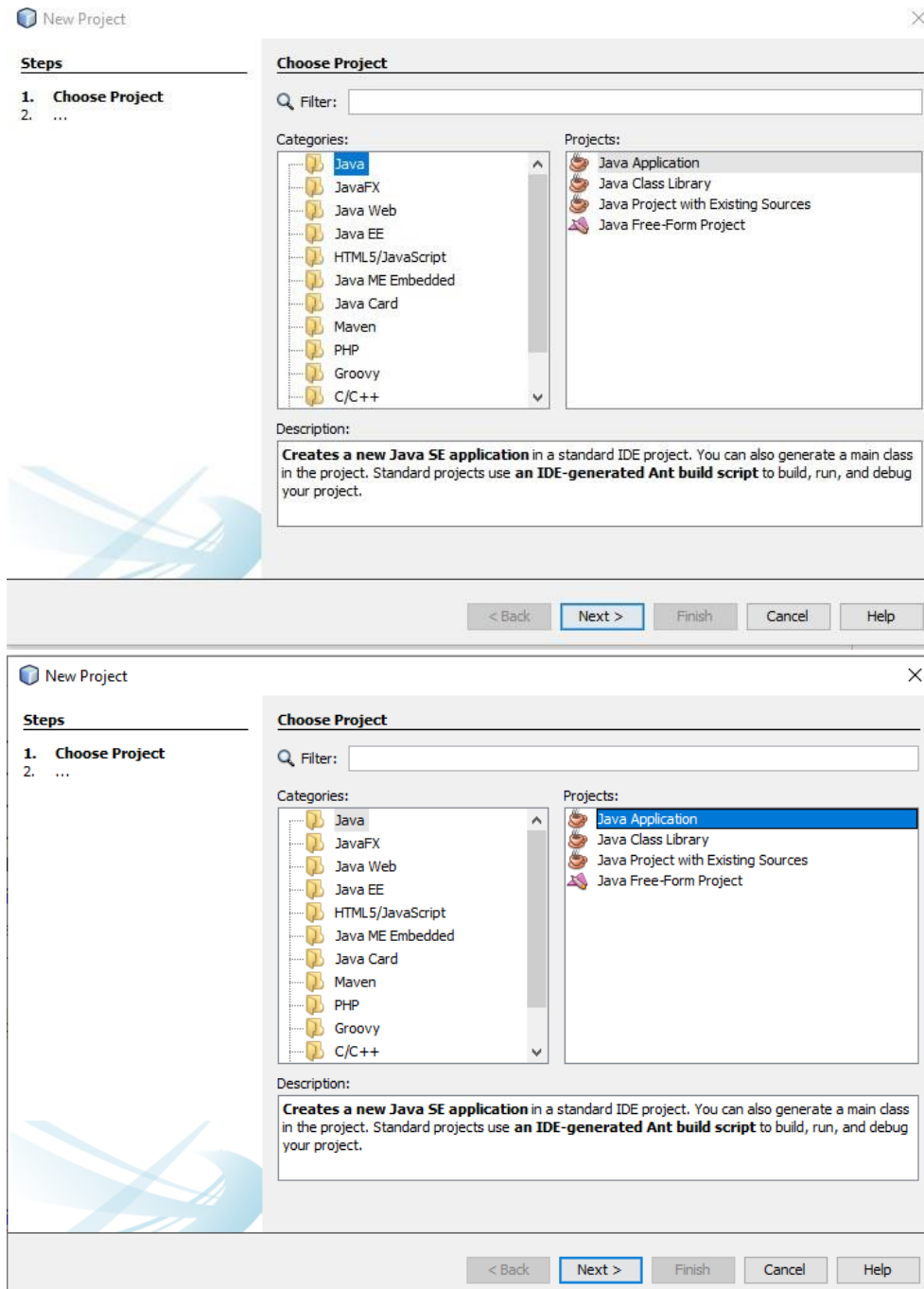


Practical No:01

Aim: Write the following programs for Blockchain in python:

1] A simple client class that generates the private and public keys by using the built-in python algorithm and tests it.

Generating keys using DSA



Code :

```
package javaapplication32;
import java.security.KeyPair;
import java.security. KeyPairGenerator;
import java.security. PrivateKey;
public static void main(String args[]) throws Exception{
    KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("DSA");
    keypairGen.initialize (2048);
    KeyPair pair = keypairGen.genKeyPair();
    PrivateKey privkey = pair.getPrivate();

    PublicKey publicKey = pair.getPublic();
    System.out.println("Keys Generated");
    System.out.println("PrivateKey"+privkey);
    System.out.println("PublicKey"+publicKey);
```

```

26
27 //Getting the public key from the key pair
28 PublicKey publicKey = pair.getPublic();
29
30 System.out.println("Keys Generated");
31 System.out.println("PrivateKey"+privKey);
32 System.out.println("PublicKey"+publicKey);
33 }
34 }
35

```

Output:

```

: Output - JavaApplication32 (run)

run:
Keys Generated
PrivateKeySun.security.provider.DSAPrivateKey@ffff1339
PublicKeySun DSA Public Key
Parameters:
p:
8f7935d9 b9aae9bf abed887a cf4951b6 f32ec59e 3baf3718 e8eac496 1f3efd36
06e74351 a9c41833 39b809e7 c2a61c53 9ba7475b 85d011ad b8b47987 75498469
5cac0e8f 14b33608 28a22ffa 27110a3d 62a99345 3409a0fe 696c4658 f84bdd20
819c3709 a01057b1 95adcd00 233dba54 84b6291f 9d648ef8 83448677 979cec04
b434a6ac 2e75e998 5de23db0 292fc111 8c9ffa9d 8181e733 8db792b7 30d7b9e3
49592f68 09987215 3915ea3d 6b8b4653 c633458f 803b32a4 c2e0f272 90256e4e
3f8a3b08 38alc450 e4e18c1a 29a37ddf 5ea143de 4b66ff04 903ed5cf 1623e158
d487c608 e97f211c d81dca23 cb6e3807 65f822e3 42be484c 05763939 601cd667
q:
baf696a6 8578f7df dee7fa67 c977c785 ef32b233 bae580c0 bcd5695d
baf696a6 8578f7df dee7fa67 c977c785 ef32b233 bae580c0 bcd5695d
g:
16a65c58 20485070 4e7502a3 9757040d 34da3a34 78c154d4 e4a5c02d 242ee04f
96e61e4b d0904abd ac8f37ee b1e09f31 82d23c90 43cb642f 88004160 edf9ca09
b32076a7 9c32a627 f2473e91 879ba2c4 e744bd20 81544cb5 5b802c36 8d1fa83e
d489e94e 0fa0688e 32428a5c 78c478c6 8d0527b7 1c9a3abb 0b0be12c 44689639
e7d3ce74 db101a65 aa2b87f6 4c6826db 3ec72f4b 5599834b b4edb02f 7c90e9a4
96d3a55d 535bebfc 45d4f619 f63f3ded bb873925 c2f224e0 7731296d a887ec1e
4748f87e fb5fdeb7 5484316b 2232dee5 53dda02 112b0dlf 02da3097 3224fe27
aeda8b9d 4b2922d9 ba8be39e d9e103a6 3c52810b c688b7e2 ed4316e1 ef17dbde

y:
59a29e0e 3dee7aa8 dc0526c9 6130416a 3870485c 7dfe447 c34da309 c046a809
4387f933 30fcf982 c3335f0e 961baa31 6b031780 9247eacb a6786fa0 77e3d98f
5c23b611 d945bf27 b12cfd1b 37c77318 a1a101c4 7bf04ee6 0d3db4e5 ab10b610
d7b1c74f 771e3e24 ec27c5d3 8d6e781a c023ed70 c247109f 83dcd6f0 8e4c2b83
4be82d4e b96b2ed0 36d32a4f d66ad10e dd5708db 0dc16d78 0cfe810f bea023f0
16ad22c4 299dlee8 e5471991 1281f40e f6985403 ccce39c9 bell1f66 11ac7d9e
0040df2d f66a7df2 aafe1547 f375d184 10fc2ff4 0e470c1d a591806f c03e7c4c
882f51da 3bd8be22 f0d2a319 e5cb695b 6ec24fc2 f8eef769 9143ce2e bf5139e7

BUILD SUCCESSFUL (total time: 0 seconds)

```

Generating keys using RSA

```
[1]: pip install pycryptodome
```

Requirement already satisfied: pycryptodome in c:\users\admin\anaconda3\lib\site-packages (3.20.0)
Note: you may need to restart the kernel to use updated packages.

```
[2]: from Crypto.PublicKey import RSA  
from Crypto.Cipher import PKCS1_OAEP
```

```
class RSAKeyGenerator:  
    def __init__(self, key_size=2048):  
        self.key_size=key_size  
    def generate_keys(self):  
        #Generate RSA key pair  
        key_pair = RSA.generate(self.key_size)  
  
        #Extract public and private key  
        public_key=key_pair.publickey().export_key()  
        private_key=key_pair.export_key()  
        return private_key, public_key
```

```
class RSAClient:  
    def __init__(self):  
        self.key_generator=RSAKeyGenerator()  
    def encrypt_message(self, public_key, message):  
        #Load public key  
        recipient_key=RSA.import_key(public_key)  
  
        #create cipher object  
        cipher_rsa=PKCS1_OAEP.new(recipient_key)  
  
        #Encrypt the message  
        encrypted_message=cipher_rsa.encrypt(message.encode())  
        return encrypted_message  
    def decrypt_message(self, private_key, encrypted_message):  
        #Load private key  
        private_key=RSA.import_key(private_key)  
  
        #create cipher object  
        cipher_rsa=PKCS1_OAEP.new(private_key)  
  
        #Decrypt the message  
        decrypted_message=cipher_rsa.decrypt(encrypted_message)  
        return decrypted_message.decode()
```

```

#Example usage
if __name__ == "__main__":
    client=RSAClient()

    #Generate keys
    private_key,public_key=client.key_generator.generate_keys()

    #Encrypt and decrypt a message
    message="Hello, this is a test message!"
    encrypted_message=client.encrypt_message(public_key,message)
    decrypted_message=client.decrypt_message(private_key,encrypted_message)

    print("Original Message :", message)
    print("Encrypted Message :", encrypted_message)
    print("Decrypted Message :", decrypted_message)

```

Code :-

```

[1]: pip install pycryptodome
[2]: from Crypto. PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
class RSAKeyGenerator:
def __init__(self,key_size=2048):
self.key_size=key_size
def generate_keys(self):
#Generate RSA key pair
key_pair = RSA.generate(self.key_size)
#Extract public and private key
public_key=key_pair.publickey().export_key()
private_key=key_pair.export_key() return private_key,public_key
class RSAClient:
def __init__(self):
self.key_generator=RSAKeyGenerator()
def encrypt_message(self, public_key, message): #Load public key
recipient_key=RSA. import_key (public_key)
#create cipher object
cipher_rsa=PKCS1_OAEP.new(recipient_key)
#Encrypt the message
encrypted_message=cipher_rsa. encrypt (message.encode()) return encrypted_message
def decrypt_message (self, private_key, encrypted_message): #Load private key
private_key=RSA. import_key (private_key)
#create cipher object
cipher_rsa=PKCS1_OAEP.new(private_key)
#Decrypt the message
decrypted_message=cipher_rsa.decrypt (encrypted_message)
return decrypted_message.decode()

#Example usage
if __name__ == "__main__":
client=RSAClient()

```

```

#Generate keys
private_key,public_key=client.key_generator.generate_keys()
#Encrypt and decrypt a message
message="Hello, this is a test message!"
encrypted_message=client.encrypt_message(public_key, message)
decrypted_message=client.decrypt_message(private_key, encrypted_message)
print("Original Message : message)
print("Encrypted Message:" encrypted message)
print("Decrypted Message:", decrypted_message)

```

Output:

```

Original Message : Hello, this is a test message!
Encrypted Message : b'\xa1\x13+\xf8?58\x19\x08\x91k\x03\x95\x7fE\x02\xa4\xadI\xef\xd0v\x92um\x05\xe8\x8a\xa0!Y3\xd5\r\xf7{\x8cK\x91\xb2h\xe7\x97\x07\xd2
\n+v\xe5!#\x07\x8c\xb0\x06\xf5&\x1aB\xe7\xc9jq\xaf\x9c\xa8!S\xc8\x1c\x1cJ\xc1wE\x95\x86w\xda%\x05\xe1\xc6JIh\xdaV\x7f\x94M)\x0ex\xc3_.\xab\xa5\xab\x9ehf
\x8c\x0ca\xd4\xf8\\\xd1<\xdb0$\x16\xe3o\xe9\xebwL\xfe\x93\x8e9\x99\xed\xfa1\xd8\xdbL8j\xbd@\x8c\x02\x92\x1f\xaf\x00%\x8dj7\x0894\x1b@}\xa6\xc4/5v<\x01+
\xcaP(\xe3\xf3<3\xc9\xeeh\x8f1n\x88\x03%\x87q\x90b\x0cr\x9e\xd6}\xd0\x96E\xfb\x02\xa1's5s\x93\xaa\x9d\xe4\x18_\xb1o\xea7\xbd~\xe4\xa8s\x9emJ\xfb\xab\x14
\xdc\x12 <%.k\xea\x1c\xce`m\xc7\xdd\x92;f\xc7\xa7"\x9f\x88\x0e\xc9^j\x99!j\x7f\x8c\xc8\x00:J$/'
Decrypted Message : Hello, this is a test message!

