INDEX

Sr.no	Title	Pg.No	Date	Sign
1	Write a simple client class that generates the private and public keys by using the built-in Python RSA algorithm and test it.			
2	Setting up Ethereum network by using Geth command line interface.(INSTALLATION)			
3	Transfer ethers from one **contract** to another on an Ethereum testnet			
4	Transfer ethers from one **account** to another on an Ethereum testnet.			
5	Implement and demonstrate the use of the following in Solidity:			
5a	**Variable,Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs**, Mappings, Conversions, Ether Units, Special Variables			
5b	Functions, Function Modifiers, View functions, Pure Functions, Fallback Function, Function Overloading, Mathematical functions, Cryptographic functions			
6	Implement and demonstrate the use of the following in Solidity.			
6a	Withdrawal Pattern, Restricted Access			
6b	Contracts, inheritance, Constructors, Abstract Contracts, Interfaces			
6c	Libraries,Assembly, Events, Error handling.			
7	ploying a contracts on an external blockchain by using Ganache and/or MyEtherwalllet, Metamask			
8	ploy a local private blockchain over a network with Ethereum or Rust (VM)			
9	Implement the mining module of Bitcoin client . The mining module, or miner, should produce blocks that solve proof-of-work puzzle			
10	Compile and test smart contracts on a testing framework using the Ethereum Virtual Machine (EVM).			
11	monstrate the running of the blockchain node			
12	**Demonstrate the use of Bitcoin Core API.**			
13	Create your own blockchain and demonstrate its use			

PRACTICAL 1

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

Code:-

```
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

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')

Gayatri = Client()
print ("sender ",Gayatri.identity)
```

Output:-

```
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

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')

Gayatri = Client()
print ("sender ",Gayatri.identity)
```

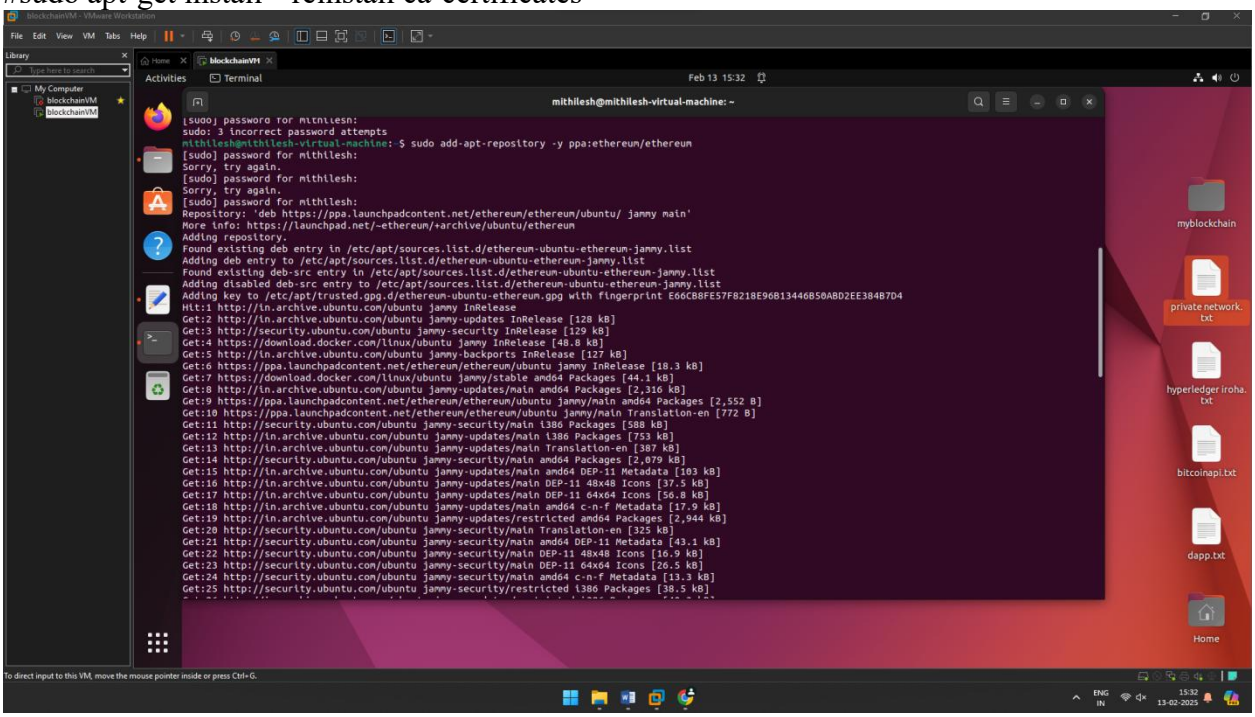
```
sender  30819f300d06092a864886f70d010101050003818d0030818902818100cb786eb7f9418677fcdcb7474d9444fdba6e51c172ee890f562e0d06e75b7054f28b06adbf45a62b52a5d94
a203e063ff48633edba8a7a3bafbf571603892d483d58650647ff8861fc58680782f1f9a428c3911c5b77b824ce2a484215371bf4567b53170a6f7f98010e067ad88cfda1650e63555da88984
8ab8284737863e370203010001
```

