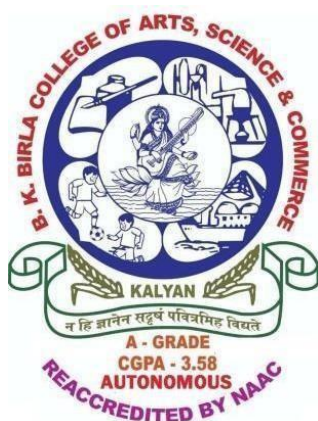


**B. K. BIRLA COLLEGE OF ARTS, SCIENCE & COMMERCE  
(AUTONOMOUS), KALYAN**

**DEPARTMENT OF INFORMATION TECHNOLOGY**



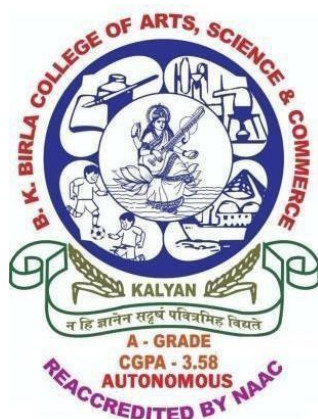
<b>Student Name:</b>	VIVEK PANDIT
<b>Student ID:</b>	4839747
<b>Class:</b>	MSC IT CC
<b>Subject:</b>	BLOCKCHAIN TECHNOLOGY

**B. K. BIRLA COLLEGE OF ARTS, SCIENCE COMMERCE  
(AUTONOMOUS), KALYAN**

*(Affiliated to University of Mumbai)*

**KALYAN-MAHARASHTRA-421301**

**DEPARTMENT OF INFORMATION TECHNOLOGY**



**CERTIFICATE**

This is to certify that Mr/Ms Vivek Pandit bearing Seat. No: (4839747), in class

Msc IT CC has successfully completed practical of the subject Blockchain Technology

Teacher's Signature: \_\_\_\_\_

Place:

Date: 06/07/2024

College Seal

## INDEX

Practical No.	Practical Name
1.	A simple client class that generates the private and public keys by using the built-in Python RSA algorithm and test it.
2.	A transaction class to send and receive money and test it.
3.	Create multiple transactions and display them.
4.	Create a blockchain, a genesis block and execute it.
5.	Create a mining function and test it.
6.	Add blocks to the miner and dump the blockchain.
7.	Implement and demonstrate the use of the following in Solidity: Variable, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs, Mappings, Conversions, Ether Units, Special Variables.
8.	Demonstrate the use of Bitcoin Core API.

## Practical No: 1

**Aim:** A simple client class that generates the private and public keys by using the built-in Python RSA algorithm and test it.

---

### Program:

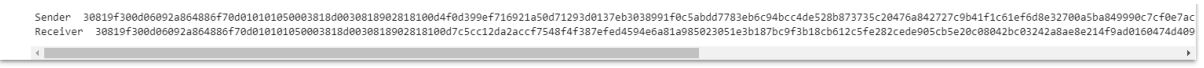
```
# following imports are required by PKI
!pip3 install pycryptodome
import hashlib
import random
import string
import binascii
import datetime
import collections
from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Cipher import PKCS1_v1_5

class Client:
    def __init__(self):

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

Account_1 = Client()
Account_2 = Client()
print ("Sender ",Account_1.identity)
print ("Receiver ",Account_2.identity)
```

### Output:



```
Sender  30819f30d06092a864886f70d010101050003818d0030818902818100d4f0d399ef716921a5ed71293d8137eb3038991f0c5abdd7783eb6c94bc4de528b873735c20476a842727c9b41f1c61ef6d0e32700a5ba849990c7cf0e7ac
Receiver 30819f30d06092a864886f70d010101050003818d0030818902818100d7c5cc12da2accf7548f4f387efed4594e6a81a985023051e3b187bc9f3b18cb612c5fe282ced905cb5e20c08042bc03242a8ae8e214f9ad0160474d409
```

## Practical No: 2

**Aim:** A transaction class to send and receive money and test it.

---

### Program:

```
# following imports are required by PKI
!pip3 install pycryptodome
!pip3 install crypto

import hashlib
import random
import binascii
import datetime
import collections

from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Cipher import PKCS1_v1_5
from collections import OrderedDict
import Crypto
import Crypto.Random
from Crypto.Hash import SHA
from Crypto.Signature import PKCS1_v1_5

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

class Transaction:
    def __init__(self, sender, recipient, value):
        self.sender = sender
        self.recipient = recipient
        self.value = value
        self.time = datetime.datetime.now()

    def to_dict(self):
        if self.sender == "Genesis":
            identity = "Genesis"
        else:
            identity = self.sender.identity
```

```

        return collections.OrderedDict({
            'sender': identity,
            'recipient': self.recipient,
            'value': self.value,
            'time' : self.time})

    def sign_transaction(self):
        private_key = self.sender._private_key
        signer = PKCS1_v1_5.new(private_key)
        h = SHA.new(str(self.to_dict()).encode('utf8'))
        return binascii.hexlify(signer.sign(h)).decode('ascii')

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

sa = Client()
rb = Client()

t1 = Transaction(
    sa,
    rb.identity,
    15.0
)
t1.sign_transaction()
display_transaction (t1)


sa2 = Client()
rb2 = Client()

t2 = Transaction(
    sa,
    rb.identity,
    15.0
)