```

II] A transaction class to send and receive money and test it.

```
jupyter Prac1-Transaction Last Checkpoint: 31 minutes ago
File Edit View Run Kernel Settings Help
+ ✂ 📄 📌 ▶ ■ 🔁 ▶▶ Code ▼

[4]: import hashlib
import json
from time import time

class Transaction:
    def __init__(self, sender, recipient, amount):
        self.sender=sender
        self.recipient=recipient
        self.amount=amount
        self.timestamp=time()
    def to_dict(self):
        return {
            'sender':self.sender,
            'recipient':self.recipient,
            'amount':self.amount,
            'timestamp':self.timestamp
        }
    def hash_transaction(self):
        transaction_string=json.dumps(self.to_dict(),sort_keys=True)
        return hashlib.sha256(transaction_string.encode()).hexdigest()

class Blockchain:
    def __init__(self):
        self.chain=[]
        self.pending_transactions=[]
        self.create_block(previous_hash = '1')
        #create the genesis block
    def create_block(self,previous_hash):
        block= {
            'index': len(self.chain) + 1,
            'timestamp': time(),
            'transactions': self.pending_transactions,
            'previous_hash': previous_hash or self.hash_block(self.chain[-1])
        }
        self.pending_transactions=[]
        self.chain.append(block)
    def add_transaction(self,transaction):
        self.pending_transactions.append(transaction.to_dict())
    def hash_block(self,block):
        block_string=json.dumps(block,sort_keys=True)
        return hashlib.sha256(block_string.encode()).hexdigest()
```



```

#Test the transaction and blockchain classes
if __name__ == "__main__":
    #create a blockchain
    blockchain=Blockchain()

    #create a transaction
    transaction=Transaction(sender="Alice",recipient="Bob",amount=10)

    #Add transaction to the blockchain
    blockchain.add_transaction(transaction)

    #create a new block
    blockchain.create_block(previous_hash=None)

    #print the blockchain
    print("Blockchain:")
    print(json.dumps(blockchain.chain,indent=4))

```

Code:-

```

[4]: import hashlib import json
from time import time
class Transaction:
def __init__(self, sender, recipient, amount):
self.sender=sender
self.recipient=recipient self.amount=amount
self.timestamp=time()
def to_dict(self):
return {
'sender': self.sender,
'recipient': self.recipient,
'amount': self.amount,
'timestamp": self.timestamp
}
def hash_transaction(self):
transaction_string=json.dumps (self.to_dict(), sort_keys=True)
hashlib.sha256(transaction_string.encode()).hexdigest()
return
class Blockchain:
def __init__(self):
self.chain=[]
self.pending_transactions=[]
self.create_block (previous_hash = '1')
#create the genesis block
def create_block (self, previous_hash):
block= {
'index': len(self.chain) + 1,
'timestamp': time(),
'transactions': self.pending_transactions,
'previous_hash': previous_hash or self.hash_block(self.chain[-1])

```



```

}

self.pending_transactions=[]
self.chain.append(block)
def add_transaction(self, transaction):
self.pending_transactions.append(transaction.to_dict())
def hash_block(self, block):
block_string=json.dumps (block, sort_keys=True)
return hashlib.sha256 (block_string.encode()).hexdigest()

if __name__=="__main__":
#create a blockchain
blockchain=Blockchain()
#create a transaction
transaction=Transaction (sender="Alice", recipient="Bob", amount=10)
#Add transaction to the blockchain blockchain.add_transaction (transaction)
#create a new block
blockchain.create_block (previous_hash=None)
#print the blockchain
print("Blockchain:")
print(json.dumps (blockchain.chain, indent=4))

```

Output:

```

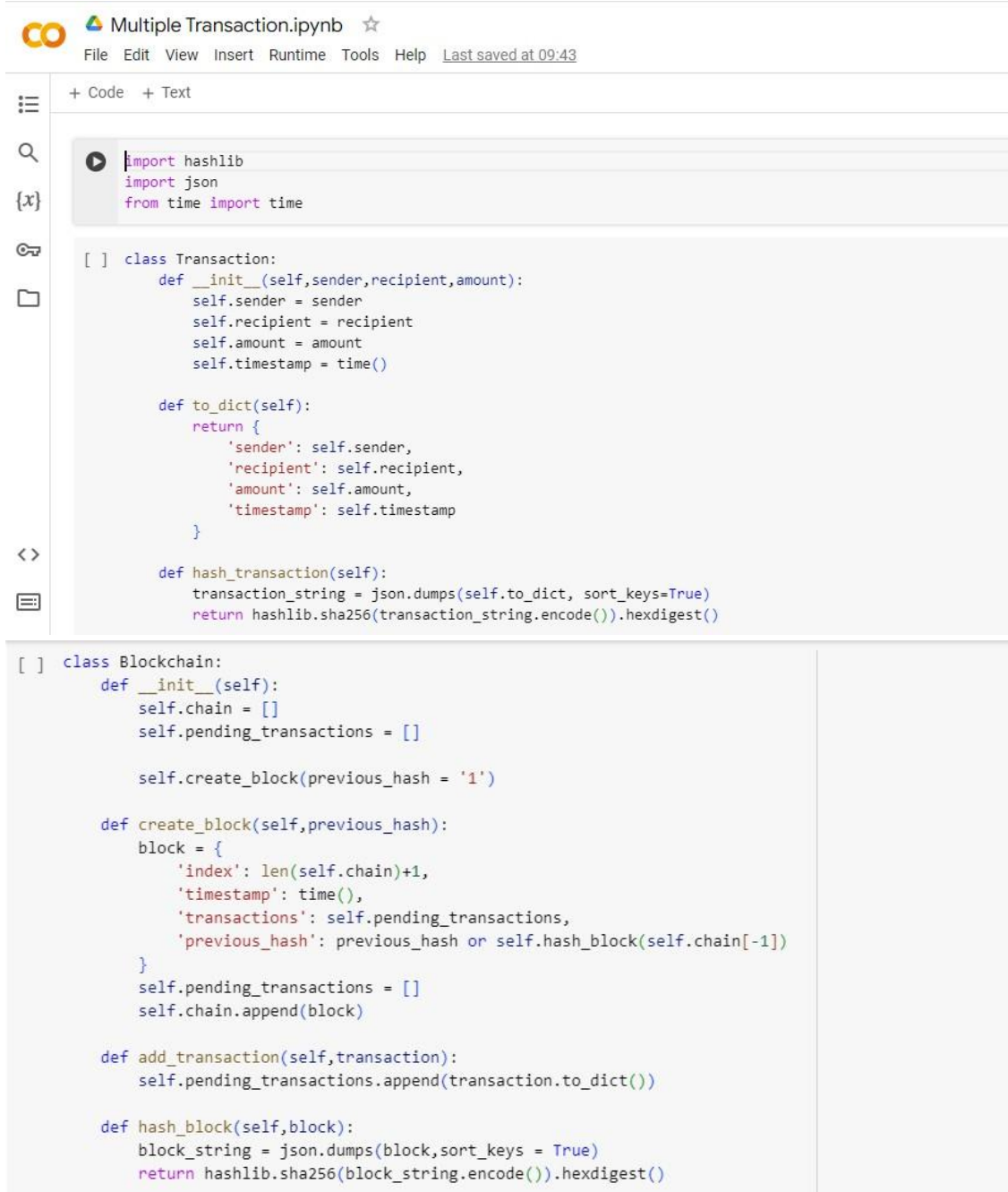
Blockchain:
[
  {
    "index": 1,
    "timestamp": 1713587481.4855232,
    "transactions": [],
    "previous_hash": "1"
  },
  {
    "index": 2,
    "timestamp": 1713587481.4855232,
    "transactions": [
      {
        "sender": "Alice",
        "recipient": "Bob",
        "amount": 10,
        "timestamp": 1713587481.4855232
      }
    ],
    "previous_hash": "1cc0053dd7951aa968f56693148502e1a48563d5b68f2cb3c085de19da58e966"
  }
]

```

Practical No: 02

Aim: Write the following programs for Blockchain in Python

I) Create multiple transactions and display them



```
Multiple Transaction.ipynb ☆
File Edit View Insert Runtime Tools Help Last saved at 09:43

+ Code + Text

import hashlib
import json
from time import time

[ ] class Transaction:
    def __init__(self, sender, recipient, amount):
        self.sender = sender
        self.recipient = recipient
        self.amount = amount
        self.timestamp = time()

    def to_dict(self):
        return {
            'sender': self.sender,
            'recipient': self.recipient,
            'amount': self.amount,
            'timestamp': self.timestamp
        }

    def hash_transaction(self):
        transaction_string = json.dumps(self.to_dict, sort_keys=True)
        return hashlib.sha256(transaction_string.encode()).hexdigest()

[ ] class Blockchain:
    def __init__(self):
        self.chain = []
        self.pending_transactions = []

        self.create_block(previous_hash = '1')

    def create_block(self, previous_hash):
        block = {
            'index': len(self.chain)+1,
            'timestamp': time(),
            'transactions': self.pending_transactions,
            'previous_hash': previous_hash or self.hash_block(self.chain[-1])
        }
        self.pending_transactions = []
        self.chain.append(block)

    def add_transaction(self, transaction):
        self.pending_transactions.append(transaction.to_dict())

    def hash_block(self, block):
        block_string = json.dumps(block, sort_keys = True)
        return hashlib.sha256(block_string.encode()).hexdigest()
```

```
[ ] if __name__ == "__main__":

    blockchain = Blockchain()

    transcation = Transaction(sender="Alice",recipient="Bob",amount=10)

    blockchain.add_transaction(transcation)

    blockchain.create_block(previous_hash = None)

    transcation1 = Transaction(sender="Bob",recipient="Jhon",amount=10)

    blockchain.add_transaction(transcation1)

    blockchain.create_block(previous_hash = None)

    print("Blockchain: ")
    print(json.dumps(blockchain.chain, indent = 4))
```

Code:-

```
Import hashlib import json
from time import time
[] class Transaction:
def __init__(self, sender, recipient, amount):
self.sender sender
self.recipient recipient
self.amount amount
self.timestamp time()
def to dict(self):
return {
}
'sender': self.sender,
'recipient': self.recipient, 'amount': self.amount,
'timestamp': self.timestamp
def hash_transaction(self):
transaction_string = json.dumps (self.to_dict, sort_keys=True) return
hashlib.sha256(transaction_string.encode()).hexdigest()
class Blockchain:
def __init__(self):
self.chain []
```

```

self.pending_transactions = []
self.create_block (previous_hash='1')
def create_block(self,previous_hash): block = {
}
'index': len(self.chain)+1, 'timestamp': time(),
'transactions': self.pending_transactions,
"previous_hash": previous_hash or self.hash_block(self.chain[-1])
self.pending_transactions = []
self.chain.append(block)
def add_transaction(self, transaction):
self.pending_transactions.append(transaction.to_dict())
def hash_block(self, block):
block_string = json.dumps (block, sort_keys = True)
return hashlib.sha256 (block_string.encode()).hexdigest()

if __name__=="__main__":
blockchain = Blockchain()
transaction = Transaction(sender="Alice", recipient="Bob", amount=10)
blockchain.add_transaction(transaction)
blockchain.create_block (previous_hash = None)
transaction1 = Transaction (sender="Bob", recipient="Jhon", amount=18)
blockchain.add_transaction(transaction1)
blockchain.create_block (previous_hash = None)
print("Blockchain: ")
print(json.dumps (blockchain.chain, indent = 4))