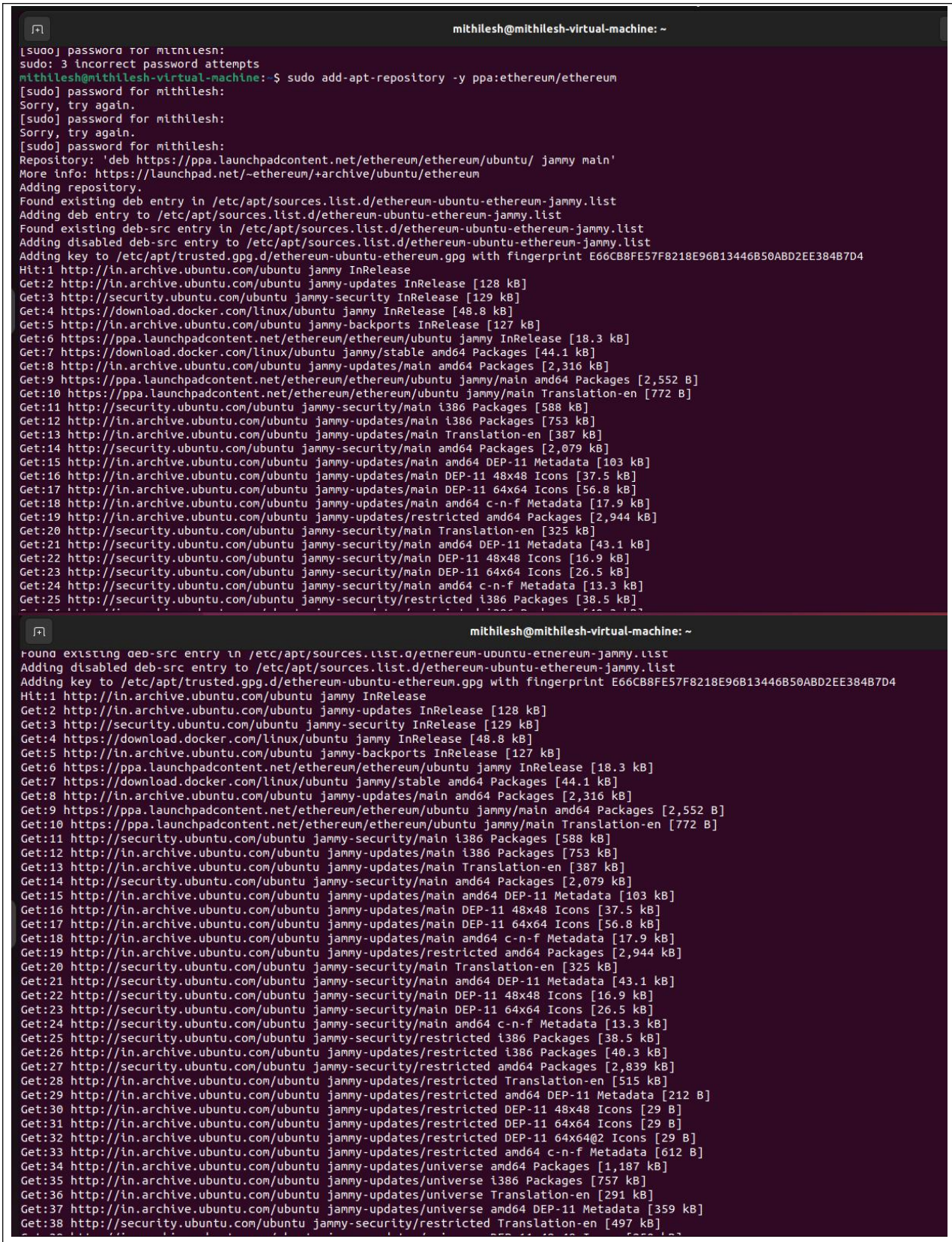
PRACTICAL 2

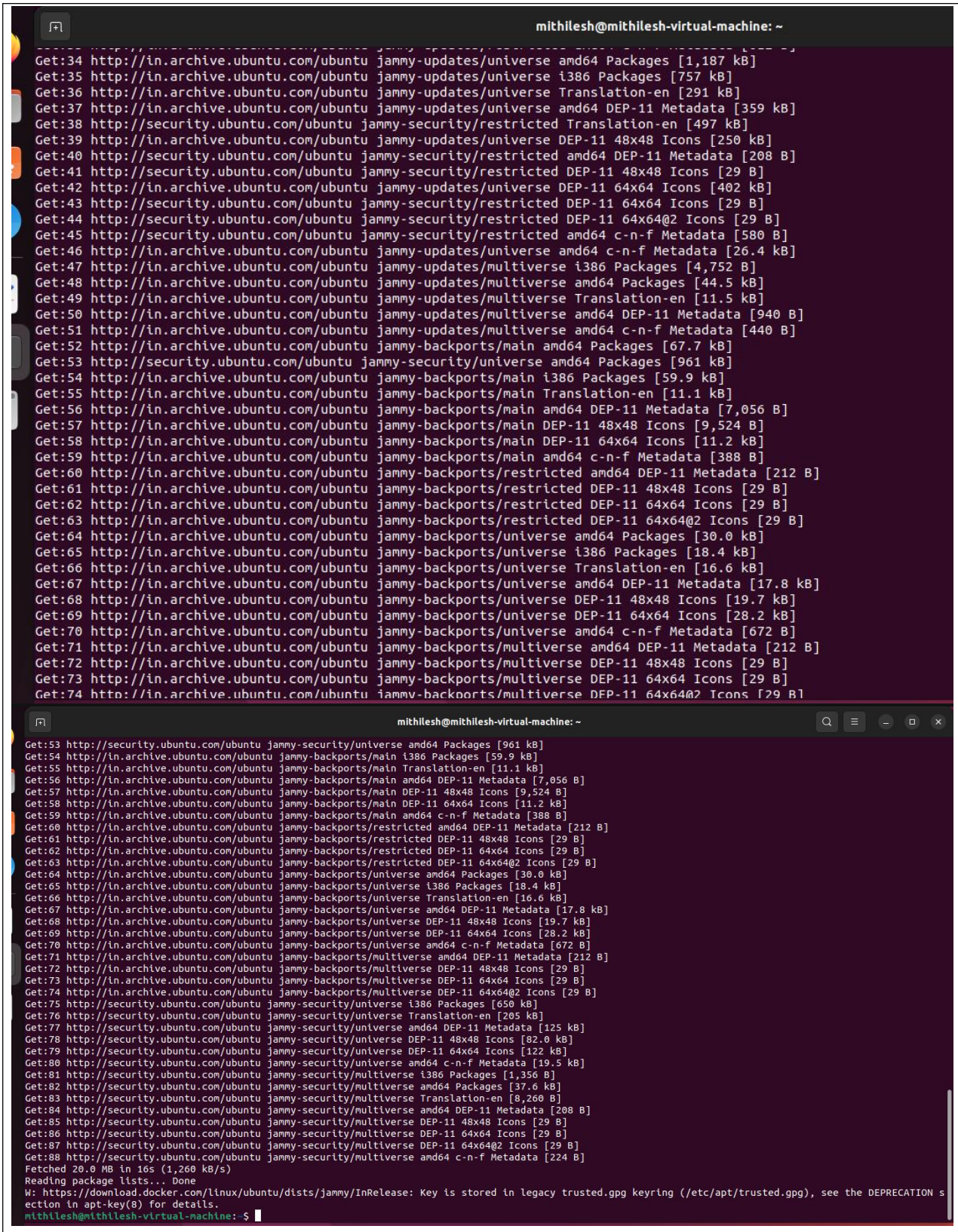
Aim: Setting up Ethereum network by using Geth command line interface.(INSTALLATION)
Aim: Setting up Ethereum network by using Geth command line interface.(INSTALLATION
Install on Ubuntu via PPAs
The easiest way to install go-ethereum on Ubuntu-based distributions is with the built-in launchpad PPAs (Personal Package Archives). We provide a single PPA repository that contains both our stable and development releases for Ubuntu versions trusty, xenial, zesty and artful.
Code:-
linux:
To enable our launchpad repository run:
Step 1: open new terminal
Step 2: on terminal type this command
sudo add-apt-repository -y ppa:ethereum/ethereum
#if above command gives error then run
#sudo apt-get install --reinstall ca-certificates

Step 3: install the stable version of go-ethereum:
sudo apt-get update
sudo apt-get install ethereum

linux:
To enable our launchpad repository run:
Step 1: open new terminal
Step 2: on terminal type this command
sudo add-apt-repository -y ppa:ethereum/ethereum
#if above command gives error then run
#sudo apt-get install --reinstall ca-certificates




mithilesh@mithilesh-virtual-machine: ~
[sudo] password for mithilesh:
sudo: 3 incorrect password attempts
mithilesh@mithilesh-virtual-machine:~$ sudo add-apt-repository -y ppa:ethereum/ethereum
[sudo] password for mithilesh:
Sorry, try again.
[sudo] password for mithilesh:
Sorry, try again.
[sudo] password for mithilesh:
Repository: 'deb https://ppa.launchpadcontent.net/ethereum/ethereum/ubuntu/ jammy main'
More info: https://launchpad.net/~ethereum/+archive/ubuntu/ethereum
Adding repository.
Found existing deb entry in /etc/apt/sources.list.d/ethereum-ubuntu-ethereum-jammy.list
Adding deb entry to /etc/apt/sources.list.d/ethereum-ubuntu-ethereum-jammy.list
Found existing deb-src entry in /etc/apt/sources.list.d/ethereum-ubuntu-ethereum-jammy.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/ethereum-ubuntu-ethereum-jammy.list
Adding key to /etc/apt/trusted.gpg.d/ethereum-ubuntu-ethereum.gpg with fingerprint E66CB8FE57F8218E96B13446B50ABD2EE384B7D4
Hit:1 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:4 https://download.docker.com/linux/ubuntu jammy InRelease [48.8 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:6 https://ppa.launchpadcontent.net/ethereum/ethereum/ubuntu jammy InRelease [18.3 kB]
Get:7 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [44.1 kB]
Get:8 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2,316 kB]
Get:9 https://ppa.launchpadcontent.net/ethereum/ethereum/ubuntu jammy/main amd64 Packages [2,552 B]
Get:10 https://ppa.launchpadcontent.net/ethereum/ethereum/ubuntu jammy/main Translation-en [772 B]
Get:11 http://security.ubuntu.com/ubuntu jammy-security/main i386 Packages [588 kB]
Get:12 http://in.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [753 kB]
Get:13 http://in.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [387 kB]
Get:14 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [2,079 kB]
Get:15 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metadata [103 kB]
Get:16 http://in.archive.ubuntu.com/ubuntu jammy-updates/main DEP-11 48x48 Icons [37.5 kB]
Get:17 http://in.archive.ubuntu.com/ubuntu jammy-updates/main DEP-11 64x64 Icons [56.8 kB]
Get:18 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [17.9 kB]
Get:19 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [2,944 kB]
Get:20 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [325 kB]
Get:21 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [43.1 kB]
Get:22 http://security.ubuntu.com/ubuntu jammy-security/main DEP-11 48x48 Icons [16.9 kB]
Get:23 http://security.ubuntu.com/ubuntu jammy-security/main DEP-11 64x64 Icons [26.5 kB]
Get:24 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [13.3 kB]
Get:25 http://security.ubuntu.com/ubuntu jammy-security/restricted i386 Packages [38.5 kB]
mithilesh@mithilesh-virtual-machine: ~
Found existing deb-src entry in /etc/apt/sources.list.d/ethereum-ubuntu-ethereum-jammy.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/ethereum-ubuntu-ethereum-jammy.list
Adding key to /etc/apt/trusted.gpg.d/ethereum-ubuntu-ethereum.gpg with fingerprint E66CB8FE57F8218E96B13446B50ABD2EE384B7D4
Hit:1 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:4 https://download.docker.com/linux/ubuntu jammy InRelease [48.8 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:6 https://ppa.launchpadcontent.net/ethereum/ethereum/ubuntu jammy InRelease [18.3 kB]
Get:7 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [44.1 kB]
Get:8 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2,316 kB]
Get:9 https://ppa.launchpadcontent.net/ethereum/ethereum/ubuntu jammy/main amd64 Packages [2,552 B]
Get:10 https://ppa.launchpadcontent.net/ethereum/ethereum/ubuntu jammy/main Translation-en [772 B]
Get:11 http://security.ubuntu.com/ubuntu jammy-security/main i386 Packages [588 kB]
Get:12 http://in.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [753 kB]
Get:13 http://in.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [387 kB]
Get:14 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [2,079 kB]
Get:15 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metadata [103 kB]
Get:16 http://in.archive.ubuntu.com/ubuntu jammy-updates/main DEP-11 48x48 Icons [37.5 kB]
Get:17 http://in.archive.ubuntu.com/ubuntu jammy-updates/main DEP-11 64x64 Icons [56.8 kB]
Get:18 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [17.9 kB]
Get:19 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [2,944 kB]
Get:20 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [325 kB]
Get:21 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [43.1 kB]
Get:22 http://security.ubuntu.com/ubuntu jammy-security/main DEP-11 48x48 Icons [16.9 kB]
Get:23 http://security.ubuntu.com/ubuntu jammy-security/main DEP-11 64x64 Icons [26.5 kB]
Get:24 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [13.3 kB]
Get:25 http://security.ubuntu.com/ubuntu jammy-security/restricted i386 Packages [38.5 kB]
Get:26 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted i386 Packages [40.3 kB]
Get:27 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [2,839 kB]
Get:28 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [515 kB]
Get:29 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 DEP-11 Metadata [212 B]
Get:30 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted DEP-11 48x48 Icons [29 B]
Get:31 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted DEP-11 64x64 Icons [29 B]
Get:32 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted DEP-11 64x64@2 Icons [29 B]
Get:33 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 c-n-f Metadata [612 B]
Get:34 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1,187 kB]
Get:35 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe i386 Packages [757 kB]
Get:36 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [291 kB]
Get:37 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 DEP-11 Metadata [359 kB]
Get:38 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [497 kB]