```

```
t2.sign_transaction()  
display_transaction (t2)
```

## Output:

```
sender: 30819f300d06092a864886f70d0101050003818d0030818902818100be77cec2b4b9f8c0d4b467aea8d5ec7283e9567befbeb8366b0b4c627cfa48d8db579bd18250301d1783f22b21524faa2b0d9e73cfa9d6a4fb135b8d604e77  
-----  
recipient: 30819f300d06092a864886f70d0101050003818d0030818902818100b2a6515cece1bb7afc9f31ea8d8a69257dcce7b32eb57f64dde96dcb1a8315478949238e300e490ef77519c81f66a05024a666a15c8249efab5ce2e614cb  
-----  
value: 15.0  
-----  
time: 2023-05-27 11:56:58.460585  
-----  
sender: 30819f300d06092a864886f70d0101050003818d0030818902818100be77cec2b4b9f8c0d4b467aea8d5ec7283e9567befbeb8366b0b4c627cfa48d8db579bd18250301d1783f22b21524faa2b0d9e73cfa9d6a4fb135b8d604e77  
-----  
recipient: 30819f300d06092a864886f70d0101050003818d0030818902818100b2a6515cece1bb7afc9f31ea8d8a69257dcce7b32eb57f64dde96dcb1a8315478949238e300e490ef77519c81f66a05024a666a15c8249efab5ce2e614cb  
-----  
value: 15.0  
-----  
time: 2023-05-27 11:56:59.310625  
-----
```

## Practical No: 3

**Aim:** Create multiple transactions and display them.

---

### Program:

```
# following imports are required by PKI
!pip3 install pycryptodome

import hashlib
import random
import binascii
import datetime
import collections

from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Cipher import PKCS1_v1_5
from collections import OrderedDict
import Crypto
import Crypto.Random
from Crypto.Hash import SHA
from Crypto.Signature import PKCS1_v1_5

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

class Transaction:
    def __init__(self, sender, recipient, value):
        self.sender = sender
        self.recipient = recipient
        self.value = value
        self.time = datetime.datetime.now()

    def to_dict(self):
        if self.sender == "Genesis":
            identity = "Genesis"
        else:
            identity = self.sender.identity
```



```

        return collections.OrderedDict({
            'sender': identity,
            'recipient': self.recipient,
            'value': self.value,
            'time' : self.time})

    def sign_transaction(self):
        private_key = self.sender._private_key
        signer = PKCS1_v1_5.new(private_key)
        h = SHA.new(str(self.to_dict()).encode('utf8'))
        return binascii.hexlify(signer.sign(h)).decode('ascii')

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

transactions = []

a = Client()
b = Client()
c = Client()

t1 = Transaction(
    a,
    b.identity,
    15.0
)

t1.sign_transaction()
transactions.append(t1)

t2 = Transaction(
    b,
    c.identity,
    25.0
)

t2.sign_transaction()
transactions.append(t2)

t3 = Transaction(

```

```

    a,
    c.identity,
    200.0
)
t3.sign_transaction()
transactions.append(t3)

tn=1
for t in transactions:#t1 t2 t3
    print("Transaction #",tn)
    display_transaction (t)
    tn=tn+1
    print ('-----')
```

## Output:

```

Transaction # 1
sender: 30819f300d06092a864886f70d0101050003818d003081890281810091112edc113228971a35d6448aa7ec4c1db2631fa632783ea98e7612668c14b6f6dd229d9d03894772fbbec4e213d605bb7f35eae6e9aad019a2d745f8a925
-----
recipient: 30819f300d06092a864886f70d0101050003818d0030818902818100c13bc303d071fe26e1856614534722442d4e05896d92637e59e5ef82edd277027cadd203816737025dabc327314f8e4d855286f5b64ac9d037ad926b3305ae
-----
value: 15.0
-----
time: 2023-05-27 11:58:45.925822
-----
Transaction # 2
sender: 30819f300d06092a864886f70d0101050003818d0030818902818100c13bc303d071fe26e1856614534722442d4e05896d92637e59e5ef82edd277027cadd203816737025dabc327314f8e4d855286f5b64ac9d037ad926b3305ae
-----
recipient: 30819f300d06092a864886f70d0101050003818d00308189028181008e8cd78b62bd4adedc0876749f145478d86a74859698c88afb3f69947a2ca5dae7087c58f2a9406fd11550d8ea1a2b889d433feb543e29e2c8ea4abae4
-----
value: 25.0
-----
time: 2023-05-27 11:58:45.930661
-----
Transaction # 3
sender: 30819f300d06092a864886f70d0101050003818d003081890281810091112edc113228971a35d6448aa7ec4c1db2631fa632783ea98e7612668c14b6f6dd229d9d03894772fbbec4e213d605bb7f35eae6e9aad019a2d745f8a925
-----
recipient: 30819f300d06092a864886f70d0101050003818d00308189028181008e8cd78b62bd4adedc0876749f145478d86a74859698c88afb3f69947a2ca5dae7087c58f2a9406fd11550d8ea1a2b889d433feb543e29e2c8ea4abae4
-----
value: 200.0
-----
time: 2023-05-27 11:58:45.932274
-----
```

## Practical No: 4

**Aim:** Create a blockchain, a genesis block and execute it.

---

### Program:

```
# following imports are required by PKI
!pip3 install pycryptodome

import hashlib
import random
import binascii
import numpy as np
import pandas as pd
import datetime
import collections

from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Cipher import PKCS1_v1_5
from collections import OrderedDict
import Crypto
import Crypto.Random
from Crypto.Hash import SHA
from Crypto.Signature import PKCS1_v1_5

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

class Transaction:
    def __init__(self, sender, recipient, value):
        self.sender = sender
        self.recipient = recipient
        self.value = value
        self.time = datetime.datetime.now()

    def to_dict(self):
        if self.sender == "Genesis":
            identity = "Genesis"
        else:
```

```

        identity = self.sender.identity

    return collections.OrderedDict({
        'sender': identity,
        'recipient': self.recipient,
        'value': self.value,
        'time' : self.time})

def sign_transaction(self):
    private_key = self.sender._private_key
    signer = PKCS1_v1_5.new(private_key)
    h = SHA.new(str(self.to_dict()).encode('utf8'))
    return binascii.hexlify(signer.sign(h)).decode('ascii')

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

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

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

a = Client()

t0 = Transaction (
    "Genesis",
    a.identity,
    500.0