```

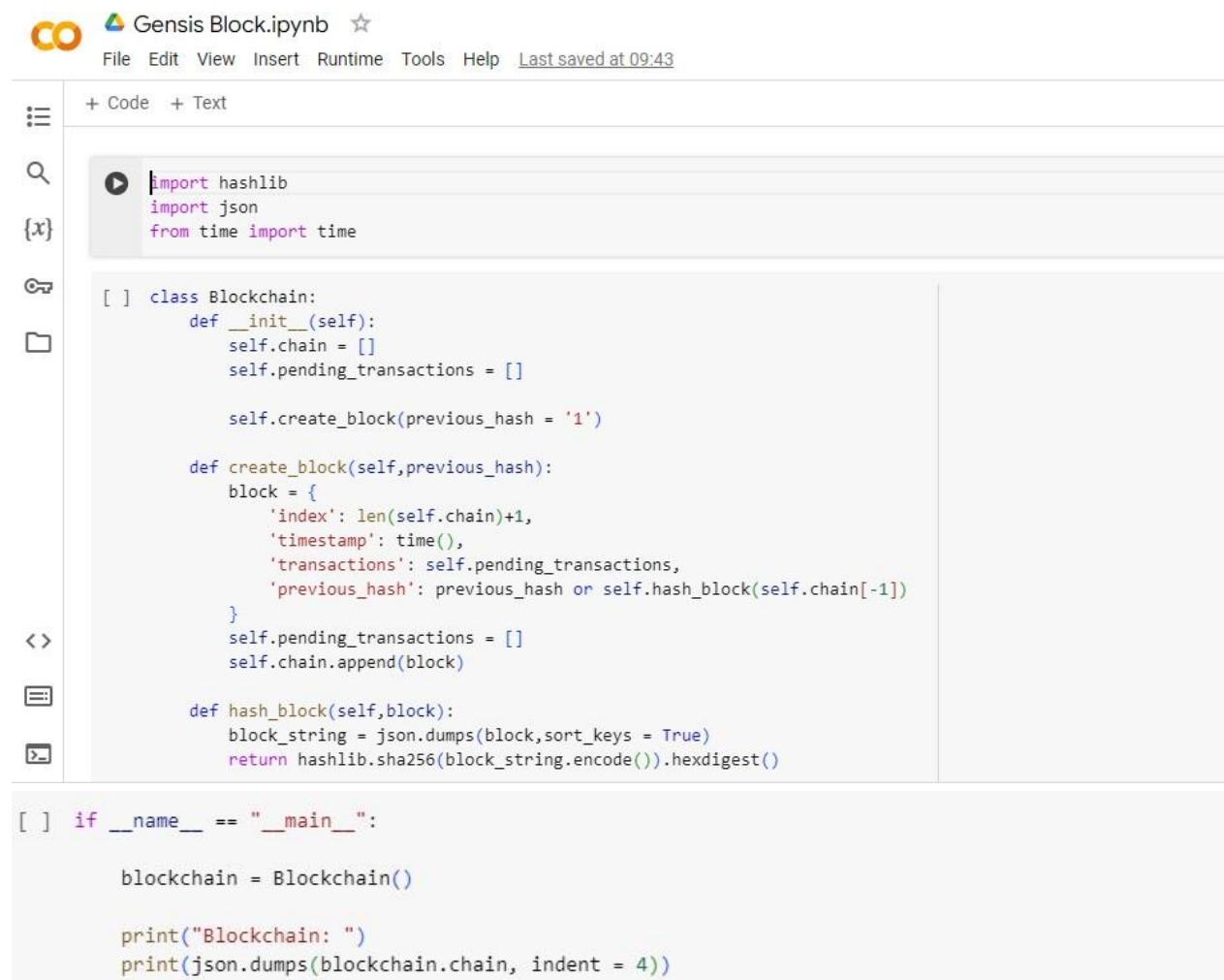
Output:

```

Blockchain:
[
  {
    "index": 1,
    "timestamp": 1714709055.7096033,
    "transactions": [],
    "previous_hash": "1"
  },
  {
    "index": 3,
    "timestamp": 1714709055.7096033,
    "transactions": [
      {
        "sender": "Bob",
        "recipient": "Jhon",
        "amount": 10,
        "timestamp": 1714709055.7096033
      }
    ],
    "previous_hash": "c382fda82ca2f6c0d861fc7a5da1070e19ca013deff3a80808717c8b4375fb19"
  }
]

```

II] Create a blockchain, a genesis block and execute it.



The screenshot shows a Jupyter Notebook interface with the title 'Gensis Block.ipynb'. The code is as follows:

```
import hashlib
import json
from time import time

[ ] class Blockchain:
    def __init__(self):
        self.chain = []
        self.pending_transactions = []

        self.create_block(previous_hash = '1')

    def create_block(self, previous_hash):
        block = {
            'index': len(self.chain)+1,
            'timestamp': time(),
            'transactions': self.pending_transactions,
            'previous_hash': previous_hash or self.hash_block(self.chain[-1])
        }
        self.pending_transactions = []
        self.chain.append(block)

    def hash_block(self, block):
        block_string = json.dumps(block, sort_keys = True)
        return hashlib.sha256(block_string.encode()).hexdigest()

[ ] if __name__ == "__main__":

    blockchain = Blockchain()

    print("Blockchain: ")
    print(json.dumps(blockchain.chain, indent = 4))
```

Code -:

```
Import hashlib import json
from time import time
[ ] class Blockchain:
def __init__(self): self.chain []
If
__name__
self.pending transactions = []
self.create_block(previous_hash = '1')
def create_block(self, previous_hash):
block = {
```

```

'index': len(self.chain)+1, 'timestamp': time(),
'transactions': self.pending_transactions,
'previous_hash': previous_hash or self.hash_block(self.chain[-1])
}

```

```

self.pending_transactions = []
self.chain.append(block)
def hash_block(self, block):
    block_string = json.dumps(block, sort_keys = True)
    return hashlib.sha256(block_string.encode()).hexdigest()
if __name__ == "__main__":
    blockchain = Blockchain()
    print("Blockchain: ")
    print(json.dumps(blockchain.chain, indent = 4))

```

Output:

```

Blockchain:
[
  {
    "index": 1,
    "timestamp": 1714709377.308939,
    "transactions": [],
    "previous_hash": "1"
  }
]

```

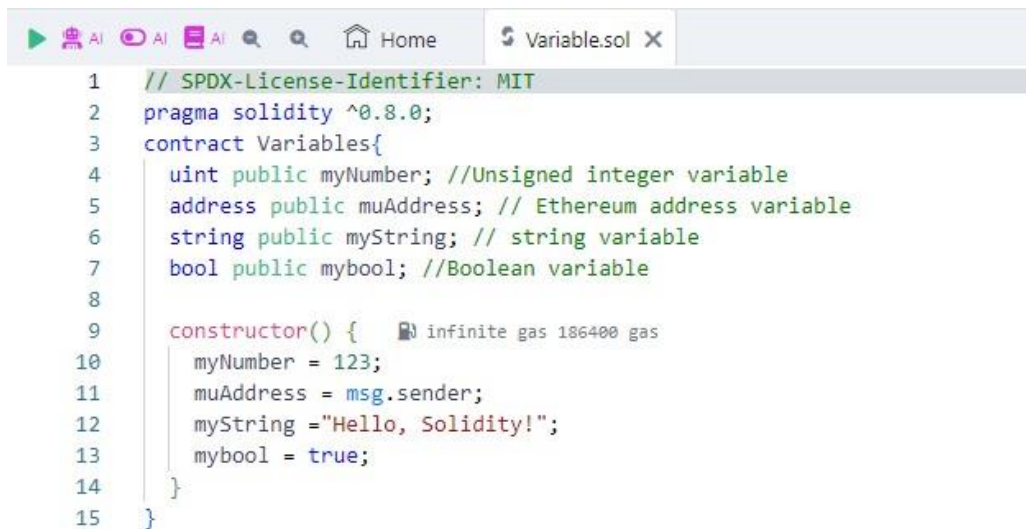
Practical No: 03

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

- 1) Variable
- 2) Operator
- 3) Loops
- 4) Decision Making
- 5) Strings

Open Remix.ide

1] Variable








The screenshot shows the OpenRemix IDE interface. At the top, there's a toolbar with icons for AI, search, and navigation. Below the toolbar, a tab labeled 'Variable.sol' is active. The code editor displays a Solidity contract named 'Variables'. The code includes a SPDX license identifier, a pragma statement for Solidity version ^0.8.0, and a contract definition. Inside the contract, four public variables are declared: 'myNumber' of type 'uint', 'muAddress' of type 'address', 'myString' of type 'string', and 'mybool' of type 'bool'. A constructor function is also defined, which initializes 'myNumber' to 123, 'muAddress' to 'msg.sender', 'myString' to 'Hello, Solidity!', and 'mybool' to 'true'. The code is line-numbered from 1 to 15.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3 contract Variables{
4     uint public myNumber; //Unsigned integer variable
5     address public muAddress; // Ethereum address variable
6     string public myString; // string variable
7     bool public mybool; //Boolean variable
8
9     constructor() { infinite gas 186400 gas
10         myNumber = 123;
11         muAddress = msg.sender;
12         myString = "Hello, Solidity!";
13         mybool = true;
14     }
15 }
```


Code:

```
pragma solidity ^0.8.0;
contract Variables{
uint public myNumber
address public muAddress;
string public myString;
bool public mybool;
constructor(){
myNumber = 123;
muAddress msg.sender;
myString "Hello, Solidity!";
mybool true;
}
```


Output:



SOLIDITY COMPILER

COMPILER + 

0.8.25+commit.b61c2a91

☐ Include nightly builds

☐ Auto compile

☐ Hide warnings


Advanced Configurations

Compile Variable.sol

Compile and Run script

DEPLOY & RUN TRANSACTIONS

VM

ACCOUNT 

0x5B3...eddC4 (99.999999999%)

GAS LIMIT

☒ Estimated Gas

☐ Custom

3000000

VALUE

0

Wei

CONTRACT

Variables - contracts/Variable.sol

evm version: cancun

Deploy

VARIABLES AT 0XD8B...33FA8 (ME)

Balance: 0 ETH

muAddress

mybool

myNumber

myString

Balance: 0 ETH

muAddress
0: address: 0x5B38Da6a701c568545dCfcB03Fc8875f56beddC4

mybool
0: bool: true

myNumber
0: uint256: 123

myString
0: string: Hello, Solidity!

2] Operator

```

pragma solidity ^0.8.0;
contract Operators{
    uint public result;
    bool public ans;

    constructor() {
        uint a = 10;
        uint b = 5;

        //Arithmetic operators
        result = a + b; // Addition
        result = a - b; // Substraction
        result = a * b; // Multiplication
        result = a / b; // Division
        result = a % b; // Modulus

        // //Comparison operators
        ans = (a == b); //Equal to
        ans = (a != b); // Not Equal to
        ans = (a > b); //Greater than
        ans = (a < b); //Less than
        ans = (a >= b); //Greater than or equal to
        ans = (a <= b); //Less than or Equal to

        //Logical Operators
        ans = (a > 0 && b > 0); //Logical AND
        ans = (a > 0 || b > 0); //Logical OR
        ans = !(a > 0); //Logical NOT
    }
}

```

Code:-