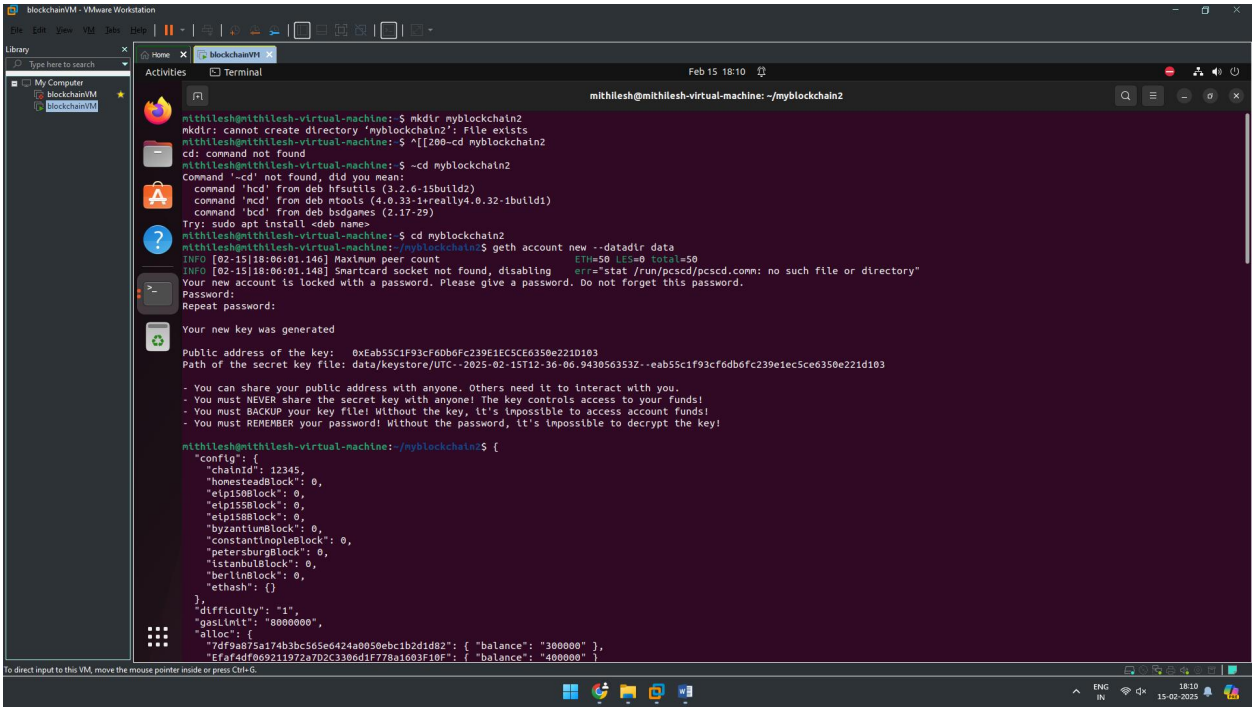


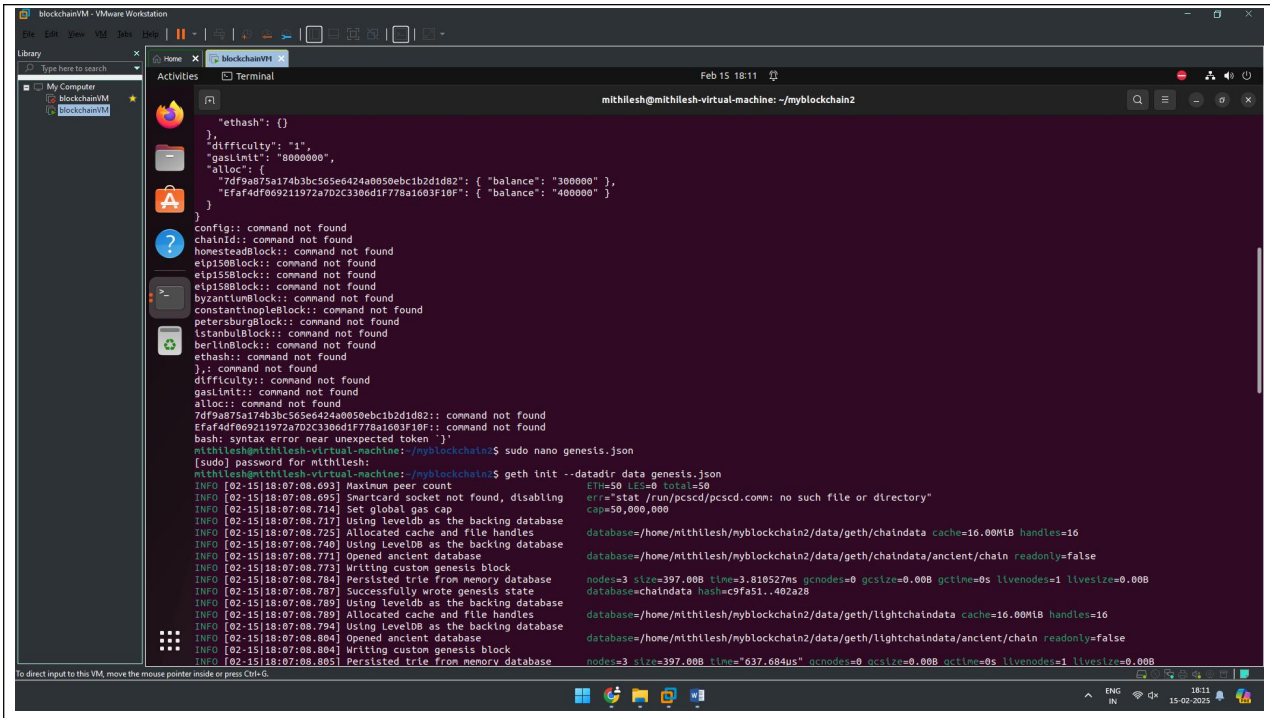
Step 3: install the stable version of go-ethereum:
sudo apt-get update

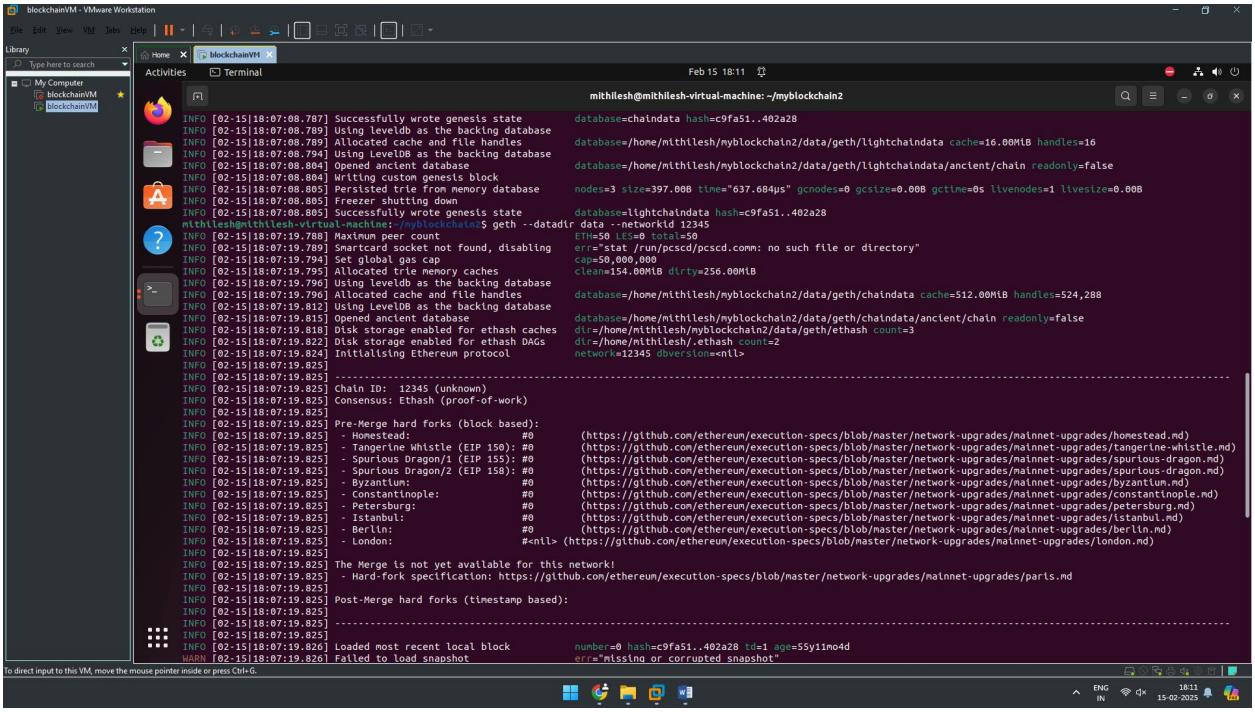
```
mithilesh@mithilesh-virtual-machine: ~
Get:87 http://security.ubuntu.com/ubuntu jammy-security/multiverse DEP-11 64x64@2 Icons [29 B]
Get:88 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [224 B]
Fetched 20.0 MB in 16s (1,260 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/jammy/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION s
ection in apt-key(8) for details.
mithilesh@mithilesh-virtual-machine:~$ sudo apt-get update
Hit:1 https://download.docker.com/linux/ubuntu jammy InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Get:5 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Hit:6 https://ppa.launchpadcontent.net/ethereum/ethereum/ubuntu jammy InRelease
Get:7 http://in.archive.ubuntu.com/ubuntu jammy-backports/main amd64 DEP-11 Metadata [7,048 B]
Get:8 http://in.archive.ubuntu.com/ubuntu jammy-backports/restricted amd64 DEP-11 Metadata [216 B]
Get:9 http://in.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 DEP-11 Metadata [17.8 kB]
Get:10 http://in.archive.ubuntu.com/ubuntu jammy-backports/multiverse amd64 DEP-11 Metadata [212 B]
Fetched 152 kB in 3s (55.8 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/jammy/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION s
ection in apt-key(8) for details.
```

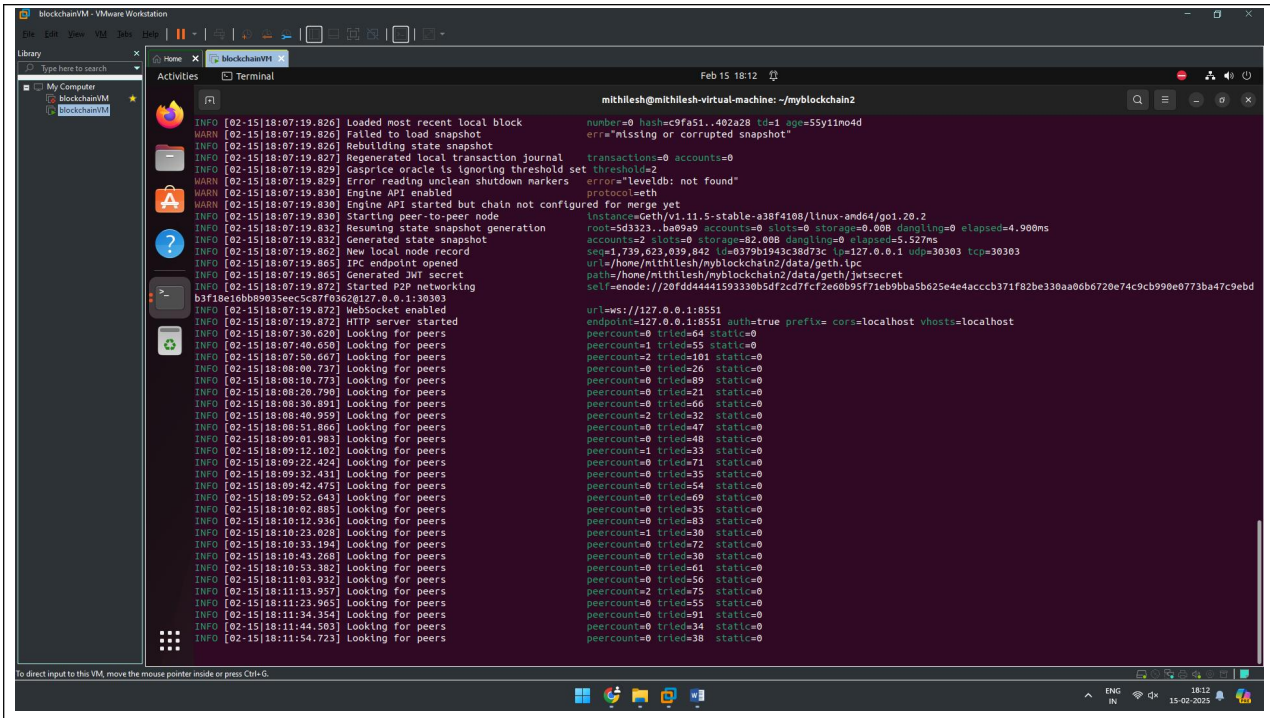
sudo apt-get install ethereum

```
mithilesh@mithilesh-virtual-machine:~$ sudo apt-get install ethereum
'
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  bootnode
Use 'sudo apt autoremove' to remove it.
The following packages will be upgraded:
  ethereum
1 upgraded, 0 newly installed, 0 to remove and 536 not upgraded.
Need to get 1,454 B of archives.
After this operation, 0 B of additional disk space will be used.
Get:1 https://ppa.launchpadcontent.net/ethereum/ethereum/ubuntu jammy/main amd64 ethereum amd64 1.15.0+build30732+jammy [1,454 B]
Fetched 1,454 B in 1s (1,669 B/s)
(Reading database ... 194923 files and directories currently installed.)
Preparing to unpack .../ethereum_1.15.0+build30732+jammy_amd64.deb ...
Unpacking ethereum (1.15.0+build30732+jammy) over (1.11.5+build28443+jammy) ...
Setting up ethereum (1.15.0+build30732+jammy) ...
```