```

```

)

block0 = Block()
block0.previous_block_hash = None
Nonce = None
block0.verified_transactions.append (t0)

digest = hash (block0)
last_block_hash = digest

TPCoins = [] #coinbase
TPCoins.append (block0)

dump_blockchain(TPCoins)

```

## Output:

```

Number of blocks in the chain: 1
block # 0
sender: Genesis
-----
recipient: 30819f300d06092a864886f70d0101050003818d0030818902818100c78e4d501699fd81bf11afc757e86db5277bb8be8dbef051346e6485e8dc9886686e9ffe99d8091c3daaf2e14ca924a466a533a1e8f3cab22cd37e52954
-----
value: 500.0
-----
time: 2023-05-27 12:00:35.840627
-----
=====

```

## Practical No: 5

**Aim:** Create a mining function and test it.

**Program:**

```
import hashlib

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

def mine(message, difficulty=1):
    assert difficulty >= 1
    #if(difficulty <1):
    #    return
    #'1'*3=> '111'
    prefix = '1' * difficulty
    print("prefix",prefix)
    for i in range(1000):
        digest = sha256(str(hash(message)) + str(i))
        print("testing=>" + digest)
        if digest.startswith(prefix):
            print ("after " + str(i) + " iterations found nonce: " +
digest)
            return i #i= nonce value

n=mine ("test message",3)
print(n)
```

**Output:**

```
prefix 111
testing->e4fbb7a5b3b4ee727a664002e544f712c47c4bbec816add3a56c403b10e6101e
testing->0e28a1c0020905d48f3500d74dbb4596e356aaab0c9857cfe02b359d0588c0a3
testing->df52800a1e505bc91d50a23c4490c30a51eda7fb7ce7cc4be704a87a766f05
testing->1787fd29b08ca3b0b674091a53d1950919354c8dc86c3e58d44e24af148ac336
testing->8fa8fa9a0a209fc041ce28ae509e196fb3dd09d79cb9bddd4239c4f41682798
testing->da2c315bb71f0ec7bd54bf442b50f527f6866e103227c3f363ed80865d43e84c
testing->ce2bc95ba27b7497c228aa5e2b9697cd3b0680e909d2794c759a9e1f044cc56d
testing->3ebe392a5e8890a1f183e92ba3f39c408cc75c64cbe75e3b590c50ed2ba85925
testing->4a339028191aa09f26f913d40d175e00e4c029f0e690519e2ffa98ece3fd0e
testing->234e23ffe02ec522f5511902d4f67c90bc307dca3668e31279e44afb7000914e
testing->b158410a3d0f37e8bd32812d2842e9782da4f5013c955b7d058b17ca99d30511
testing->360d4ce279585a59566bba21b3c822d150a9ac7d5328c7c15a8a339701afdcba
testing->3c6fc3dc274c2f1c58ddaab7d900c5142dfd7b7bf82d615ad01e7a41f55fa80
testing->08c14f5a52b06e76b7fe10274521c36dd331e5823f79e231efef0f70dce3a63
testing->e21cc137080e50efe137b4fc32973a671da30aa0353c83c3cee5bfc9067cd25
testing->d5a48c73f9338d45f83782a0ad0f54bdc1808ceca30093bac200ffe2b99c250
testing->c7dd2d9f13ecc6f0f558dc7676258060cde05b330912fdb95e3733c229acf0bd
testing->24326395d33a0b449ef094373ce3734d2a3cda2f966446d750ae816c987122ff
testing->ae78912ca77795394a8a85b92f599e86506bc5216f3ae0d54586807c35c6cd87
testing->1db59f2b30f44acb58ea7b92abe4abb16d1468f4d4667535ce0752afe67691b1
testing->5a4ff5540a341aa72226718fc94a781bbd6b5efb137f593f4132e88087c2a3b
testing->a59a3a7a456e410595b74b00930e8570890b901562179b099751ce4766b48
testing->218006ec6cf9d634d7d09d2752d3c236ca7ea202e6636dbf51fcc681bd42ae
testing->8d465a85b380814d8f97ab0f38003ff7e0b2a00171d1f5395bf32acc0c92bf9
testing->a547a3077ec79b7085b39d946284adf3085882f80e1cdcae4ac0be7c75858598
testing->b311b1e5c7499455b0c86b762ac06cf94d7287e0d30507693adbc0d7ad5b4
testing->b4bb6154a5aeb3457b6836ade519c1296722c42508fc9b69016cdeedd31379
testing->0e707e0944519a5f9dea922b25934050d130be91515a462b7e99cde408043884
testing->dfa9a438f37ed582d4ec05d5175c424d1b19f928ee758619d5f378cb019cb8
testing->1b9a0cdf0d07d2e811255886daac79b5969cbcefeb7d61ff077fbcb0151b06
testing->a66de53757d8ea651e3ab856f505ad8219e2f96b79511c2ec43307c53a5425e
testing->8cd132570a5acf458a58ee70fc4fd5eb47d0832bf552a18d398ec95191fd5813
testing->e9cb11355927046d0a7884df03e73dcba480c3281fac4c1e58854ca451580c7
testing->75a509619c9d3947806858f5c6ae7949655ba96fd3ed62e031c8e078fbef7
fact-tnn-v22KazBh37fahB6f7ahf0u77aCa6f72r0fahhfrsaszrhafr1r1c02f08177c308
```