```
pragma solidity ^0.8.0;
contract Operators{
uint public result;
bool public ans;
constructor(){
uint a = 10;
uint b = 5;
result = a + b;
result= a- b;
result =a *b
result= a /b;
result a % b;
////Comparison operators
ans = (a==b); //Equal to ans
ams = (a != b); // Not Equal to
ans =(a >); //Greater than
ans (a<b); //Less than
ans = (a >= b); //Greater than or equal to
ans (a <- b); //Less than or Equal to
//Logical Operators
ans = (a > 0 && b > 0); //Logical AND
ans = (ae || b > 0); //Logical OR
ans = !(a> 0); //Logical NOT
}
}
```

Output:

The image shows the Solidity Compiler interface. On the left, the 'SOLIDITY COMPILER' section is active, displaying the compiler version '0.8.25+commit.b61c2a91'. Below this, there are checkboxes for 'Include nightly builds', 'Auto compile', and 'Hide warnings'. An 'Advanced Configurations' link is also present. At the bottom of this section are two buttons: 'Compile operators.sol' and 'Compile and Run script'. On the right, the 'CONTRACT' section shows the selected contract 'Operators - operators.sol'. Below the contract name, it indicates the 'evm version: cancun' and a 'Deploy' button. Below the compiler and contract sections, a window titled 'OPERATORS AT 0XD2A...FD005' is visible. This window shows the contract's state: 'Balance: 0 ETH', a button labeled 'ans', a variable '0: bool: false', a button labeled 'result', and a variable '0: uint256: 0'.

SOLIDITY COMPILER ✓ >

COMPILER +

0.8.25+commit.b61c2a91

☐ Include nightly builds

☐ Auto compile

☐ Hide warnings

Advanced Configurations >

Compile operators.sol

Compile and Run script

CONTRACT

Operators - operators.sol

evm version: cancun

Deploy

OPERATORS AT 0XD2A...FD005

Balance: 0 ETH

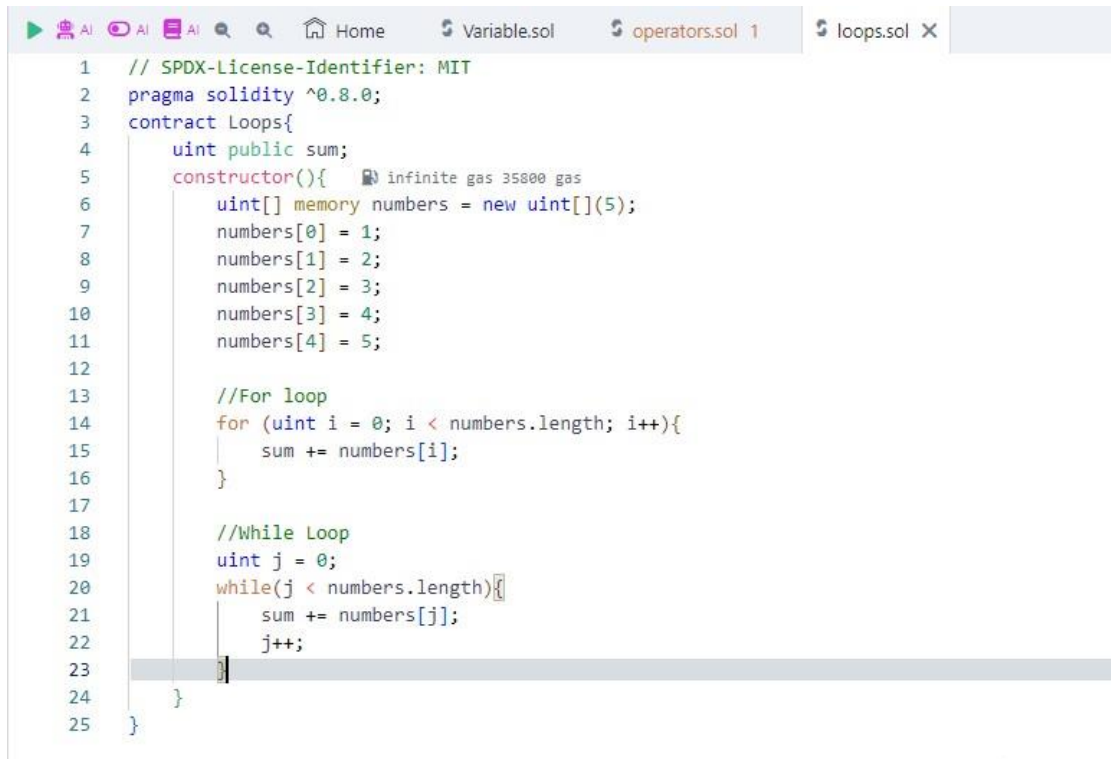
ans

0: bool: false

result

0: uint256: 0

3] Loops



```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3 contract Loops{
4     uint public sum;
5     constructor(){ infinite gas 35800 gas
6         uint[] memory numbers = new uint[](5);
7         numbers[0] = 1;
8         numbers[1] = 2;
9         numbers[2] = 3;
10        numbers[3] = 4;
11        numbers[4] = 5;
12
13        //For loop
14        for (uint i = 0; i < numbers.length; i++){
15            sum += numbers[i];
16        }
17
18        //While Loop
19        uint j = 0;
20        while(j < numbers.length){
21            sum += numbers[j];
22            j++;
23        }
24    }
25 }
```

Code:-

```
pragma solidity "0.8.0;
contract Loops{
uint public sum;
constructor(){
infinite gas 35800 gas
uint[] memory numbers new uint[](5);
numbers[0] = 1;
numbers[1] = 2;
numbers[2]= 3;
numbers[3]=4;
numbers[4]= 5;
//For loop
for (uint i = 0; i < numbers.length; i++) {
sum += numbers[1];
}
//While Loop
uint j=0;
```

```
while(j < numbers.length){  
  sum numbers[j];  
  j++;  
}  
}  
}
```

Output

The screenshot displays the Remix IDE interface. On the left, the 'Advanced Configurations' panel is open, showing a blue button 'Compile loops.sol' and a grey button 'Compile and Run script'. Below this, a console window shows the execution of the 'sum' function, resulting in '0: uint256: 30'. On the right, the 'CONTRACT' panel shows the selected contract 'Loops - contracts/loops.sol' and a 'Deploy' button. The EVM version is set to 'cancun'.

Advanced Configurations >

Compile loops.sol

Compile and Run script i

CONTRACT

Loops - contracts/loops.sol

evm version: cancan

Deploy

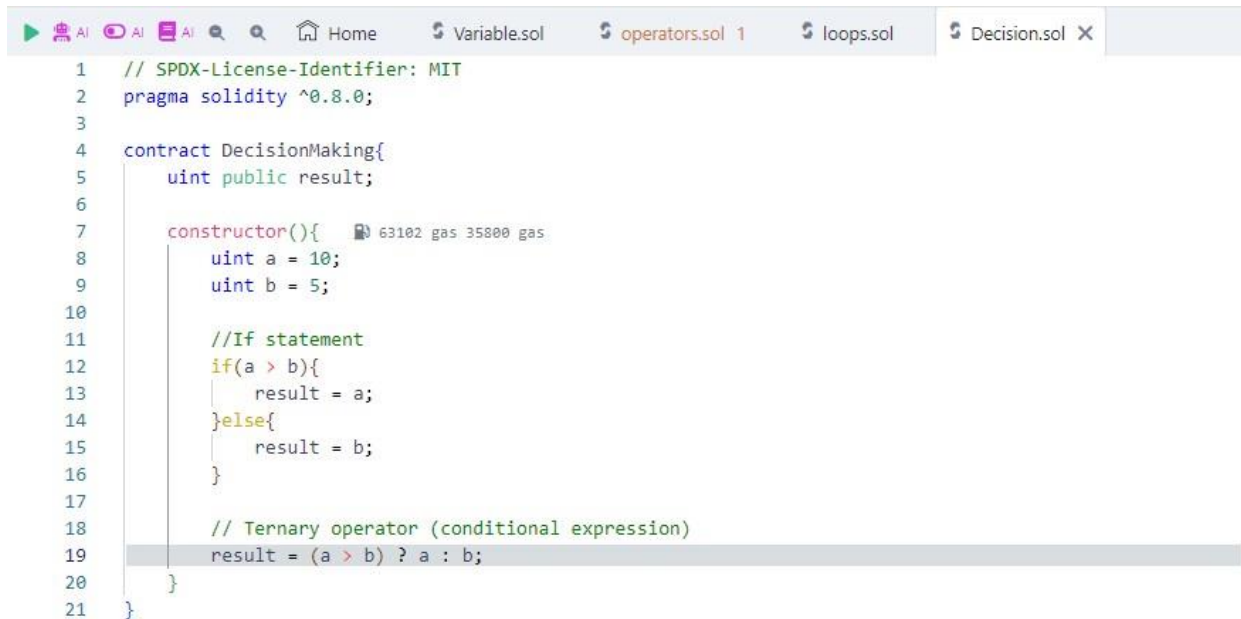
LOOPS AT 0XB27...07C2C (MEMC)

Balance: 0 ETH

sum

0: uint256: 30

4] Decision Making



```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract DecisionMaking{
5     uint public result;
6
7     constructor(){
8         uint a = 10;
9         uint b = 5;
10
11         //If statement
12         if(a > b){
13             result = a;
14         }else{
15             result = b;
16         }
17
18         // Ternary operator (conditional expression)
19         result = (a > b) ? a : b;
20     }
21 }
```

Code:-

```
// SPDX-License-Identifier: MIT
pragma solidity "0.8.0;
contract Decision Making{
uint public result;
constructor(){
uint a 10;
uint b = 5;
//If statement
if(a> b){
result = a;
}else{
result = b;
}

// Ternary operator (conditional expression)
result (ab)? a: b;
}
}
```


Output:

Advanced Configurations >

Compile Decision.sol

Compile and Run script i

CONTRACT

DecisionMaking - contracts/Decisior

evm version: cancun

Deploy

DECISIONMAKING AT 0XCD6...99

Balance: 0 ETH

result

0: uint256: 10

5] String

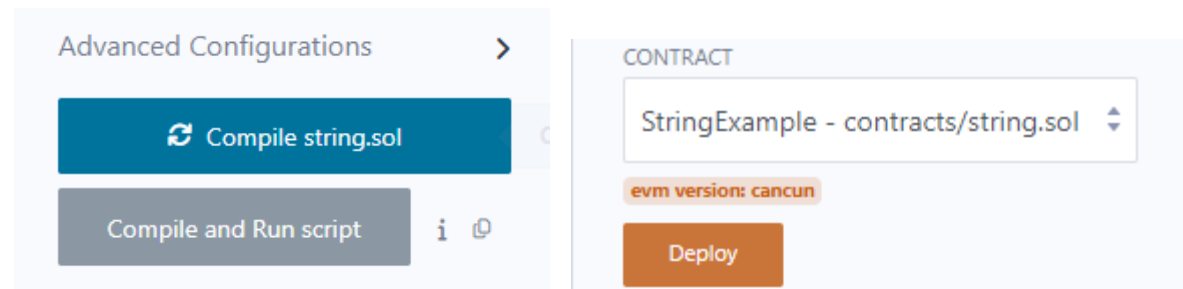
```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract StringExample{
5     string public myString;
6     string public substring;
7     bool public Comparing;
8     uint public length;
9
10    constructor(){
11        //Assigning a string
12        myString = "Hello, Solidity!";
13
14        //Concatenation
15        myString = string(abi.encodePacked(myString, "Welcome!"));
16
17        //Length of string
18        length = bytes(myString).length;
19
20        //Substring extraction
21        substring = substr(myString, 7, 8);
22
23        //Comparing strings
24        Comparing = compareStrings(myString, "Hello, Solidity! welcome");
25    }
26
27    function substr(string memory str, uint startIndex, uint length) internal pure returns (string memory){
28        bytes memory strBytes = bytes(str);
29        bytes memory result = new bytes(length);
30        for (uint i =0; i < length; i++){
31            result[i] = strBytes[startIndex + i];
32        }
33        return string(result);
34    }
35
36    function compareStrings(string memory str1, string memory str2) internal pure returns (bool){
37        return keccak256(abi.encodePacked(str1)) == keccak256(abi.encodePacked(str2));
38    }
39 }
40
41
42
```

Code:-

```
pragma solidity "0.8.0;
contract StringExample{
string public myString;
string public substring;
bool public Comparing;
uint public length;
constructor(){
//Assigning a string
myString = "Hello, Solidity!";
//Concatenation
myString string (abi.encodePacked (myString, "Welcome!"));
```

```
//Length of string
length bytes (myString).length;
//Substring extraction
substring = substr(myString, 7, 8);
//Comparing strings
Comparing compareStrings (myString, "Hello, Solidity! welcome");
```

Output:



▼

STRINGEXAMPLE AT 0X9D8...A56

×

Balance: 0 ETH

Comparing

0: bool: false

length

0: uint256: 24

myString

0: string: Hello, Solidity!Welcome!

substring

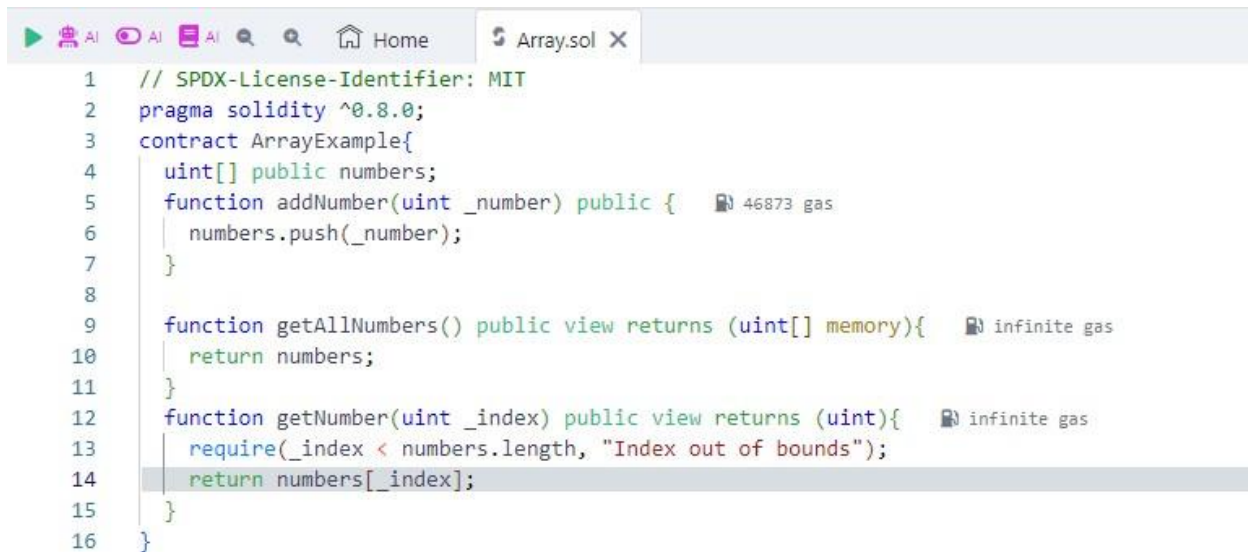
0: string: Solidity

Practical No:04

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

- 1] Arrays**
- 2] Enums**
- 3] Structs**
- 4] Mappings**
- 5] Ether units**
- 6] Special Variables**

1] Arrays