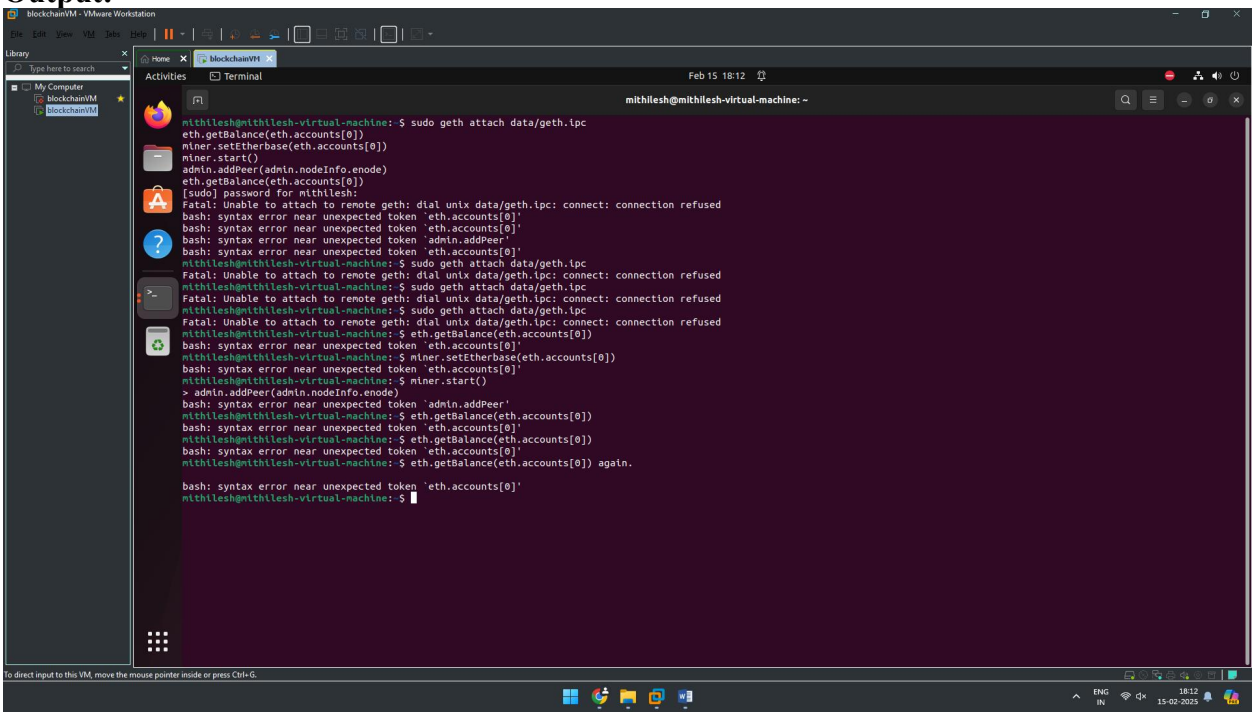

blockchainVM - VMware Workstation
Activities
Terminal
Feb 15 18:11
mithilesh@mithilesh-virtual-machine: ~/myblockchain2
"ethash": {}
},
"difficulty": "1",
"gasLimit": "8000000",
"alloc": {
"7df9a875a174b3bc565e6424a0050ebc1b2d1d82": { "balance": "300000" },
"Efaf4df069211972a7D2C3306d1F778a1603F10F": { "balance": "400000" }
}
}
config:: command not found
chainId:: command not found
homesteadBlock:: command not found
eip150Block:: command not found
eip155Block:: command not found
eip158Block:: command not found
byzantiumBlock:: command not found
constantinopleBlock:: command not found
petersburgBlock:: command not found
istanbulBlock:: command not found
berlinBlock:: command not found
ethash:: command not found
},: command not found
difficulty:: command not found
gasLimit:: command not found
alloc:: command not found
7df9a875a174b3bc565e6424a0050ebc1b2d1d82:: command not found
Efaf4df069211972a7D2C3306d1F778a1603F10F:: command not found
bash: syntax error near unexpected token `}'
mithilesh@mithilesh-virtual-machine:~/myblockchain2$ sudo nano genesis.json
[sudo] password for mithilesh:
mithilesh@mithilesh-virtual-machine:~/myblockchain2$ geth init --datadir data genesis.json
INFO [02-15|18:07:08.693] Maximum peer count ETH=50 LES=0 total=50
INFO [02-15|18:07:08.695] Smartcard socket not found, disabling err="stat /run/pcscd/pcscd.comm: no such file or directory"
INFO [02-15|18:07:08.714] Set global gas cap cap=50,000,000
INFO [02-15|18:07:08.717] Using leveldb as the backing database
INFO [02-15|18:07:08.725] Allocated cache and file handles database=/home/mithilesh/myblockchain2/data/geth/chaindata cache=16.00MiB handles=16
INFO [02-15|18:07:08.740] Using LevelDB as the backing database
INFO [02-15|18:07:08.771] Opened ancient database database=/home/mithilesh/myblockchain2/data/geth/chaindata/ancient/chain readonly=false
INFO [02-15|18:07:08.773] Writing custom genesis block
INFO [02-15|18:07:08.784] Persisted trie from memory database nodes=3 size=397.00B time=3.810527ms gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [02-15|18:07:08.787] Successfully wrote genesis state database=chaindata hash=c9fa51..402a28
INFO [02-15|18:07:08.789] Using leveldb as the backing database
INFO [02-15|18:07:08.789] Allocated cache and file handles database=/home/mithilesh/myblockchain2/data/geth/lightchaindata cache=16.00MiB handles=16
INFO [02-15|18:07:08.794] Using LevelDB as the backing database
INFO [02-15|18:07:08.804] Opened ancient database database=/home/mithilesh/myblockchain2/data/geth/lightchaindata/ancient/chain readonly=false
INFO [02-15|18:07:08.804] Writing custom genesis block
INFO [02-15|18:07:08.805] Persisted trie from memory database nodes=3 size=397.00B time="637.684µs" gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
ENG IN
18:11
15-02-2025


blockchainVM - VMware Workstation
Activities
Terminal
Feb 15 18:11
mithilesh@mithilesh-virtual-machine: ~/myblockchain2
INFO [02-15|18:07:08.787] Successfully wrote genesis state database=chaindata hash=c9fa51..402a28
INFO [02-15|18:07:08.789] Using leveldb as the backing database
INFO [02-15|18:07:08.789] Allocated cache and file handles database=/home/mithilesh/myblockchain2/data/geth/lightchaindata cache=16.00MiB handles=16
INFO [02-15|18:07:08.794] Using LevelDB as the backing database
INFO [02-15|18:07:08.804] Opened ancient database database=/home/mithilesh/myblockchain2/data/geth/lightchaindata/ancient/chain readonly=false
INFO [02-15|18:07:08.804] Writing custom genesis block
INFO [02-15|18:07:08.805] Persisted trie from memory database nodes=3 size=397.00B time="637.684µs" gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [02-15|18:07:08.805] Freezer shutting down
INFO [02-15|18:07:08.805] Successfully wrote genesis state database=lightchaindata hash=c9fa51..402a28
mithilesh@mithilesh-virtual-machine:~/myblockchain2$ geth --datadir data --networkid 12345
INFO [02-15|18:07:19.780] Maximum peer count ETH=50 LES=0 total=50
INFO [02-15|18:07:19.789] Smartcard socket not found, disabling err="stat /run/pcscd/pcscd.comm: no such file or directory"
INFO [02-15|18:07:19.794] Set global gas cap cap=50,000,000
INFO [02-15|18:07:19.795] Allocated trie memory caches clean=154.00MiB dirty=256.00MiB
INFO [02-15|18:07:19.796] Using leveldb as the backing database
INFO [02-15|18:07:19.796] Allocated cache and file handles database=/home/mithilesh/myblockchain2/data/geth/chaindata cache=512.00MiB handles=524,288
INFO [02-15|18:07:19.812] Using LevelDB as the backing database
INFO [02-15|18:07:19.815] Opened ancient database database=/home/mithilesh/myblockchain2/data/geth/chaindata/ancient/chain readonly=false
INFO [02-15|18:07:19.818] Disk storage enabled for ethash caches dir=/home/mithilesh/myblockchain2/data/geth/ethash count=3
INFO [02-15|18:07:19.822] Disk storage enabled for ethash DAGs dir=/home/mithilesh/.ethash count=2
INFO [02-15|18:07:19.824] Initialising Ethereum protocol network=12345 dbversion=<nil>
INFO [02-15|18:07:19.825]
INFO [02-15|18:07:19.825] ----------
INFO [02-15|18:07:19.825] Chain ID: 12345 (unknown)
INFO [02-15|18:07:19.825] Consensus: Ethash (proof-of-work)
INFO [02-15|18:07:19.825]
INFO [02-15|18:07:19.825] Pre-Merge hard forks (block based):
INFO [02-15|18:07:19.825] - Homestead: #0 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/homestead.md)
INFO [02-15|18:07:19.825] - Tangerine Whistle (EIP 150): #0 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/tangerine-whistle.md)
INFO [02-15|18:07:19.825] - Spurious Dragon/1 (EIP 155): #0 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/spurious-dragon.md)
INFO [02-15|18:07:19.825] - Spurious Dragon/2 (EIP 158): #0 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/spurious-dragon.md)
INFO [02-15|18:07:19.825] - Byzantium: #0 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/byzantium.md)
INFO [02-15|18:07:19.825] - Constantinople: #0 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/constantinople.md)
INFO [02-15|18:07:19.825] - Petersburg: #0 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/petersburg.md)
INFO [02-15|18:07:19.825] - Istanbul: #0 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/istanbul.md)
INFO [02-15|18:07:19.825] - Berlin: #0 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/berlin.md)
INFO [02-15|18:07:19.825] - London: #<nil> (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/london.md)
INFO [02-15|18:07:19.825]
INFO [02-15|18:07:19.825] The Merge is not yet available for this network!
INFO [02-15|18:07:19.825] - Hard-fork specification: https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/paris.md
INFO [02-15|18:07:19.825]
INFO [02-15|18:07:19.825] Post-Merge hard forks (timestamp based):
INFO [02-15|18:07:19.825]
INFO [02-15|18:07:19.825] ----------
INFO [02-15|18:07:19.825]
INFO [02-15|18:07:19.826] Loaded most recent local block number=0 hash=c9fa51..402a28 td=1 age=55y11mo4d
WARN [02-15|18:07:19.826] Failed to load snapshot err="missing or corrupted snapshot"
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
ENG IN
18:11
15-02-2025



Terminal 2 :

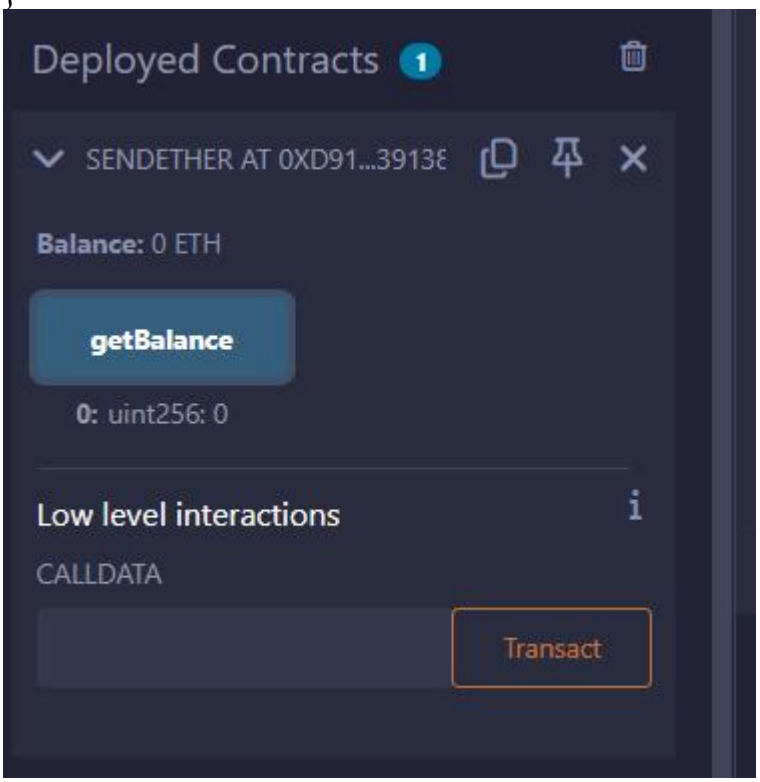
Output:-



PRACTICAL 3

Aim:-
Code:-
Output:-
Transfer ethers from one **contract** to another on an Ethereum testnet.