## Practical No: 6

**Aim:** Add blocks to the miner and dump the blockchain.

---

### Program:

```
# following imports are required by PKI
!pip3 install pycryptodome

import hashlib
import random
import binascii
import datetime
import collections

from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Cipher import PKCS1_v1_5
from collections import OrderedDict
import Crypto
import Crypto.Random
from Crypto.Hash import SHA
from Crypto.Signature import PKCS1_v1_5

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

class Transaction:
    def __init__(self, sender, recipient, value):
        self.sender = sender
        self.recipient = recipient
        self.value = value
        self.time = datetime.datetime.now()

    def to_dict(self):
        if self.sender == "Genesis":
            identity = "Genesis"
        else:
            identity = self.sender.identity
```

```

        return collections.OrderedDict({
            'sender': identity,
            'recipient': self.recipient,
            'value': self.value,
            'time' : self.time})

    def sign_transaction(self):
        private_key = self.sender._private_key
        signer = PKCS1_v1_5.new(private_key)
        h = SHA.new(str(self.to_dict()).encode('utf8'))
        return binascii.hexlify(signer.sign(h)).decode('ascii')

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

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

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

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

def mine(message, difficulty=1):
    assert difficulty >= 1
    #if(difficulty <1):
    #    return
    #'1'*3=> '111'

```



```

prefix = '1' * difficulty
for i in range(1000):
    digest = sha256(str(hash(message)) + str(i))
    if digest.startswith(prefix):
        return i #i= nonce value

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

t1 = Transaction (
    Ramesh,
    Dinesh.identity,
    40.0
)
t2 = Transaction (
    Ramesh,
    Dinesh.identity,
    70.0
)
t3 = Transaction (
    Vikas,
    Ramesh.identity,
    700.0
)
#blockchain
TPCoins = []

block0 = Block()
block0.previous_block_hash = None
Nonce = None
block0.verified_transactions.append (t0)
digest = hash (block0)
last_block_hash = digest #last_block_hash it is hash of block0
TPCoins.append (block0)

block1 = Block()
block1.previous_block_hash = last_block_hash
block1.verified_transactions.append (t1)
block1.verified_transactions.append (t2)
block1.Nonce=mine (block1, 2)
digest = hash (block1)

```

```

last_block_hash = digest
TPCoins.append (block1)

block2 = Block()
block2.previous_block_hash = last_block_hash
block2.verified_transactions.append (t3)
Nonce = mine (block2, 2)
block2.Nonce=mine (block2, 2)
digest = hash (block2)
last_block_hash = digest
TPCoins.append (block2)

dump_blockchain(TPCoins)

```

## Output:

```

Number of blocks in the chain: 3
block # 0
sender: Genesis
-----
recipient: 30819f300d06092a864886f70d0101050003818d0030818902818100b328959d51f420c82352e40a3244f5f9df72faf27dea1904e9e059889c4db56e5ad9beb8af3361d7730888349d12e325ffd70005c9df3ce0adc35006f9f
-----
value: 500.0
-----
time: 2023-05-27 12:05:27.457210
-----
-----
block # 1
sender: 30819f300d06092a864886f70d0101050003818d0030818902818100b2fce2c0bfc363b01305d00357591a13cf0347d0e2705ff19fafb2f8254bca64d948ffe175fedcf2c29a5f8f652a6f330748e0e4c0467ae4b2e43fa3b9129
-----
recipient: 30819f300d06092a864886f70d0101050003818d0030818902818100b328959d51f420c82352e40a3244f5f9df72faf27dea1904e9e059889c4db56e5ad9beb8af3361d7730888349d12e325ffd70005c9df3ce0adc35006f9f
-----
value: 40.0
-----
time: 2023-05-27 12:05:52.457510
-----
-----
sender: 30819f300d06092a864886f70d0101050003818d0030818902818100b2fce2c0bfc363b01305d00357591a13cf0347d0e2705ff19fafb2f8254bca64d948ffe175fedcf2c29a5f8f652a6f330748e0e4c0467ae4b2e43fa3b9129
-----
recipient: 30819f300d06092a864886f70d0101050003818d0030818902818100b328959d51f420c82352e40a3244f5f9df72faf27dea1904e9e059889c4db56e5ad9beb8af3361d7730888349d12e325ffd70005c9df3ce0adc35006f9f
-----
value: 70.0
-----
time: 2023-05-27 12:05:52.457756
-----
-----
block # 2
sender: 30819f300d06092a864886f70d0101050003818d0030818902818100c1f49e252e9b066917199fadb988487327308ed6ac4f42e792aa4d5388d1a4d7dda813789e0cccd787fe374a011a93a360750e52a5d642c63ad954ba91df41

```

## Practical No: 7

**Aim:** Implement and demonstrate the use of the following in Solidity: Variable, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs, Mappings, Conversions, Ether Units, Special Variables.

---

### Aim: Types of Variable

#### Program:

```
pragma solidity ^0.5.0;

contract Pract1 {

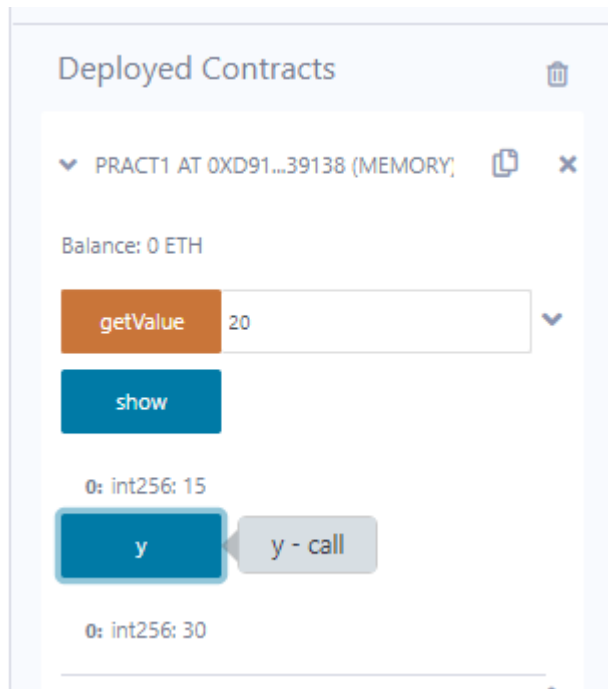
    // State variable
    int x = 15;

    // Global variable
    int public y = 10;

    // Function to get the value
    function getValue(int z) public {
        y = y + z;
    }

    // Function to show the value
    function show() public view returns (int) {
        return x;
    }
}
```