```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3 contract ArrayExample{
4     uint[] public numbers;
5     function addNumber(uint _number) public { 46873 gas
6         numbers.push(_number);
7     }
8
9     function getAllNumbers() public view returns (uint[] memory){ infinite gas
10        return numbers;
11    }
12    function getNumber(uint _index) public view returns (uint){ infinite gas
13        require(_index < numbers.length, "Index out of bounds");
14        return numbers[_index];
15    }
16 }
```

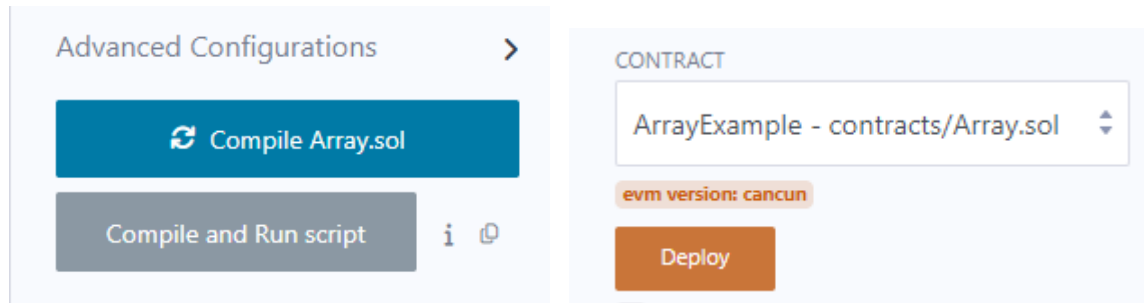
Code:-

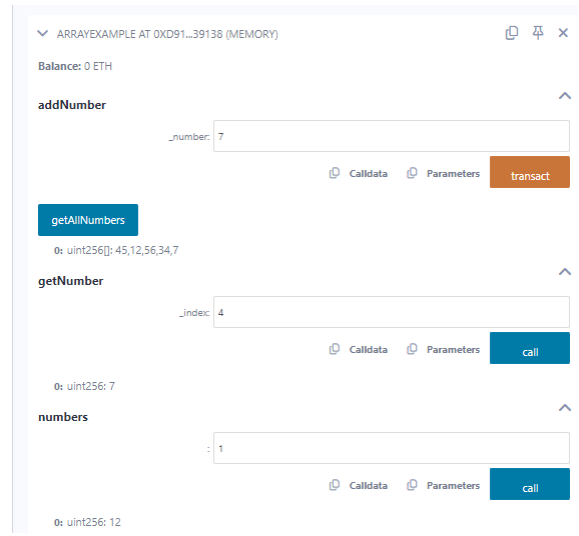
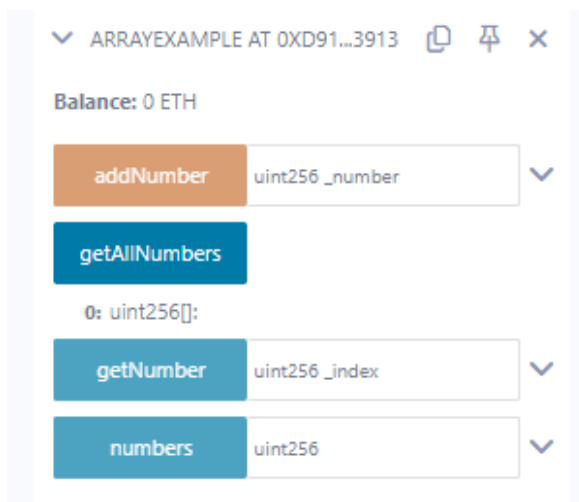
```
pragma solidity "0.8.0;
contract ArrayExample{
uint[] public numbers;
function addNumber (uint _number) public {
numbers.push(_number);
}

function getAllNumbers() public view returns (uint[] memory){
return numbers;
}
function getNumber (uint _index) public view returns (uint){
require(_index < numbers.length, "Index out of bounds");
return numbers[_index];
}
```

}

Output:





2] Enum

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract EnumExample{
5     enum Gender {Male, Female, Other}
6     enum Color {Red, Green, Blue, Yellow, Orange, Purple}
7
8     Gender public personGender;
9     Color public objectColor;
10
11     function setGender(uint8 _gender) public { 24839 gas
12         require(_gender <= uint8(Gender.Other), "Invalid gender");
13         personGender = Gender(_gender);
14     }
15     function setColor(uint8 _color) public { 24845 gas
16         require(_color <= uint8(Color.Purple), "Invalid Color");
17         objectColor = Color(_color);
18     }
19 }

```

```

pragma solidity ^0.8.0;
contract EnumExample{
enum Gender {Male, Female, Other}
enum Color {Red, Green, Blue, Yellow, Orange, Purple}
Gender public personGender;
Color public objectColor;
function setGender (uint8 _gender) public {

```

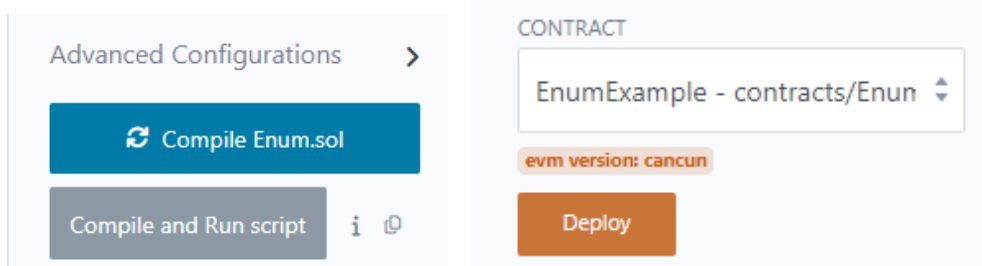


```
require(_gender <= uint8 (Gender. Other), "Invalid gender");

personGender Gender (_gender);
}
function setColor(uint8 _color) public {
require(_color<= uint8 (Color. Purple), "Invalid Color");
objectColor=Color(_color);

}
```

Output:



Balance: 0 ETH

setColor

uint8 _color

setGender

uint8 _gender

objectColor

personGender

ENUMEXAMPLE AT 0XB27...07C2C (MEMORY)

Balance: 0 ETH

setColor

_color: 3

Calldata Parameters

transact

setGender

_gender: 1

Calldata Parameters

transact

objectColor

0: uint8: 3

personGender

personGender - call

0: uint8: 1

3] Structs

```

1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.0;
3  contract StructExample{
4      struct Person{
5          string name;
6          uint age;
7      }
8
9      Person public myPerson;
10
11     function setPerson(string memory _name, uint _age) public {
12         myPerson = Person(_name,_age);
13     }
14
15     function getPerson() public view returns (string memory,uint){
16         return (myPerson.name, myPerson.age);
17     }
18 }

```

Code:-

```

pragma solidity "0.8.0;
contract StructExample{
    struct Person{
        string name;
        uint age;
    }
    Person public myPerson;

```

```
function setPerson(string memory _name, uint _age) public {  
  
    myPerson = Person(_name,_age);  
}  
function getPerson() public view returns (string memory, uint){  
    return (myPerson.name, myPerson.age);  
}  
}
```

Output:

The screenshot shows a web interface for managing Solidity contracts. On the left, under 'Advanced Configurations', there are two buttons: 'Compile Stuct.sol' (blue) and 'Compile and Run script' (grey). On the right, under 'CONTRACT', there is a dropdown menu showing 'StructExample - contracts/Stuct.so'. Below the dropdown, a status bar indicates 'evm version: cancun'. At the bottom right is a 'Deploy' button.

Balance: 0 ETH

setPerson
string _name, uint256 _age

getPerson

myPerson

STRUCTEXAMPLE AT 0xE28...4157A (MEMORY) 📄 📌 ✕

Balance: 0 ETH

setPerson ^

_name:

_age:

📄 Calldata
📄 Parameters
transact

getPerson

0: string: Mansi
1: uint256: 23

myPerson

myPerson - call

0: string: name Mansi
1: uint256: age 23

4) Mappings

```

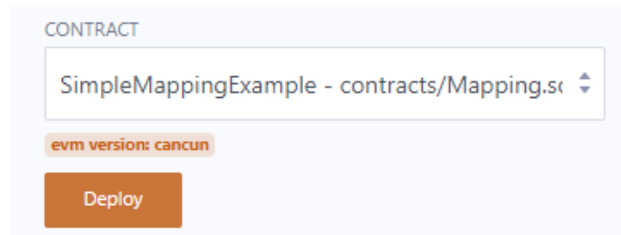
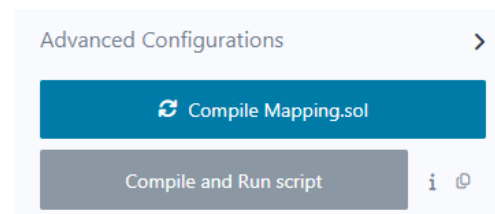
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.0;
3  contract SimpleMappingExample{
4      mapping (uint => bool) public myMapping;
5
6      function setValue(uint _index) public { 24729 gas
7          myMapping[_index] = true;
8      }
9
10 }
```




Code:-

```

pragma solidity "0.8.0;
contract SimpleMappingExample{
mapping (uint => bool) public myMapping;
function setValue(uint _index) public {
myMapping[_index] = true;
}
}
```

Output:



▼ SIMPLEMAPPINGEXAMPLE AT 0XC3B...AAECA (I)   

Balance: 0 ETH

setValue




uint256 _index

▼

myMapping

uint256

▼


▼ SIMPLEMAPPINGEXAMPLE AT 0XC3B...AAECA (I)   


Balance: 0 ETH

setValue

_index:

uint256

 Calldata


 Parameters


transact

myMapping

:




2

 Calldata

 Parameters

call

0: bool: false


▼ SIMPLEMAPPINGEXAMPLE AT 0XC3B...AAECA (I)   


Balance: 0 ETH

setValue

_index:

2

 Calldata


 Parameters


transact

myMapping

:




2

 Calldata

 Parameters

call

0: bool: false


▼ SIMPLEMAPPINGEXAMPLE AT 0XC3B...AAECA (I)   


Balance: 0 ETH

setValue

_index:

2

 Calldata


 Parameters


transact

myMapping

:

2

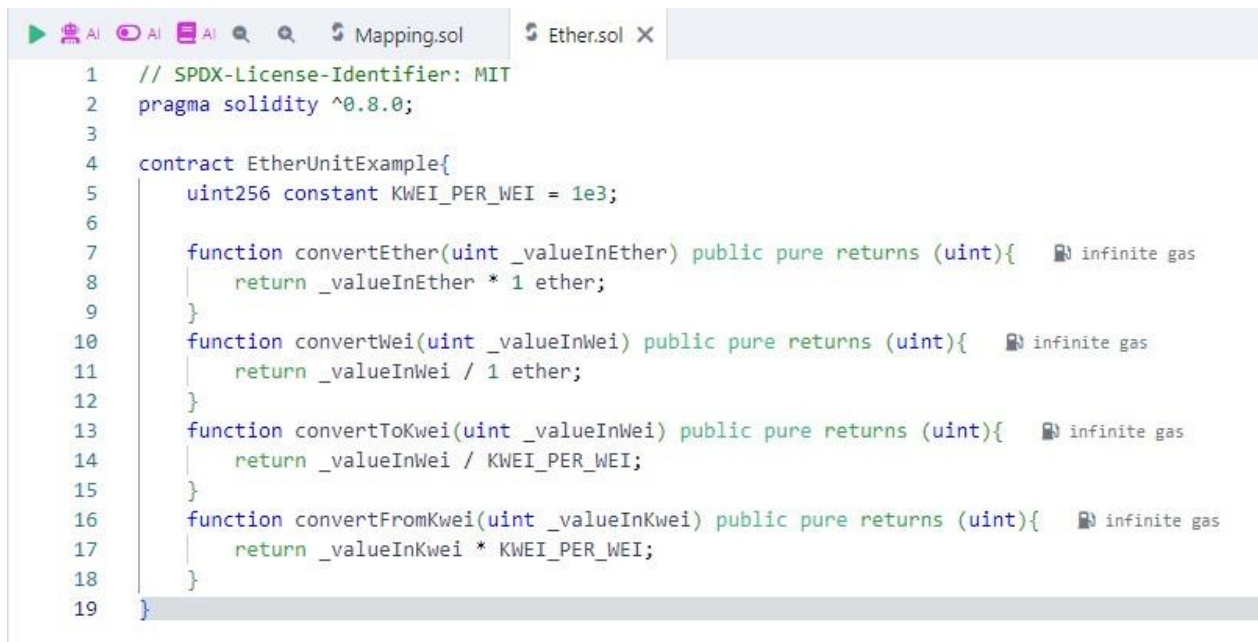
 Calldata

 Parameters

call

0: bool: true

5] Ether units

A screenshot of a Solidity code editor. The editor has a tab bar at the top with 'Mapping.sol' and 'Ether.sol'. The 'Ether.sol' tab is active. The code is as follows:

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract EtherUnitExample{
5     uint256 constant KWEI_PER_WEI = 1e3;
6
7     function convertEther(uint _valueInEther) public pure returns (uint){ infinite gas
8         return _valueInEther * 1 ether;
9     }
10    function convertWei(uint _valueInWei) public pure returns (uint){ infinite gas
11        return _valueInWei / 1 ether;
12    }
13    function convertToKwei(uint _valueInWei) public pure returns (uint){ infinite gas
14        return _valueInWei / KWEI_PER_WEI;
15    }
16    function convertFromKwei(uint _valueInKwei) public pure returns (uint){ infinite gas
17        return _valueInKwei * KWEI_PER_WEI;
18    }
19 }
```

Code:-

```
pragma solidity "0.8.0;
contract EtherUnitExample{
uint256 constant KWEI_PER_WEI = 1e3;
function convertEther (uint _valueInEther) public pure returns (uint){
    return _valueInEther * 1 ether;
}
function convertwei (uint value Inwei) public pure returns (uint) {
    return _valueInwei / 1 ether;
}
function convertTokwei (uint valueInwei) public pure returns (uint) {
    return _valueInwei / KWEI_PER_WEI;
}
function convertFromKwei (uint _value Inkwei) public pure returns (uint) {
    return _valueInKwei KWEI_PER_WEI;
}
}
```

Output:

Advanced Configurations



 Compile Ether.sol

Compile and Run script



CONTRACT

EtherUnitExample - contracts/Ether.sol



evm version: cancan

Deploy

▼ ETHERUNITEXAMPLE AT 0X081...A0F85 (MEMO)   

Balance: 0 ETH

convertEther uint256 _valueInEther ▼

convertFromK... uint256 _valueInKwei ▼

convertToKwei uint256 _valueInWei ▼

convertWei uint256 _valueInWei ▼

ETHERUNITEXAMPLE AT 0X081...A0F85 (MEMORY)

Balance: 0 ETH

convertEther

_valueInEther:

0: uint256: 2000000000000000000

convertFromKwei

_valueInKwei:

0: uint256: 1000

convertToKwei

_valueInWei:

0: uint256: 0

convertWei

_valueInWei:

0: uint256: 0

6] Special Variables

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3 contract SpecialVariableExample{
4     address public owner;
5
6     constructor(){ 142831 gas 118400 gas
7         owner = msg.sender;
8     }
9     function getMessageSender() public view returns (address){ 385 gas
10        return msg.sender;
11    }
12    function getContractBalance() public view returns (uint){ 317 gas
13        return address(this).balance;
14    }
15    function isOwner() public view returns (bool){ 2540 gas
16        return msg.sender == owner;
17    }
18 }
19 }

```

Code:-

```
pragma solidity ^0.8.0;
contract SpecialVariableExample{
address public owner;
constructor(){ 142831 gas 118400 gas
owner msg.sender;
}
function getMessageSender() public view returns (address) {
return msg.sender;
}
function getContractBalance() public view returns (uint){
return address (this).balance;
}
function isOwner() public view returns (bool) {
return msg.sender == owner;
}
}
```

Output:

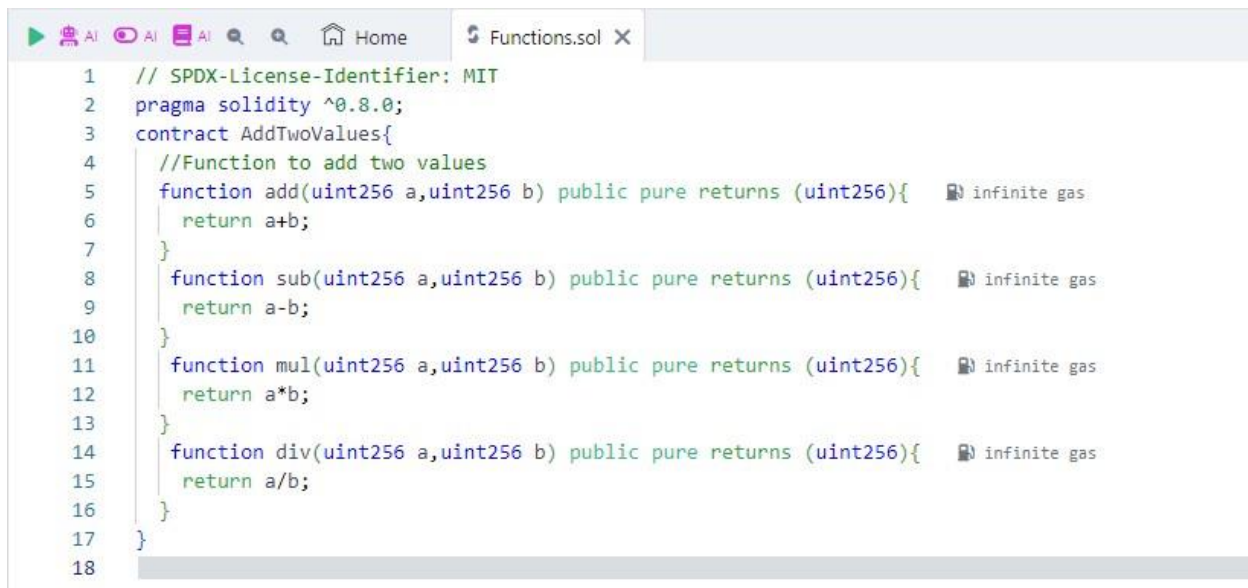
The screenshot displays the Remix IDE interface. On the left, the 'Advanced Configurations' panel shows a 'Compile SpecialVari.sol' button and a 'Compile and Run script' button. The main editor shows the Solidity code for the 'SpecialVariableExample' contract. On the right, the 'CONTRACT' panel shows the contract name 'SpecialVariableExample - contracts/SpecialVari.sol' and the EVM version 'cancun'. Below this, there is a 'Deploy' button. The bottom panel shows the contract's state after deployment. It displays the contract's address '0x5B38Da6a701c568545dCfcB03FcB875f56beddC' and its balance '0 ETH'. The contract's state variables are shown: 'owner' is '0x5B38Da6a701c568545dCfcB03FcB875f56beddC'. The contract's methods are listed: 'getContractBal...', 'getMessageSe...', 'isOwner', and 'owner'. The 'owner' method is highlighted, showing its return value '0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC'.

Practical No:05

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

- 1] Functions**
- 2] View functions**
- 3] Pure functions**
- 4] Fallback functions**
- 5] Function Overloading**
- 6] Mathematical functions**
- 7] Cryptographic functions**

1] Functions



```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3 contract AddTwoValues{
4     //Function to add two values
5     function add(uint256 a,uint256 b) public pure returns (uint256){ infinite gas
6         return a+b;
7     }
8     function sub(uint256 a,uint256 b) public pure returns (uint256){ infinite gas
9         return a-b;
10    }
11    function mul(uint256 a,uint256 b) public pure returns (uint256){ infinite gas
12        return a*b;
13    }
14    function div(uint256 a,uint256 b) public pure returns (uint256){ infinite gas
15        return a/b;
16    }
17 }
18
```

Code:-

```
pragma solidity "0.8.0;
contract AddTwoValues{
//Function to add two values
function add (uint256 a,uint256 b) public pure returns (uint256) {
return a+b;
}
function sub(uint256 a, uint256 b) public pure returns (uint256) {
return a-b;
}
function mul (uint256 a, uint256 b) public pure returns (uint256){
return a*b;
```

```
}  
function div(uint256 a, uint256 b) public pure returns (uint256) {  
    return a/b;  
}  
}
```

Output:

The screenshot displays two panels from a Solidity development interface. The left panel, titled "Advanced Configurations", contains a blue button labeled "Compile Functions.sol" with a refresh icon, and a grey button labeled "Compile and Run script" with an information icon and a share icon. The right panel, titled "CONTRACT", features a dropdown menu showing "AddTwoValues - contracts/Functions.sol", a status indicator "evm version: cancun", and an orange "Deploy" button.

ADDTWOVALUES AT 0XF8E...9FBEB (MEM)

Balance: 0 ETH

add

uint256 a, uint256 b

div

uint256 a, uint256 b

mul

uint256 a, uint256 b

sub

uint256 a, uint256 b

ADDTWOVALUES AT 0XF8E...9FBEB (MEMORY)

Balance: 0 ETH

add

a: 5

b: 4

CalldataParameterscall

0: uint256: 9

div

a: 10

b: 5

CalldataParameterscall

0: uint256: 2

mul

a: 4

b: 2

CalldataParameterscall

0: uint256: 8

sub

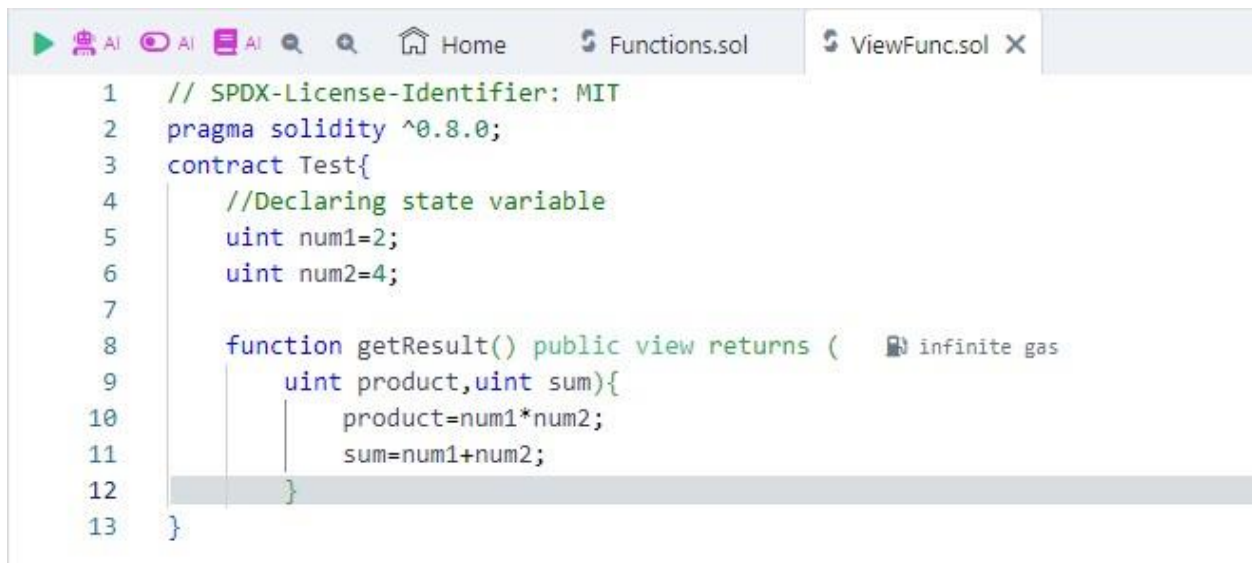
a: 5

b: 2

CalldataParameterscall

0: uint256: 3

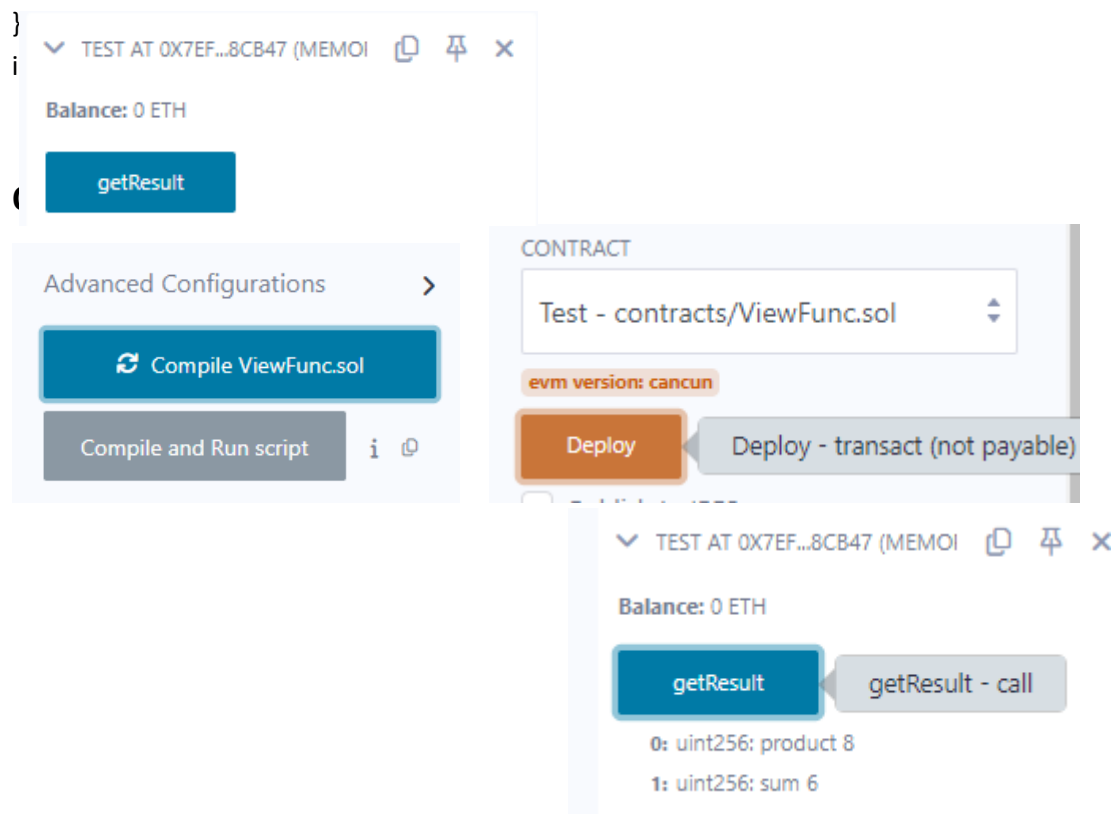
2] View functions - Read Only



```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3 contract Test{
4     //Declaring state variable
5     uint num1=2;
6     uint num2=4;
7
8     function getResult() public view returns ( uint product, uint sum){
9         product=num1*num2;
10        sum=num1+num2;
11    }
12 }
13 }
```