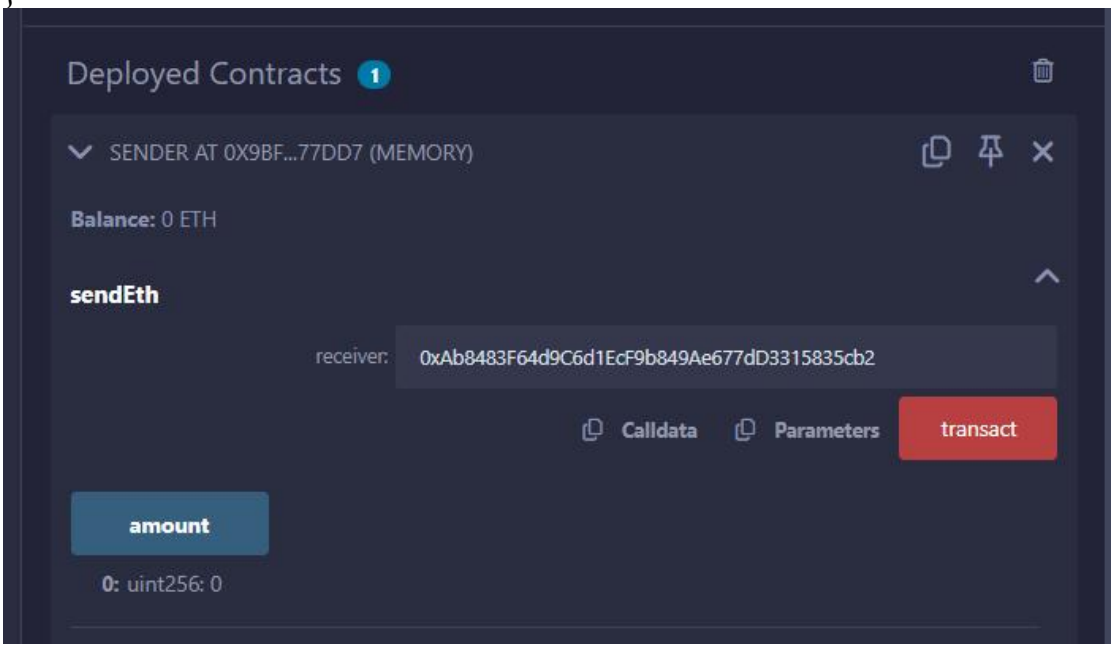
```
pragma solidity ^0.8.0;
contract sendEther{
function getBalance() external view returns(uint)
{
   return address(this).balance;
}
receive() external payable { }
}
```

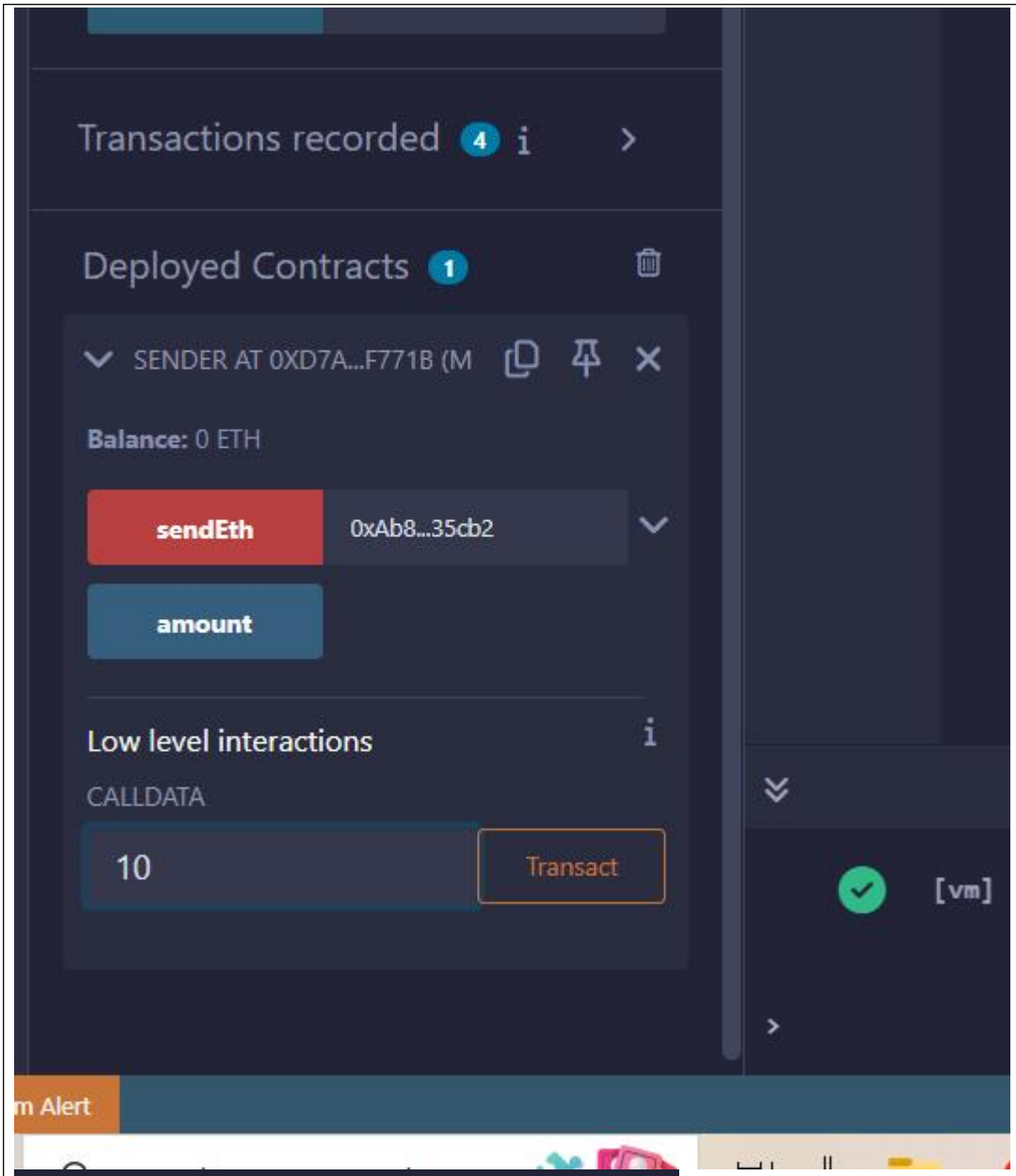


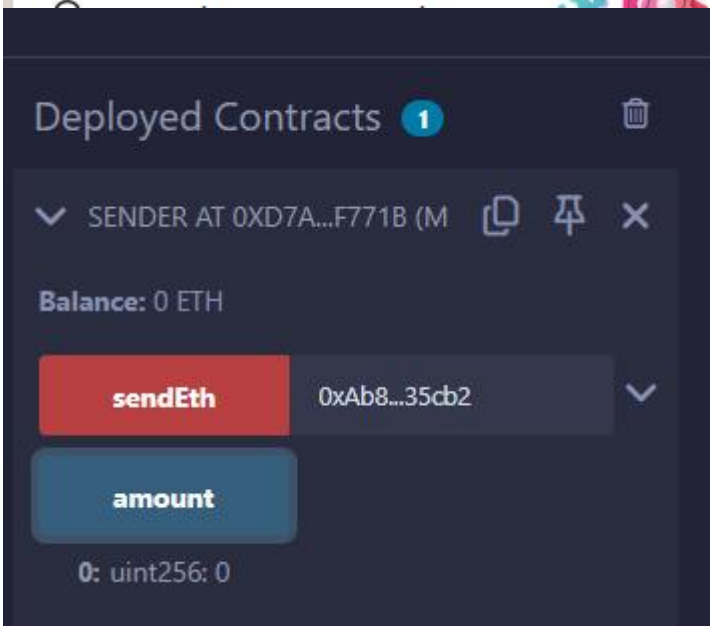
PRACTICAL 4

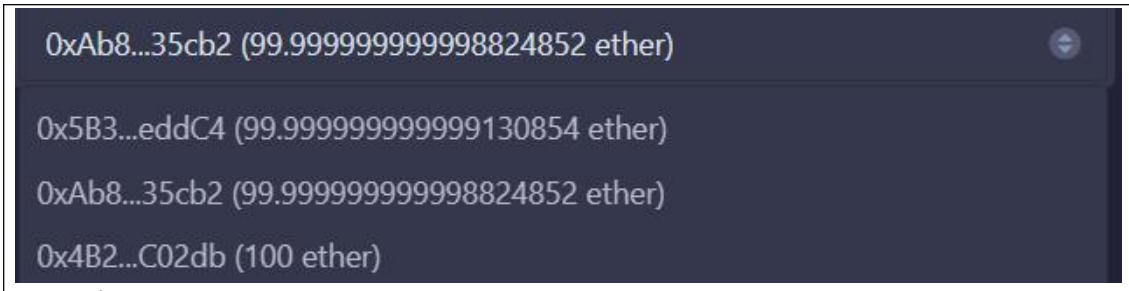
Aim:-
Code:-
Output: Transfer ethers from one **account** to another on an Ethereum testnet.

```
//https://dev.to/sparklesix/solidity-tutorial-how-to-build-and-deploy-a-smart-contract-to-send-ether-from-one-account-to-another-n54
pragma solidity ^0.8.11;

contract Sender {
  uint public amount;
  address payable owner;

  constructor (){
    owner = payable(msg.sender); // set the deployer of contract as the owner
  }
  function sendEth(address payable receiver) payable public{
    require(owner == msg.sender, "Only the owner can send funds");
    amount = msg.value;
    receiver.transfer(amount);
}
}
```




Transactions recorded 4
Deployed Contracts 1
SENDER AT 0XD7A...F771B (M
Balance: 0 ETH
sendEth
0xAb8...35cb2
amount
Low level interactions
CALLDATA
10
Transact
[vm]
m Alert