#### Output:



## Aim: Relational Operators

### Program:

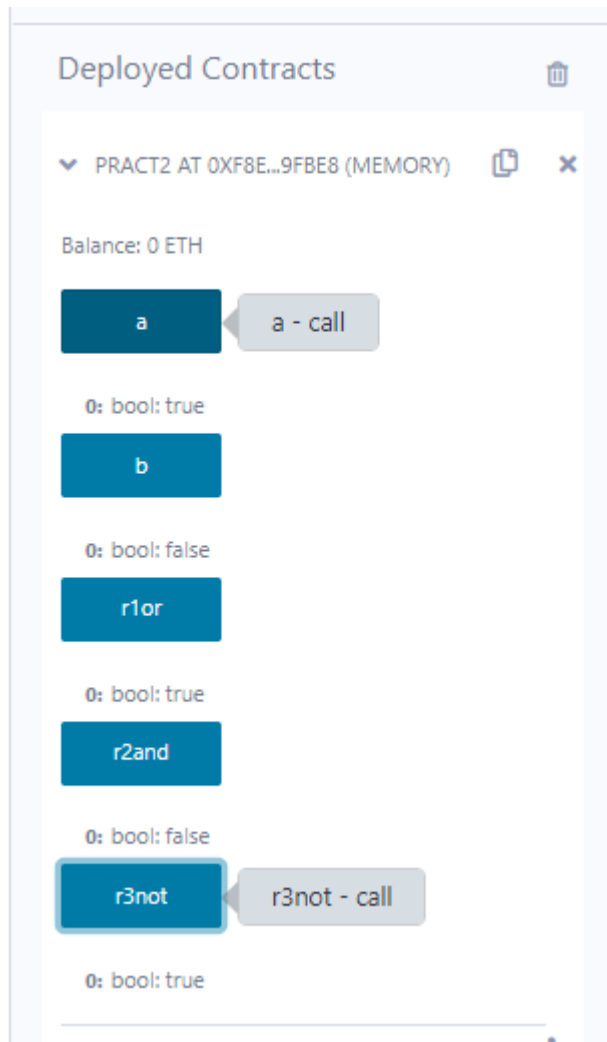
```
pragma solidity ^0.5.0;

contract Pract2 {

    // State variables
    bool public a = true;
    bool public b = false;

    // Boolean operators
    bool public r1or = a || b;
    bool public r2and = a && b;
    bool public r3not = !b;
}
```

### Output:



## Aim: For Loop

### Program:

```
pragma solidity ^0.5.0;

contract Pract3 {

    function test(int s, int e) public view returns(int) {

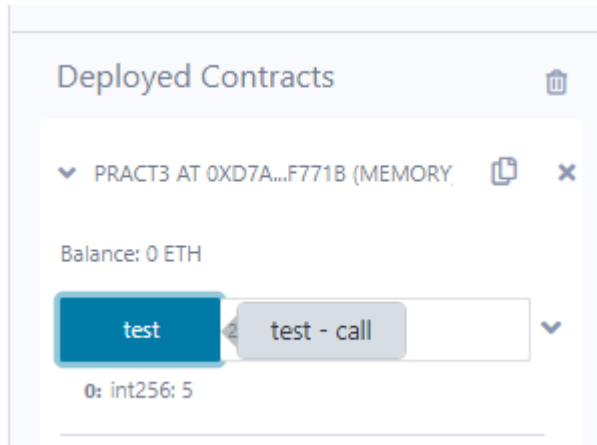
        int i;
        int sum = 0;

        // For loop
        for (i = s; i <= e; i++) {
            sum += i;
        }

        return sum;
    }
}
```

```
}  
}
```

### Output:

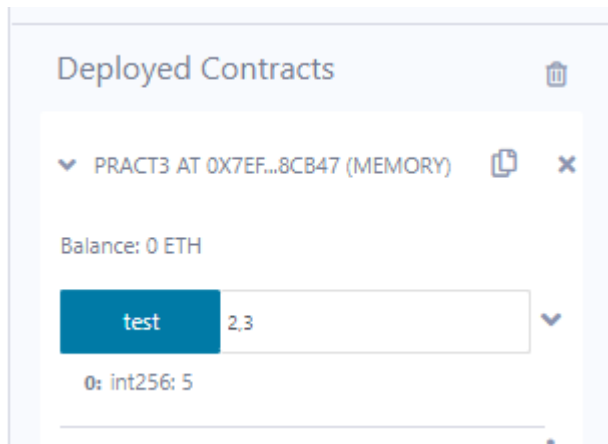


### Aim: While Loop

### Program:

```
pragma solidity ^0.5.0;  
  
contract Pract3 {  
  
    function test(int s, int e) public view returns(int) {  
  
        int i;  
        int sum = 0;  
  
        // While loop  
        i = s;  
        while (i <= e) {  
            sum += i;  
            i++;  
        }  
  
        return sum;  
    }  
}
```

### Output:



### Aim: Do-While Loop

#### Program:

```
pragma solidity ^0.5.0;

contract Pract3 {

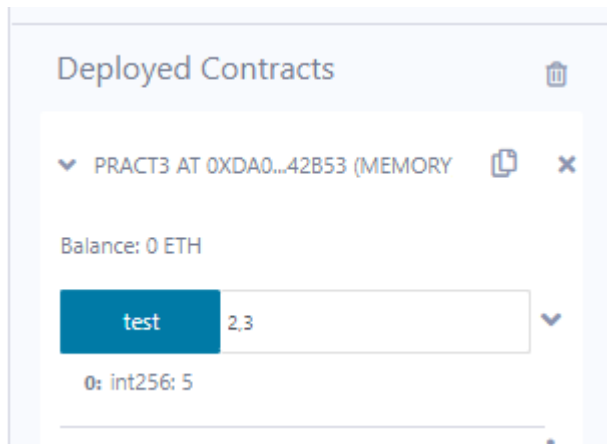
    function test(int s, int e) public view returns(int) {

        int i;
        int sum = 0;

        // Do-while loop
        i = s;
        do {
            sum += i;
            i++;
        } while (i <= e);

        return sum;
    }
}
```