Code:-

```
pragma solidity 0.8.0;
contract Test{
uint num1=2;
uint num2=4;
function getResult() public view returns ( uint product, uint sum){
product=num1*num2;
Sum=num1+num2;
}
}
```



TEST AT 0X7EF...8CB47 (MEMOI) [Copy] [Share] [Close]

Balance: 0 ETH

getResult

Advanced Configurations >

Compile ViewFunc.sol

Compile and Run script [Info] [Share]

CONTRACT

Test - contracts/ViewFunc.sol

evm version: cancan

Deploy [Deploy - transact (not payable)]

TEST AT 0X7EF...8CB47 (MEMOI) [Copy] [Share] [Close]

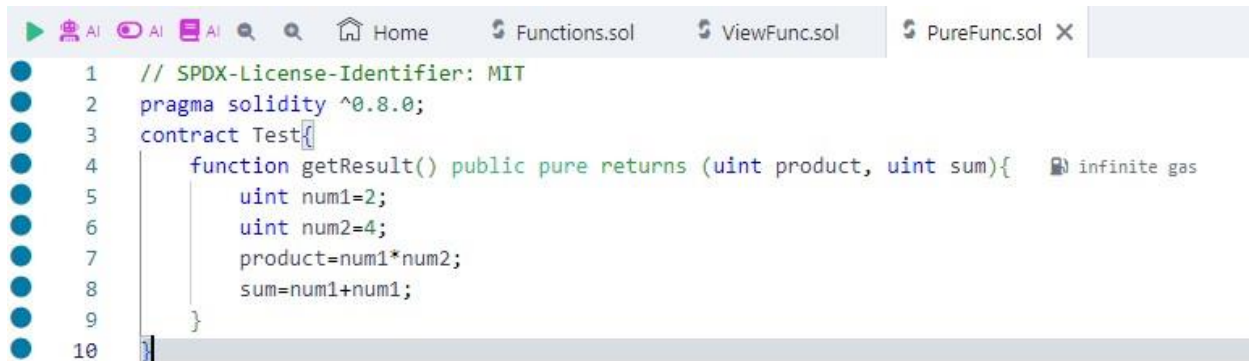
Balance: 0 ETH

getResult [getResult - call]

0: uint256: product 8

1: uint256: sum 6

3] Pure Functions

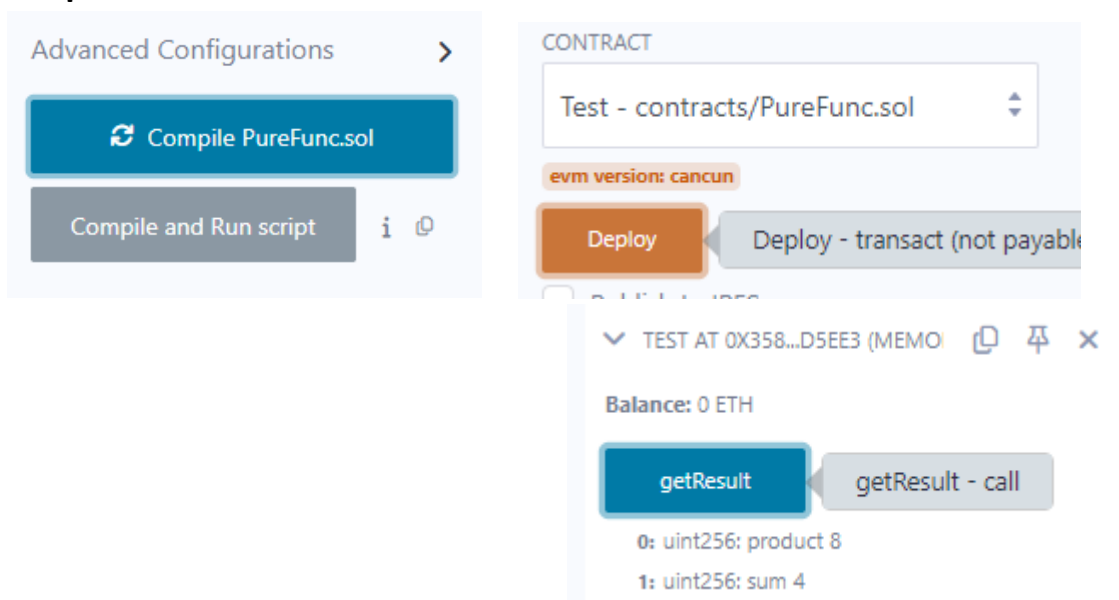


```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3 contract Test{
4     function getResult() public pure returns (uint product, uint sum){ infinite gas
5         uint num1=2;
6         uint num2=4;
7         product=num1*num2;
8         sum=num1+num2;
9     }
10 }
```

Code:-

```
pragma solidity ^0.8.0;
contract Test{
function getResult() public pure returns (uint product, uint sum){
uint num1=2;
uint num2=4;
product=num1*num2;
sum=num1+num2;
}
}
```

Output:



Advanced Configurations >

Compile PureFunc.sol

Compile and Run script i

CONTRACT

Test - contracts/PureFunc.sol

evm version: cancun

Deploy Deploy - transact (not payable)

TEST AT 0X358...D5EE3 (MEMO)

Balance: 0 ETH

getResult getResult - call

0: uint256: product 8

1: uint256: sum 6

[getResult](#)

4] Fallback function

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3 contract FallbackDemo{
4     //Fallback function
5     fallback() external payable { undefined gas
6         //Log the fact that fallback function was called
7         emit FallbackCalled(msg.sender,msg.value);
8     }
9
10    //Event to log when the fallback function is called
11    event FallbackCalled(address caller,uint256 value);
12 }
```

Code:-

```
pragma solidity 0.8.0;
contract FallbackDemo{
    fallback() external payable {
        emit FallbackCalled (msg.sender, msg.value);
    }
    event FallbackCalled (address caller, uint256 value);
}
```

Output:

Advanced Configurations >

Compile FallbackFunc.sol

Compile and Run script

CONTRACT

FallbackDemo - contracts/Fallback

evm version: cancun

Deploy

FALLBACKDEMO AT 0XDDA...54

Balance: 0 ETH

5] Function Overloading

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3 contract FunctionDemo{
4     function add(uint256 a, uint256 b) public pure returns (uint256){
5         return a + b;
6     }
7     function add(uint256 a, uint256 b, uint256 c) public pure returns (uint256){
8         return a + b + c;
9     }
10 }

```

Code:-

```

pragma solidity 0.8.0;
contract FunctionDemo{
function add(uint256 a, uint256 b) public pure returns (uint256){
return a + b;
}
function add(uint256 a, uint256 b, uint256 c) public pure returns (uint256) {
return a + b + c;
}
}

```

Output:

Advanced Configurations >

Compile OverloadingFun.sol

Compile and Run script

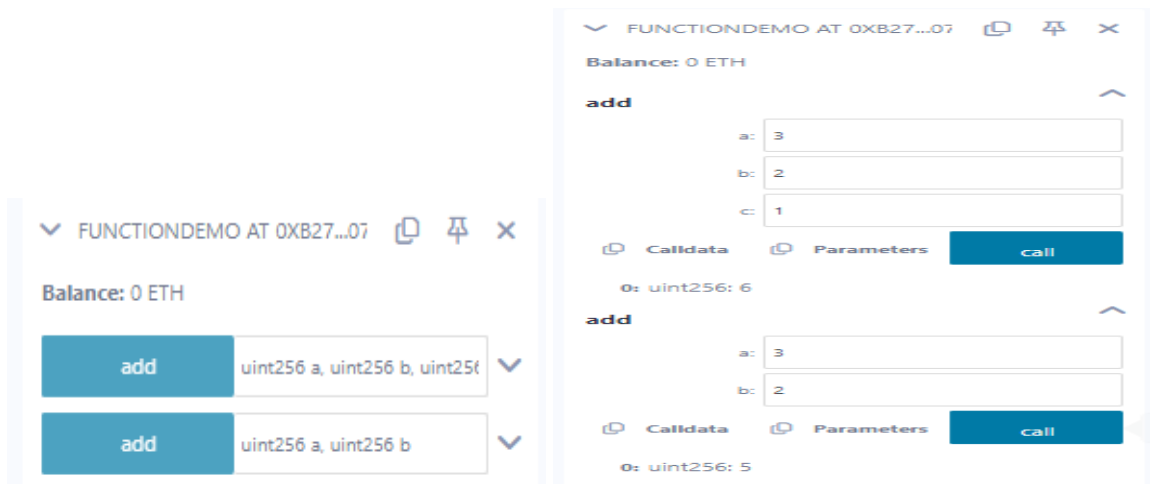
CONTRACT

FunctionDemo - contracts/Overloa

evm version: cancun

Deploy

Deploy - transact (not payable)



6] Mathematical functions


```



1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.0;
3
4  contract MathFunctions{
5      //Function to calculate the square of a number
6      function square(uint256 x) public pure returns (uint256){ infinite gas
7          return x * x;
8      }
9
10     //Function to calculate the cube of a number
11     function cube(uint256 x) public pure returns (uint256){ infinite gas
12         return x * x * x;
13     }
14
15     //Function to calculate the factorial of a number
16     function factorial(uint256 n) public pure returns(uint256){ infinite gas
17         uint256 result = 1;
18         for (uint256 i =2;i<=n;i++){
19             result *=i;
20         }
21         return result;
22     }
23
24     //Function to calculate the nth fibonacci number
25     function fibonacci(uint256 n) public pure returns(uint256){ infinite gas
26         if(n==0) return 0;
27         uint256 a = 0;
28         uint256 b = 1;
29         for(uint256 i = 2; i<=n;i++){
30             (a,b)=(b,a+b);
31         }
32         return b;
33     }
34 }

```

Output:

Advanced Configurations >

 Compile MathematicalFun.sol



Compile and Run script  

CONTRACT


MathFunctions - contracts/Mathen


evm version: cancun


Deploy


MATHFUNCTIONS AT 0XAE0...9   X


Balance: 0 ETH

cube uint256 x 



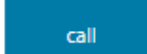
factorial uint256 n 

fibonacci uint256 n 


square uint256 x 

fibonacci 



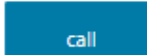
n:

 Calldata  Parameters 



0: uint256: 13

square 


x:

 Calldata  Parameters 



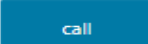
0: uint256: 9

MATHFUNCTIONS AT 0XAE0...9   X


Balance: 0 ETH

cube 




x:

 Calldata  Parameters 

0: uint256: 8

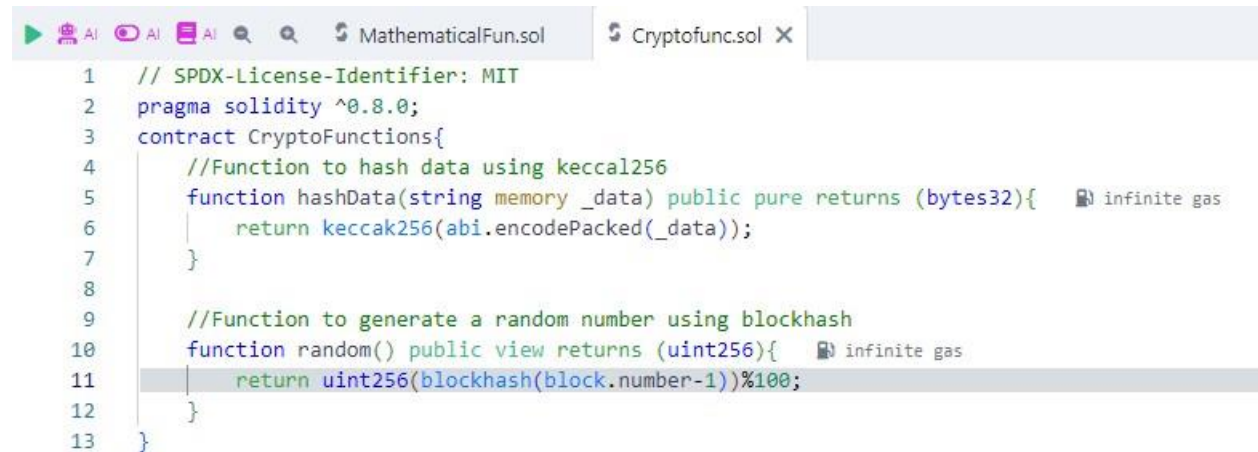
factorial 

n:

 Calldata  Parameters 

0: uint256: 120

7] Cryptographic functions



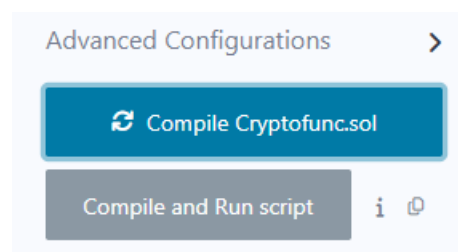
```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3 contract CryptoFunctions{
4     //Function to hash data using keccak256
5     function hashData(string memory _data) public pure returns (bytes32){ infinite gas
6         return keccak256(abi.encodePacked(_data));
7     }
8
9     //Function to generate a random number using blockhash
10    function random() public view returns (uint256){ infinite gas
11        return uint256(blockhash(block.number-1))%100;
12    }
13 }
```

Code:-

```
pragma solidity "0.8.0;
contract CryptoFunctions{
function hashData (string memory _data) public pure returns (bytes32){
return keccak256 (abi.encodePacked(_data));
}
function random() public view returns (uint256) {
return uint256(blockhash (block.number-1))%100;

}
}
```

Output:



Practical No: 06

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

- 1] Contracts**
- 2] Inheritance**
- 3] Abstract Class**

1] Contracts

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.2;
3
4  //base contract
5  contract Shape{
6      uint internal area;
7
8      //set area function
9      function setArea(uint _area) external{ 22520 gas
10         area = _area;
11     }
12
13     //get area function
14     function getArea() external view returns(uint){ 2415 gas
15         return area;
16     }
17 }
18 //derived contract
19 contract Circle is Shape{
20     uint internal radius;
21
22     //constructor to set the radius
23     constructor(uint _radius){ infinite gas 101200 gas
24         radius = _radius;
25     }
26
27     //function to calculate the area of a circle
28     function calculateArea() external{ infinite gas
29         area = 3* radius * radius; //Approximation of area for simplicity
30     }
```

Code:-

```
pragma solidity ^0.8.2;
//base contract contract Shape{
uint internal area;
//set area function
function setArea (uint area) external{
area = _area;
```

```

}
22520 gas
//get area function
function getArea() external view returns (uint) {
return area;
}
}
2415 gas
//derived contract
contract Circle is Shape
uint internal radius;
//constructor to set the radius
constructor (uint _radius) { infinite gas 101200 gas radius_radius;
}
//function to calculate the area of a circle
function calculateArea() external{ infinite gas
}
area 3* radius radius; //Approximation of area for simplicity

```

Output:

Advanced Configurations >

↻ Compile contract.sol

Compile and Run script i

CONTRACT

Circle - contracts/contract.sol

evm version: cancun

Deploy uint256 _radius

CIRCLE AT 0X7EF...8CB47 (MEMO)

Balance: 0 ETH

calculateArea

setArea

_area: uint256

Calldata Parameters transact

getArea

getArea - call

0: uint256: 75

DEPLOY

_RADIUS: 5

Calldata Parameters transact

Balance: 0 ETH

calculateArea

setArea

_area: 43

Calldata Parameters transact

getArea

getArea - call

0: uint256: 43

2] Inheritance

```
Pract 7 inheritance.sol

4  //base contract
5  contract Person{
6      string internal name;
7
8      constructor(string memory _name){  infinite gas 110400 gas
9          name = _name;
10     }
11
12     //function to get the person's name
13     function getName() external view returns (string memory){  infinite gas
14         return name;
15     }
16 }
17
18
19 //derived contract for employee
20 contract Employee is Person{
21
22     string internal position;
23
24     constructor(string memory _name, string memory _position) Person(_name){  infinite gas 147800 gas
25         position = _position;
26     }
27
28     //function to get the person's name
29     function getPosition() external view returns (string memory){  infinite gas
30         return position;
31     }
32 }
33
34 //derived contract for student
35 contract Student is Person{
36
37     uint internal grade;
38
39     constructor(string memory _name, uint _grade) Person(_name){  infinite gas 131000 gas
40         grade = _grade;
41     }
42
43     //function to get the students's grade
44     function getGrade() external view returns (uint){  2437 gas
45         return grade;
46     }
47 }
```

Code:-

```
<
contract Person{
string internal name;
```

```

constructor(string memory _name) { name = _name;
infinite gas 110400 gas
}
//function to get the person's name
function getName() external view returns (string memory){
return name;
}
}
//derived contract for employee
contract Employee is Person{
string internal position;
constructor(string memory _name, string memory position) Person(_name){
position position;

}
function getPosition() external view returns (string memory){
return position;
}
}
//derived contract for student
contract Student is Person{
uint internal grade;
constructor(string memory _name, uint grade) Person (_name){
grade = _grade;
}
//function to get the students's grade
function getGrade() external view returns (uint){
return grade;
}
}

```

Output:

Advanced Configurations >

Compile Pract 7 inheritance.sol

Compile and Run script

CONTRACT

Employee - Pract 7 inheritance.sol

evm version: cancun

DEPLOY

_NAME: Jhon

_POSITION: CEO

Calldata Parameters

EMPLOYEE AT 0X5FD...9D88D (MI)

Balance: 0 ETH

getName

0: string: Jhon

getPosition

0: string: CEO

CONTRACT

Student - Pract 7 inheritance.sol

evm version: cancun

DEPLOY

_NAME: Jhon

_grade: A

Calldata Parameters

STUDENT AT 0X7B9...B6ACE (ME)

Balance: 0 ETH

getGrade

0: string: A

getName

0: string: jhon

3] Abstract Class

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 //abstract contract for shape
5 abstract contract Shape{
6     //abstract contract to calculate area
7     function calculateArea() external view virtual returns (uint); - gas
8 }
9
10 //concrete contract for circle
11 contract Circle is Shape{
12
13     uint internal radius;
14
15     //constructor to set the radius
16     constructor(uint _radius){ infinite gas 61400 gas
17         radius = _radius;
18     }
19
20     //implementation of abstract function
21     function calculateArea() external view override returns (uint){ infinite gas
22         return 3 * radius * radius;//approximation of area for simplicity
23     }
24 }
```

Code:-

```
pragma solidity ^0.8.0;
//abstract contract for shape.
abstract contract Shape{
function calculateArea() external view virtual returns (uint);

}
//concrete contract for circle
contract Circle is Shape{
    uint internal radius;
//constructor to set the radius
constructor (uint _radius) { infinite gas 61400 gas
radius = _radius;
}
//implementation of abstract function
function calculateArea() external view override returns (uint){
return 3* radius radius;//approximation of area for simplicity
}
}
```

Output:

Advanced Configurations >

Compile Abstract class.sol

Compile and Run script i

DEPLOY ^

_RADIUS: uint256

Calldata Parameters transact

✓ CIRCLE AT 0XDDA...5482D (MEM) [copy] [pin] [close]

Balance: 0 ETH

calculateArea ← calculateArea - call

0: uint256: 27

Practical No: 07

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

- 1] Libraries**
- 2] Events**
- 3] Error Handling**

1] Libraries

```
Pract8 Libraries.sol X
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 library MathLib{
5     //function to add two numbers
6     function add(uint256 a , uint256 b) internal pure returns (uint256){ infinite gas
7         return a+b;
8     }
9
10    //function to subtract two numbers
11    function sub(uint256 a , uint256 b) internal pure returns (uint256){ infinite gas
12        require(a>=b, "Subtraction result would be negative");
13        return a-b;
14    }
15
16    //function to subtract two numbers
17    function mul(uint256 a , uint256 b) internal pure returns (uint256){ infinite gas
18        return a*b;
19    }
20
21    //function to subtract two numbers
22    function div(uint256 a , uint256 b) internal pure returns (uint256){ infinite gas
23        require(b != 0, "Division by zero");
24        return a/b;
25    }
26 }
27
28 contract MathContract{
29     using MathLib for uint256;
30
31     //fnction to perform arithmetic operations using the library
32     function operate(uint256 a, uint256 b) external pure returns (uint256, uint256, uint256, uint256){
33         uint256 addition = a.add(b);
34         uint256 subtraction = a.sub(b);
35         uint256 multiplication = a.mul(b);
36         uint division = a.div(b);
37         return (addition,subtraction,multiplication,division);
38     }
39 }
```

Output:

Advanced Configurations >

Compile Pract8 Libraries.sol

Compile and Run script

CONTRACT

MathContract - contracts/Pract8 Lib

evm version: cancun

Deploy

MATHCONTRACT AT 0XB27...07C

Balance: 0 ETH

operate

a: 45

b: 6

Calldata

Parameters

call

0: uint256: 51

1: uint256: 39

2: uint256: 270

3: uint256: 7

2] Events

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract EventExample{
5     event ValueSet(address indexed sender, uint newValue);
6
7     uint public value;
8
9     //function to set a new value and trigger an event
10    function setValue(uint _newValue) external{
11        value = _newValue;
12        emit ValueSet(msg.sender, _newValue);
13    }
14 }

```

Code:-

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract EventExample{
    event ValueSet(address indexed sender, uint newValue);
    uint public value;
    //function to set a new value and trigger an event
    function setValue(uint _newValue) external{
        value = _newValue;
        emit ValueSet (msg.sender, _newValue);
    }
}
```

Output:

Advanced Configurations >

Compile Prcat8Events.sol

Compile and Run script

i

🔗

CONTRACT

EventExample - contracts/Prcat8Ever

evm version: cancun

Deploy

▼ EVENTEXAMPLE AT 0X9D8...A569

🔗 🛠️ ✕

Balance: 0 ETH

setValue

_newValue:

35

🔗 Calldata

🔗 Parameters

transact

value

0: uint256: 35

3] Error Handling

▶

AI

🔍

AI

📄

AI

🔍

🔍

🔄

Pract 7 inheritance.sol

Pract8ErrorHandling.sol ✕

```

1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.0;
3  contract ErrorHandling {
4      mapping(address => uint256) public balances;
5      function withdraw(uint256 amount) public {
6          // Check if the sender has sufficient balance
7          require(amount <= balances[msg.sender], "Insufficient balance");
8          // If balance is sufficient, deduct the amount
9          balances[msg.sender] -= amount;
10         // Perform the withdrawal operation
11         // (In a real contract, this would involve transferring funds)
12     }
13 }

```

Code:-

```
pragma solidity ^0.8.0;
contract ErrorHandling {
    mapping (address => uint256) public balances;
    function withdraw(uint256 amount) public {
        // Check if the sender has sufficient balance
        infinite gas
        require (amount <= balances [msg.sender], "Insufficient balance");
        // If balance is sufficient, deduct the amount
        balances [msg.sender] -= amount;
        // Perform the withdrawal operation
        // (In a real contract, this would involve transferring funds)
    }
}
```

Output:

Advanced Configurations >

Compile Pract8ErrorHandling.sol

Compile and Run script i

CONTRACT

ErrorHandling - contracts/Pract8Errc

evm version: cancun

Deploy

ERRORHANDLING AT 0XE28...415

Balance: 0 ETH

withdraw uint256 amount

balances address

ERRORHANDLING AT 0XE28...415

Balance: 0 ETH

withdraw

amount: 10000

Calldata Parameters transact

balances

: 5000

Calldata Parameters call

transact to ErrorHandling.withdraw errored: Error occurred: revert.

revert

The transaction has been reverted to the initial state.

Reason provided by the contract: "Insufficient balance".

You may want to cautiously increase the gas limit if the transaction went out of gas.

call to ErrorHandling.balances errored: Error encoding arguments: Error: invalid address (argument="address", value="5000", code=INVALID_ARGUMENT, version=ad