Deployed Contracts 1
SENDER AT 0XD7A...F771B (M
Balance: 0 ETH
sendEth
0xAb8...35cb2
amount
0: uint256: 0



Ganach
https://abhibvp003.medium.com/how-to-install-and-execute-truffle-on-an-ubuntu-16-04-7d0ff6458c9b

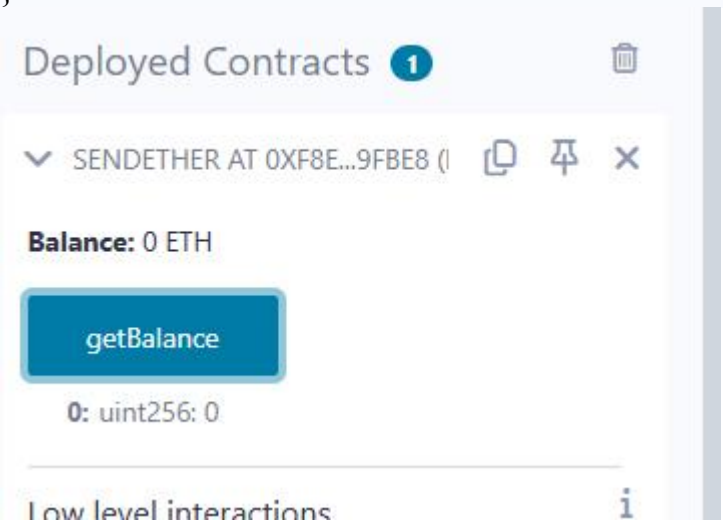
https://ethereum.stackexchange.com/questions/93533/call-an-existing-contract-function-from-truffle-console

```
sudo apt-get -y install curl git vim build-essential
sudo apt-get install curl software-properties-common

sudo apt install npm
sudo npm install -g web3
sudo apt-get install nodejs
sudo apt install python3.9
curl -sL https://deb.nodesource.com/setup_10.x | sudo bash -
sudo npm install --global node-sass@latest
sudo npm install -g truffle@latest
sudo npm install -g  ganache-cli
export NODE_OPTIONS=--openssl-legacy-provider


////to update npm//
sudo npm cache clean -f
sudo npm install -g n

sudo n latest

////////////////////////////////////
Start from here!!!

mkdir upg1
cd upg1
truffle init


////////// create contract
nano contracts/HelloWorld.sol
pragma solidity ^0.5.0;
contract HelloWorld {
    function sayHello() public pure returns(string memory){
        return("hello world");
    }
}
//////////////////create configuration
nano migrations/1_initial_migration.js
const Migrations = artifacts.require("HelloWorld");

module.exports = function (deployer) {
  deployer.deploy(Migrations,"hello");
};
```

```
////////////////network configuration
nano truffle-config.js
module.exports = {
  networks: {
    development: {
      host: "127.0.0.1",
      port: 8545,
      network_id: "*",
    }
  }
}
////////////////////start ganache-cli

ganache-cli



////////////////
truffle migrate

truffle console
#replace contact address
contract = await HelloWorld.at('0x37354B83aadd35516c56f24b724228f29300be77')
a = await contract.sayHello()
a
```

1. Transfer ethers from one **contract** to another on an Ethereum testnet.

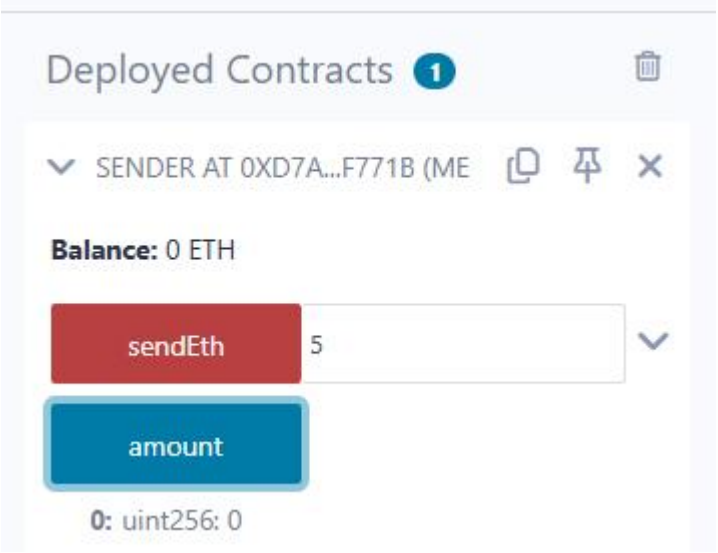
```
pragma solidity ^0.8.11;
contract sendEther{
function getBalance() external view returns(uint)
{
   return address(this).balance;
}
receive() external payable { }
}
```



2. Transfer ethers from one **account** to another on an Ethereum testnet.

```
pragma solidity ^0.8.11;

contract Sender {
  uint public amount;
```

```
  address payable owner;

  constructor (){
    owner = payable(msg.sender); // set the deployer of contract as the owner
  }
  function sendEth(address payable receiver) payable public{
    require(owner == msg.sender, "Only the owner can send funds");
    amount = msg.value;
    receiver.transfer(amount);
}
}
```



PRACTICAL 5

5Implement and demonstrate the use of the following in Solidity:

PRACTICAL 5a

Aim:-**Variable, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs**, Mappings, Conversions, Ether Units, Special Variables

Code:-

A)Variables:

supports three types of variables.

State Variables − Variables whose values are permanently stored in a contract storage.

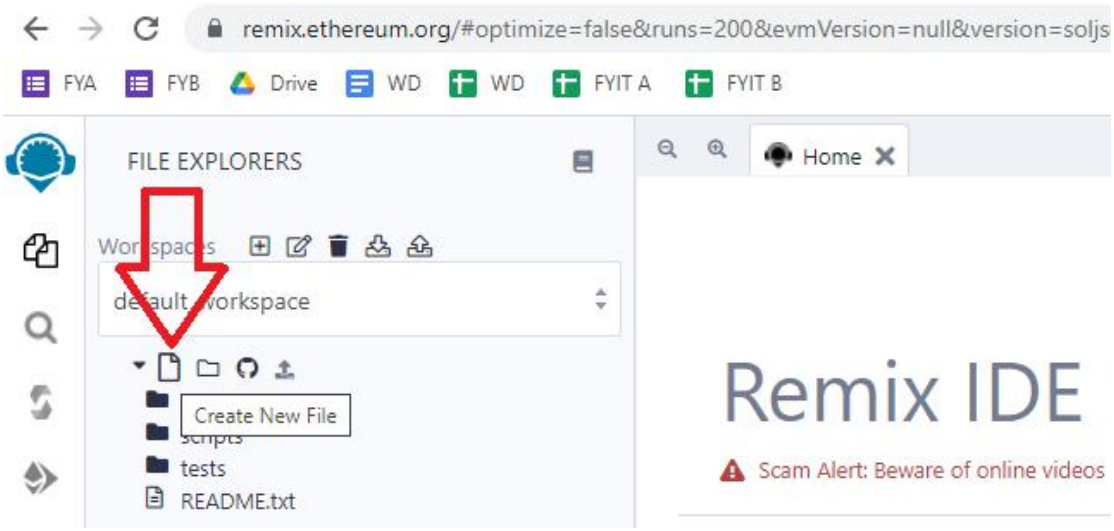
Local Variables − Variables whose values are present till function is executing.

Global Variables − Special variables exists in the global namespace used to get information about the blockchain.i.e. blockhash(uint blockNumber) returns (bytes32), block.coinbase (address payable), block.difficulty (uint)…..and many more