#### Output:



**Aim: If Else**

**Program:**

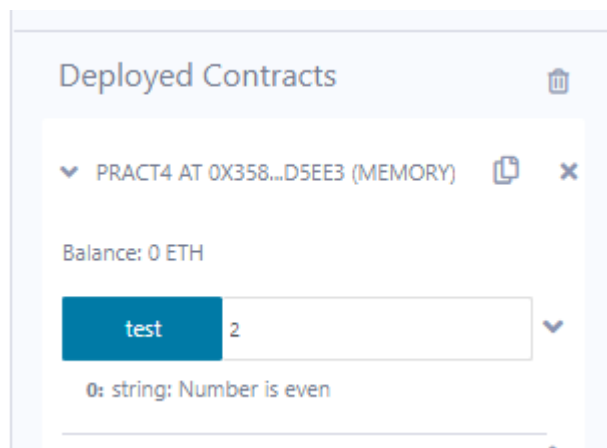
```
pragma solidity ^0.5.0;

contract Pract4 {

    function test(int x) public view returns(string memory) {

        // Check if the number is even
        if (x % 2 == 0) {
            return "Number is even";
        } else {
            return "Number is odd";
        }
    }
}
```

**Output:**





**Aim: string**

**Program:**

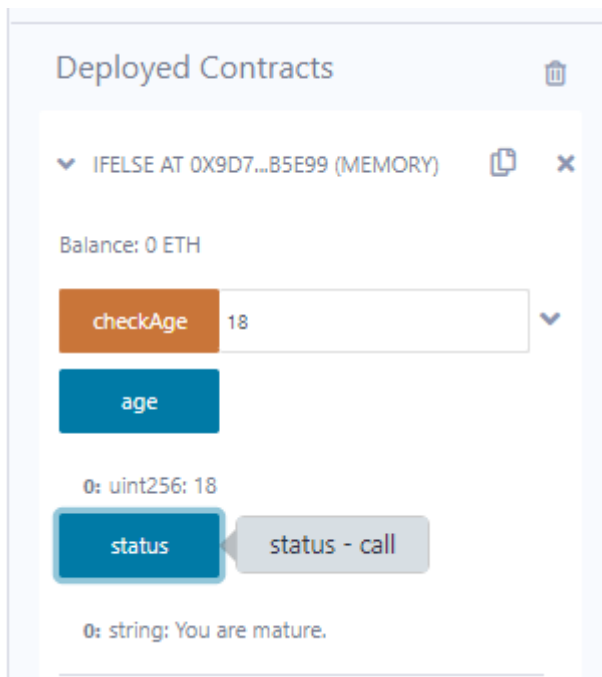
```
pragma solidity ^0.5.0;

contract IfElse {
    uint public age;
    string public status;

    function checkAge(uint _age) public returns (string memory) {
        age = _age;

        if (age >= 18) {
            status = "You are mature.";
        } else {
            status = "You are a minor.";
        }
    }
}
```

**Output:**



**Aim: Array**

**Program:**

```
pragma solidity ^0.5.0;
```

```

contract Types {

    uint[5] data;

    constructor() public {

        data = [uint(10), 20, 30, 40, 50];
    }

    function array_example() public view returns (uint, uint) {

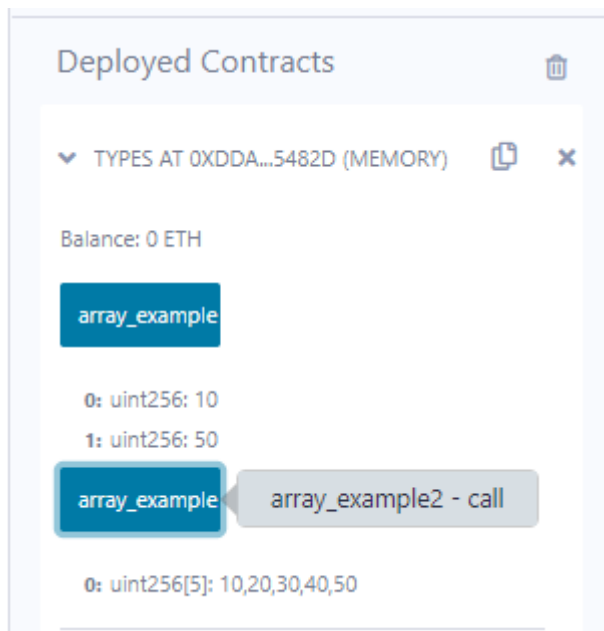
        // Return the element at index 0 of the `data` array and the element at
        index 4.
        return (data[0], data[4]);
    }

    function array_example2() public view returns (uint[5] memory) {

        return data;
    }
}

```

**Output:**



**Aim: Enum**

**Program:**

```

pragma solidity ^0.5.0;

contract Types {

    // Declare an enum called `week_days` with 7 values.
    enum week_days {
        Monday,
        Tuesday,
        Wednesday,
        Thursday,
        Friday,
        Saturday,
        Sunday
    }

    // Declare a variable called `choice` of type `week_days`.
    week_days choice;

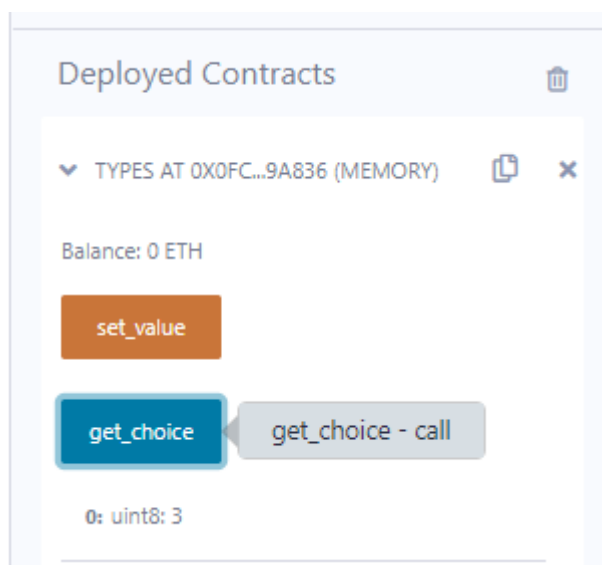
    function set_value() public {

        // Set the value of the `choice` variable to `week_days.Thursday`.
        choice = week_days.Thursday;
    }

    function get_choice() public view returns (week_days) {
        return choice;
    }
}