Step 1: Open this website

https://remix.ethereum.org/

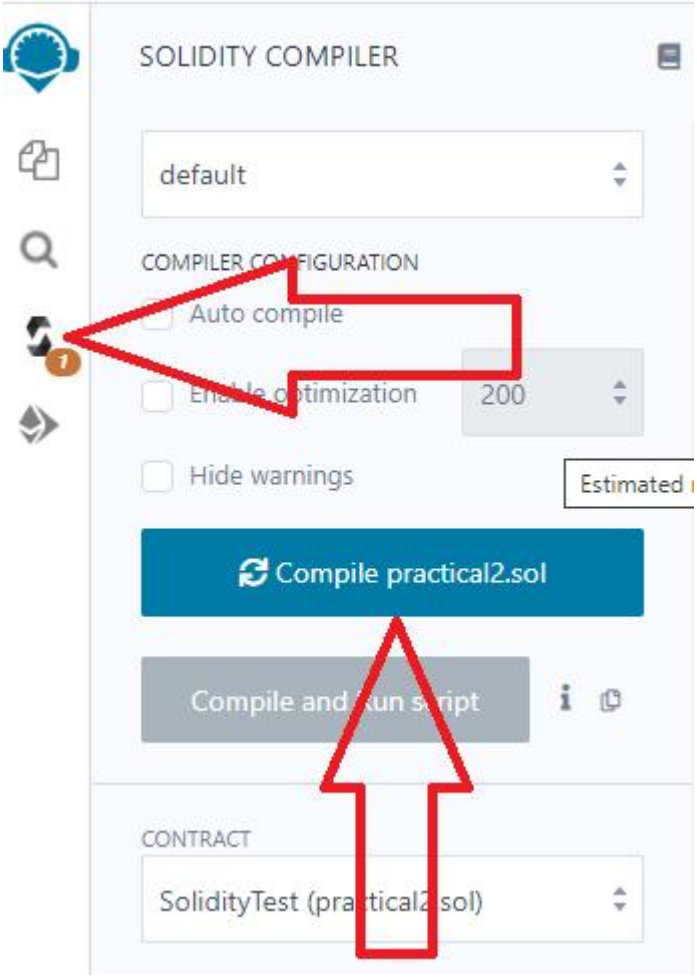
Step 2: Create new file – practical.sol



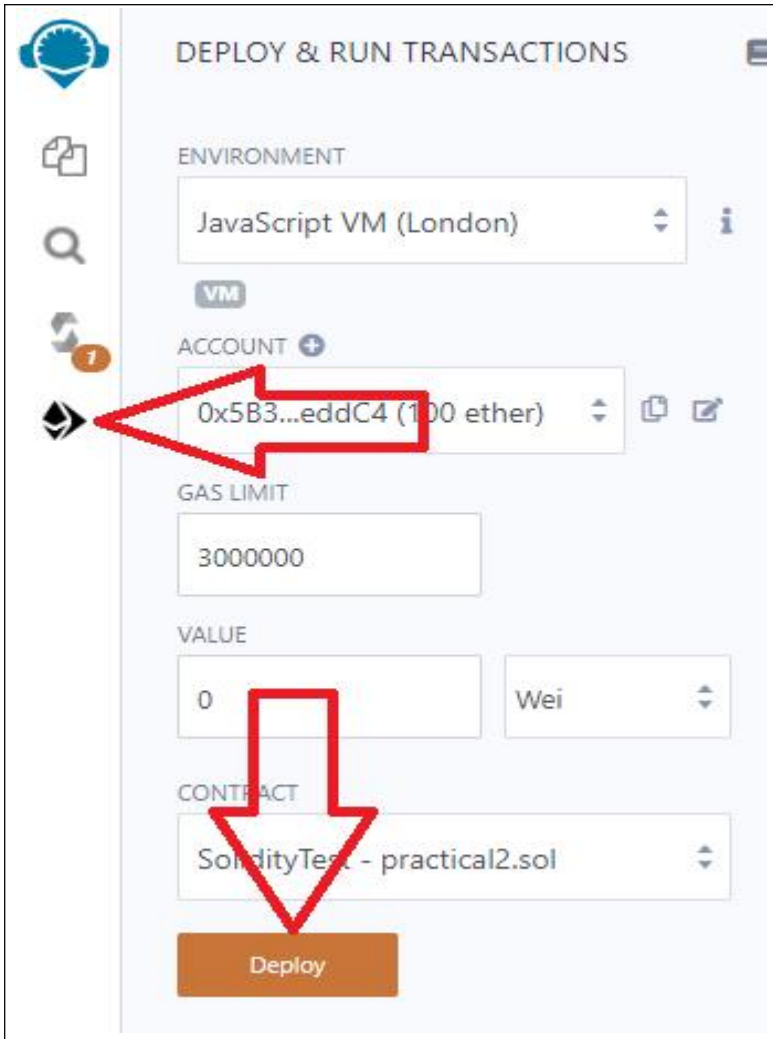
Step 3:Write the below program in new file

```
pragma solidity ^0.5.0;
contract SolidityTest {
uint storedData; // State variable
constructor() public {
storedData = 10;
}
function getResult() public view returns(uint){
uint a = 1; // local variable
uint b = 2;
uint result = a + b;
return result; //access the state variable
}
}
```

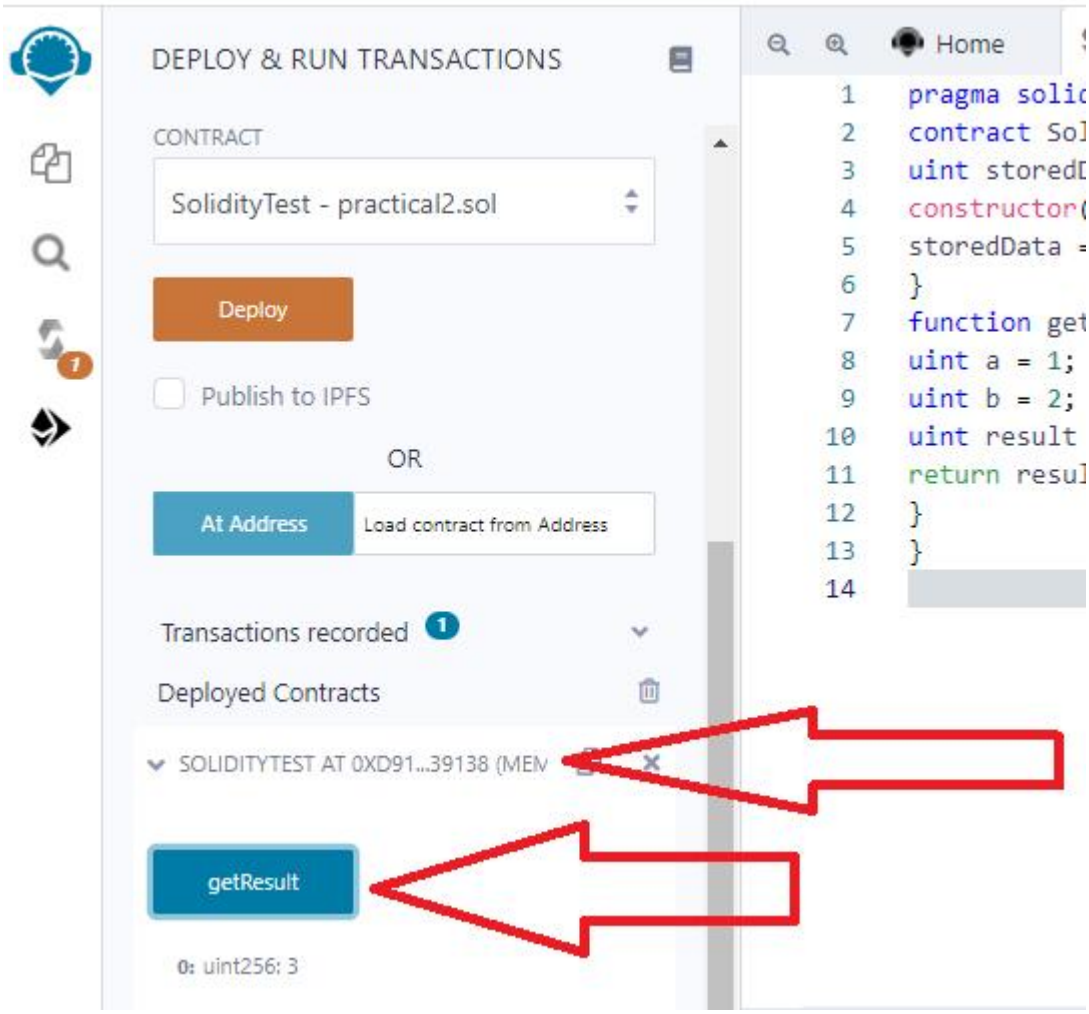
Step 4: Compile contract

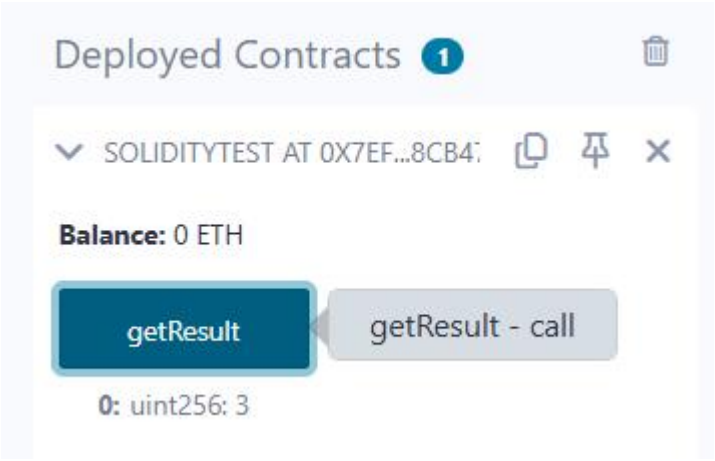


Step 5: Deploy contract



Step 6: Select the contract and click button

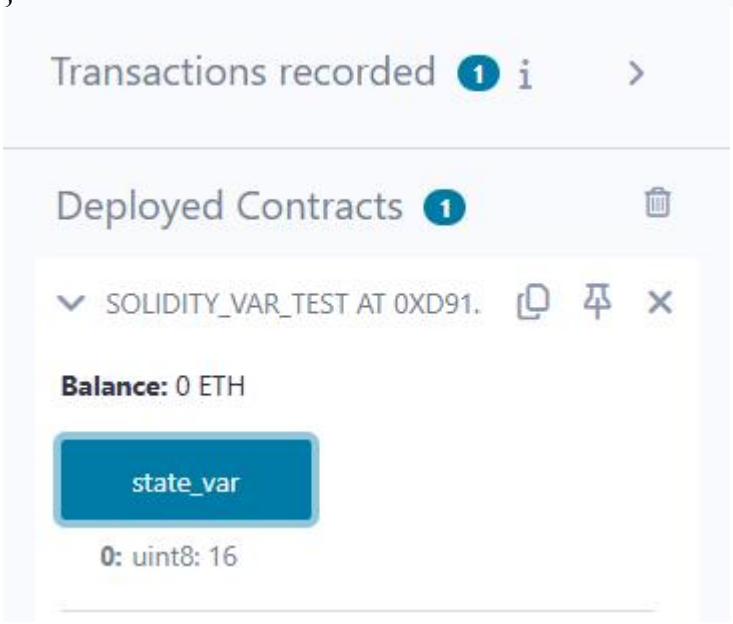




1.State Variable:

```
// Solidity program to
// demonstrate state
// variables
pragma solidity ^0.5.0;
// Creating a contract
contract Solidity_var_Test {
// Declaring a state variable
uint8 public state_var;
// Defining a constructor
```

```
constructor() public {
state_var = 16;
}
}
```



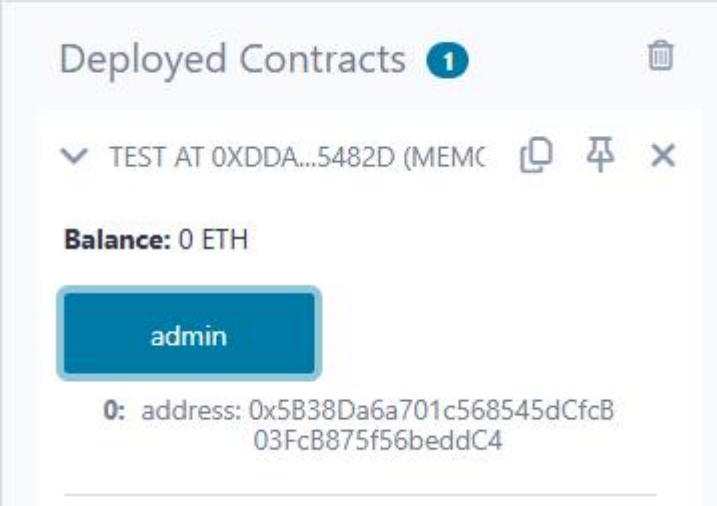
2.Local Variable:

```
// Solidity program to demonstrate
// local variables
pragma solidity ^0.5.0;
// Creating a contract
contract Solidity_var_Test {
// Defining function to show the declaration and
// scope of local variables
function getResult() public view returns(uint){
// Initializing local variables
uint local_var1 = 1;
uint local_var2 = 2;
uint result = local_var1 + local_var2;
// Access the local variable
return result;
}
}
```

3. Global variable:

```
// Solidity program to
```

```
// show Global variables
pragma solidity ^0.5.0;
// Creating a contract
contract Test {
// Defining a variable
address public admin;
// Creating a constructor to
// use Global variable
constructor() public {
admin = msg.sender;
}
}
```



Scope of local variables is limited to function in which they are defined but State variables can have three types of scopes.

Public − Public state variables can be accessed internally as well as via messages. For a public state variable, an automatic getter function is generated.

Internal − Internal state variables can be accessed only internally from the current contract or contract deriving from it without using this.

Private − Private state variables can be accessed only internally from the current contract they are defined not in the derived contract from it.

B)Operators

Solidity supports the following types of operators.