```

## Output:



## Aim: Arithmetic Operations

### Program:

```
pragma solidity ^0.5.0;

contract SolidityTest {

    uint16 public a = 20;
    uint16 public b = 10;

    // sum of `a` and `b`.
    uint public sum = a + b;

    // difference of `a` and `b`.
    uint public diff = a - b;

    // product of `a` and `b`.
    uint public mul = a * b;

    // quotient of `a` and `b`.
    uint public div = a / b;

    // modulus of `a` and `b`.
    uint public mod = a % b;

    // decrement value of `b`.
    uint public dec = --b;

    // increment value of `a`.
    uint public inc = ++a;
}
```

### Output:



**Aim: Structure**

**Program:**

```
pragma solidity ^0.5.0;
```

```
contract test {

    // This struct defines a book.
    struct Book {
        string title;
        string author;
        uint book_id;
    }

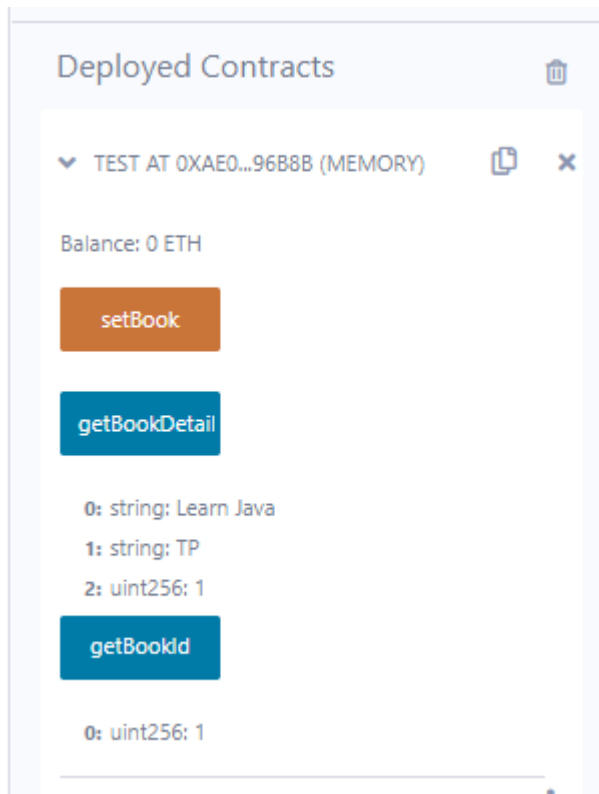
    // This variable stores a book.
    Book book;

    // This function sets the values of the book's members.
    function setBook() public {
        book.title = "Learn Java";
        book.author = "TP";
        book.book_id = 1;
    }

    // This function returns the ID of the book.
    function getBookId() public view returns (uint) {
        return book.book_id;
    }

    // This function returns the title, author, and ID of the book.
    function getBookDetail() public view returns (string memory, string memory,
uint) {
        return (book.title, book.author, book.book_id);
    }
}
```

**Output:**



### Aim: Type of Function (View, Pure)

#### Program:

```
pragma solidity ^0.5.0;

contract Test {

    // global integer.
    int public x = 10;

    // state integer.
    int y = 90;

    function f1() public returns (int) {
        // We can read and update the global integer.
        x = 100;
        return x;
    }

    function f2() public view returns (int) {
        // We can only read the global integer.
        return x;
    }

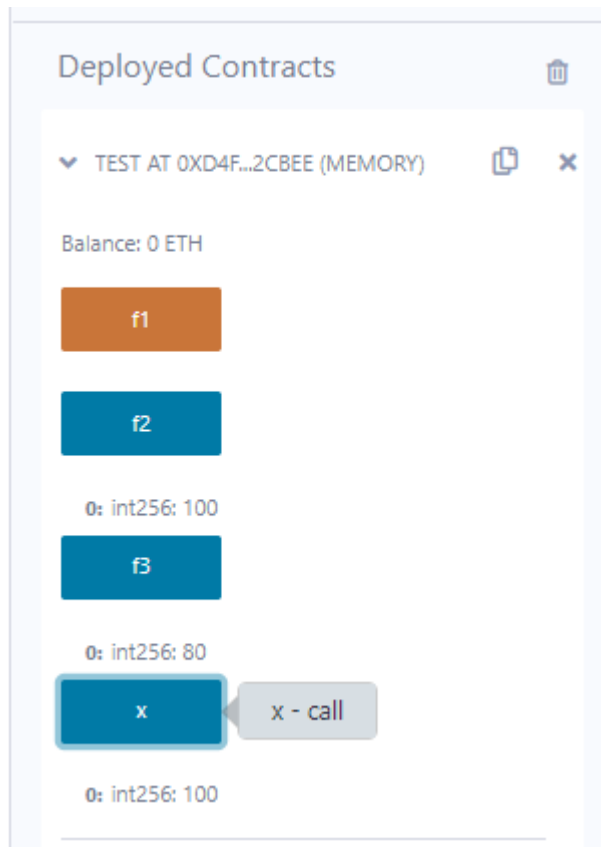
    function f3() public pure returns (int) {
```

```

    // We cannot access or update the global integer in a pure function.
    int z = 80;
    return z;
}
}

```

**Output:**



**Aim: Function Overloading**

**Program:**

```

pragma solidity ^0.5.0;

contract Test {

    function getSum(uint a, uint b) public pure returns (uint) {
        // sum of `a` and `b`.
        return a + b;
    }

    function getSum(uint a, uint b, uint c) public pure returns (uint) {

```

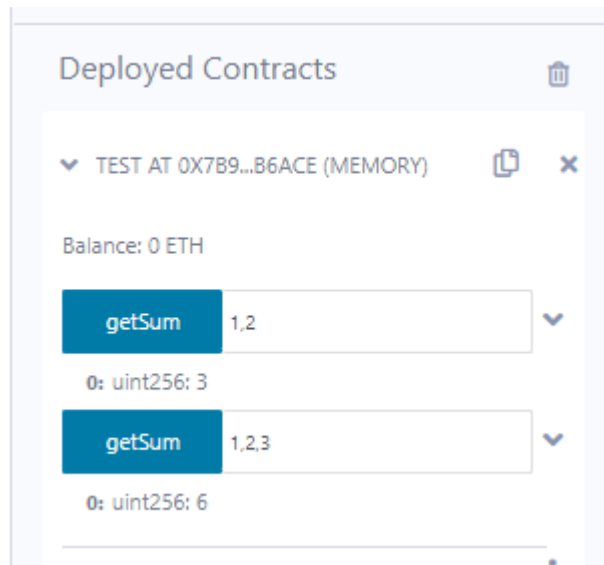


```

    // sum of `a`, `b`, and `c`.
    return a + b + c;
}
}

```

### Output:



### Aim: Mathematical Function

#### Program:

```

pragma solidity ^0.5.0;

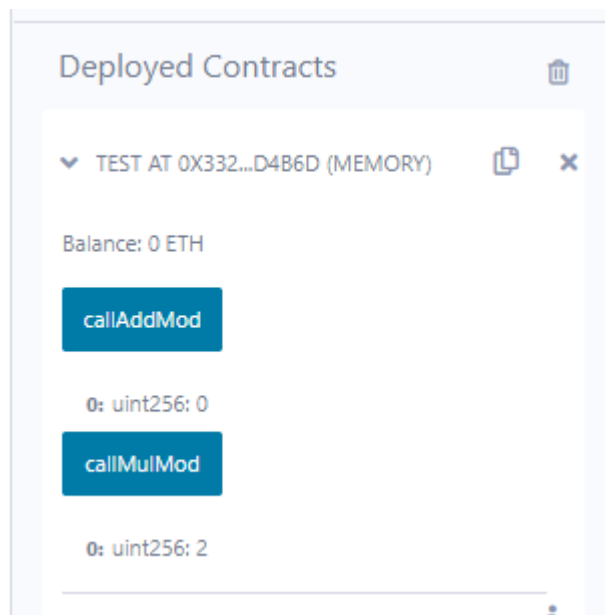
contract Test {

    function callAddMod() public pure returns (uint) {
        // Return the result of the `addmod` operation.
        return addmod(4, 5, 3);
    }

    function callMulMod() public pure returns (uint) {
        // Return the result of the `mulmod` operation.
        return mulmod(4, 5, 3);
    }
}

```

### Output:



### Aim: Cryptographic Function

#### Program:

```
pragma solidity ^0.5.0;

contract Test {

    function callKeccak256() public pure returns (bytes32) {
        // Return the result of the `keccak256` operation.
        return keccak256("ABC");
    }
}
```

### Output:

## Deployed Contracts



▼ TEST AT 0xE28...4157A (MEMORY)



Balance: 0 ETH

callKeccak256

0: bytes32: 0xe1629b9dda060bb30c790834  
6f6af189c16773fa148d3366701fbaa35d5  
4f3c8

## Practical No: 8

**Aim:** Demonstrate the use of Bitcoin Core API.

### Program:

```
from bitcoinlib.wallets import Wallet
w = Wallet.create('Wallet8')
key1 = w.get_key()
print(key1.address)
w.scan()
print(w.info())
```

### Output:



```
IDLE Shell 3.10.11
File Edit Shell Debug Options Window Help

= Wallet Master Key =
ID              33
Private         True
Depth          0

- NETWORK: bitcoin -
- Keys
  38 m/44'/0'/0'/0/0      1FWLgwBJq92CMMhgvP3QLjS4YFgoJlckGk      address index 0
  0.00000000 □
  39 m/44'/0'/0'/0/1      19vt8ZYum753k8YC66tJpbkBLkNYoag336      address index 1
  0.00000000 □
  40 m/44'/0'/0'/0/2      12gpUjEsf3tLREErYP9kuoGxFsEsb8j1hQ      address index 2
  0.00000000 □
  41 m/44'/0'/0'/0/3      1QFvEgKqcVJxtW4H3fGYeH1qWrwhRCsLA      address index 3
  0.00000000 □
  42 m/44'/0'/0'/0/4      1FehiKcN2WamT76hQMTnJyME1KH7fHFx6F      address index 4
  0.00000000 □
  44 m/44'/0'/0'/1/0      1PQnXJTyTCVX1y1GhteKyvqmgE96qiXAvm      address index 0
  0.00000000 □
  45 m/44'/0'/0'/1/1      1Gvw5xhBE1quus4JaeVZUr8yTqtQjkVySN      address index 1
  0.00000000 □
  46 m/44'/0'/0'/1/2      17HM97pPsZ6ZcYC5odmn95jgxqSkbJdQTn      address index 2
  0.00000000 □
  47 m/44'/0'/0'/1/3      1BBfA8xoRtdp5vVB5BkCkiuW39UPESpvEu      address index 3
  0.00000000 □
  48 m/44'/0'/0'/1/4      1GXkANDs95FMfjMvgbRQTL1knWC8PFXhyh6      address index 4
  0.00000000 □

- Transactions Account 0 (0)

= Balance Totals (includes unconfirmed) =

None
>>>
```