Arithmetic Operators

Comparison Operators

Logical (or Relational) Operators

Assignment Operators

Conditional (or ternary) Operators

1. Arithematic Operator

// Solidity contract to demonstrate

// Arithematic Operator

```
pragma solidity ^0.5.0;
// Creating a contract
contract SolidityTest {
// Initializing variables
```

```
uint16 public a = 20;
uint16 public b = 10;
// Initializing a variable
// with sum
uint public sum = a + b;
// Initializing a variable
// with the difference
uint public diff = a - b;
// Initializing a variable
// with product
uint public mul = a * b;
// Initializing a variable
// with quotient
uint public div = a / b;
// Initializing a variable
// with modulus
uint public mod = a % b;
// Initializing a variable
// decrement value
uint public dec = --b;
// Initializing a variable
// with increment value
uint public inc = ++a;
}
```

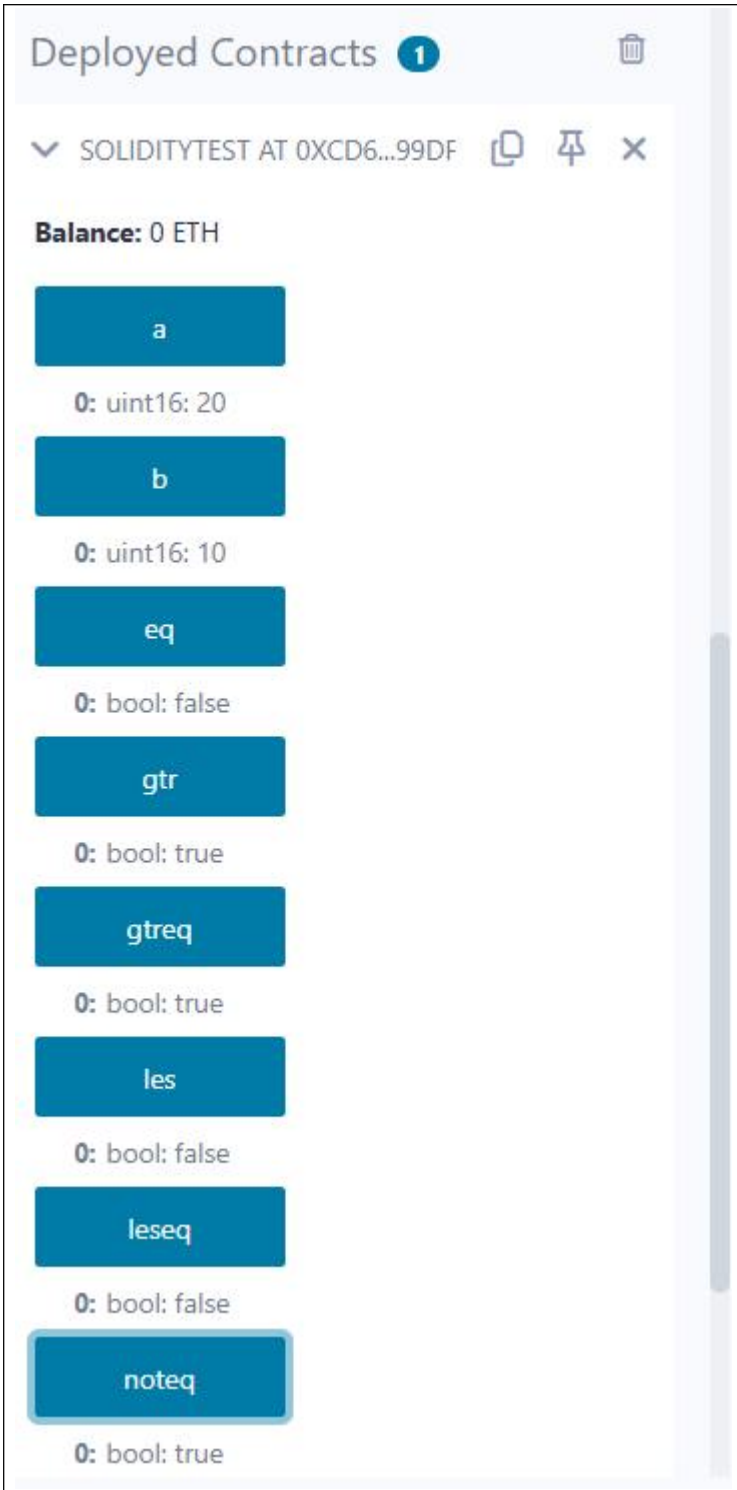
2.Relational Operator

// Solidity program to demonstrate

// Relational Operator

```
pragma solidity ^0.5.0;

// Creating a contract
contract SolidityTest {

// Declaring variables
uint16 public a = 20;
uint16 public b = 10;
```

```
// Initializing a variable
// with bool equal result
bool public eq = a == b;

// Initializing a variable
// with bool not equal result
bool public noteq = a != b;

// Initializing a variable
// with bool greater than result
bool public gtr = a > b;

// Initializing a variable
// with bool less than result
bool public les = a < b;

// Initializing a variable
// with bool greater than equal to result
bool public gtreq = a >= b;

// Initializing a variable
// bool less than equal to result
bool public leseq = a <= b;
}
```



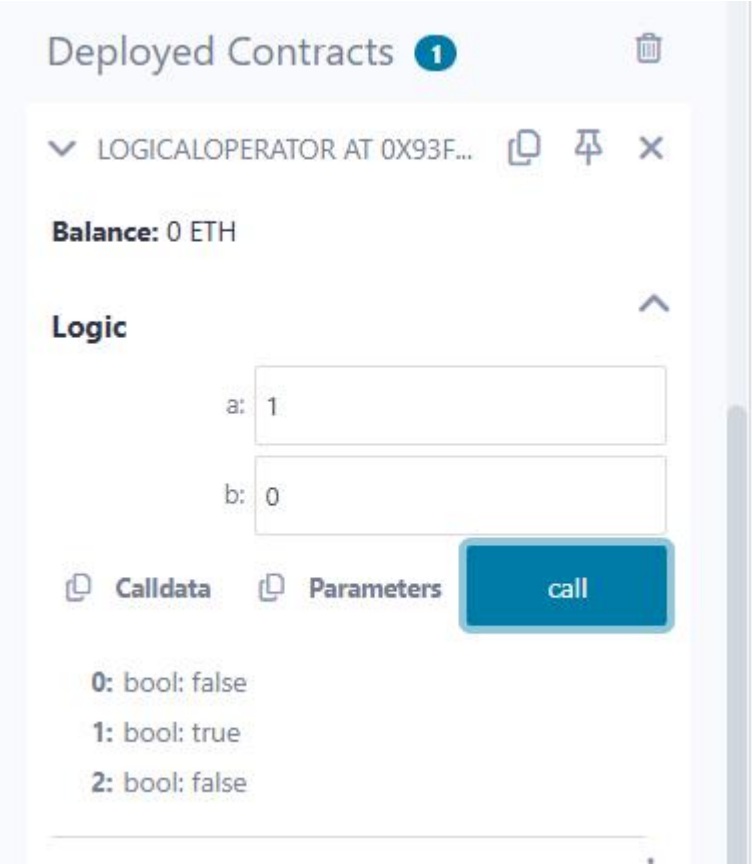
3.Logical Operators

// Solidity program to demonstrate

// Logical Operators

```
pragma solidity ^0.5.0;

// Creating a contract
contract logicalOperator{

// Defining function to demonstrate
// Logical operator
function Logic(
bool a, bool b) public view returns(
bool, bool, bool){
```

```
// Logical AND operator
bool and = a&&b;

// Logical OR operator
bool or = a||b;

// Logical NOT operator
bool not = !a;
return (and, or, not);
}
}
```



4.Bitwise Operators

// Solidity program to demonstrate

// Bitwise Operator

```
pragma solidity ^0.5.0;

// Creating a contract
contract SolidityTest {

// Declaring variables
uint16 public a = 20;
uint16 public b = 10;

// Initializing a variable
// to '&' value
```

```
uint16 public and = a & b;

// Initializing a variable
// to '|' value
uint16 public or = a | b;

// Initializing a variable
// to '^' value
uint16 public xor = a ^ b;

// Initializing a variable
// to '<<' value
uint16 public leftshift = a << b;

// Initializing a variable
// to '>>' value
uint16 public rightshift = a >> b;

// Initializing a variable
// to '~' value
uint16 public not = ~a ;

}
```

5.Assignment Operator

// Solidity program to demonstrate

// Assignment Operator

```
pragma solidity ^0.5.0;

// Creating a contract
contract SolidityTest {

// Declaring variables
uint16 public assignment = 20;
```

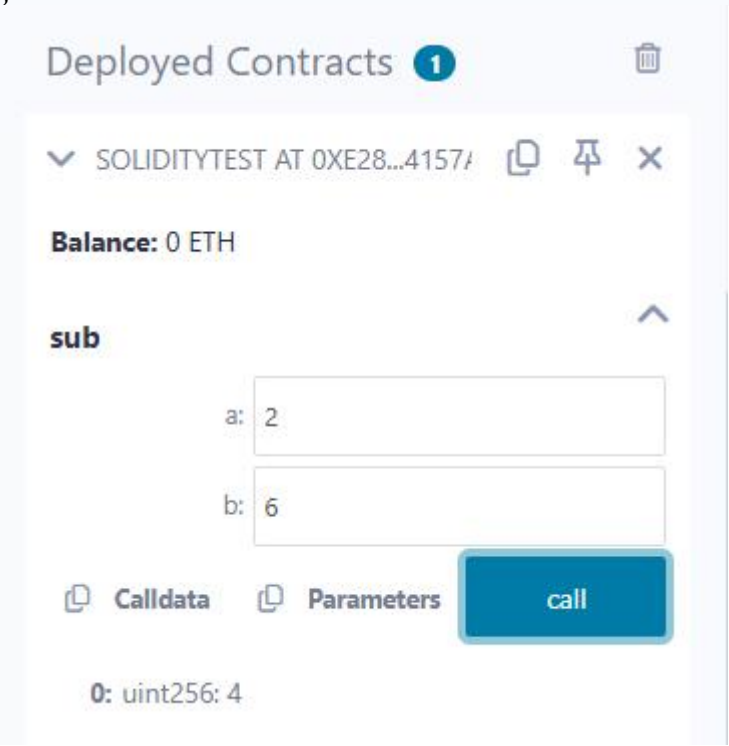
```
uint public assignment_add = 50;
uint public assign_sub = 50;
uint public assign_mul = 10;
uint public assign_div = 50;
uint public assign_mod = 32;

// Defining function to
// demonstrate Assignment Operator
function getResult() public{
assignment_add += 10;
assign_sub -= 20;
assign_mul *= 10;
assign_div /= 10;
assign_mod %= 20;
return ;
}
}
```

6.**Conditional Operators**

// Solidity program to demonstrate

// Conditional Operator

```
pragma solidity ^0.5.0;

// Creating a contract
contract SolidityTest{
// Defining function to demonstrate
// conditional operator
function sub(
uint a, uint b) public view returns(
uint){
uint result = (a > b? a-b : b-a);
return result;
}
}
```



C)Loops:

1.While loop: The most basic loop in Solidity is the **while** loop which would be discussed in this chapter. The purpose of a **while** loop is to execute a statement or code block repeatedly as long as an **expression** is true. Once the expression becomes **false,** the loop terminates.

2.do-while loop: The **do...while** loop is similar to the **while** loop except that the condition check happens at the end of the loop. This means that the loop will always be executed at least once, even if the condition is **false**.

3.for loop: The **for** loop is the most compact form of looping. It includes the following three important parts −

The **loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.

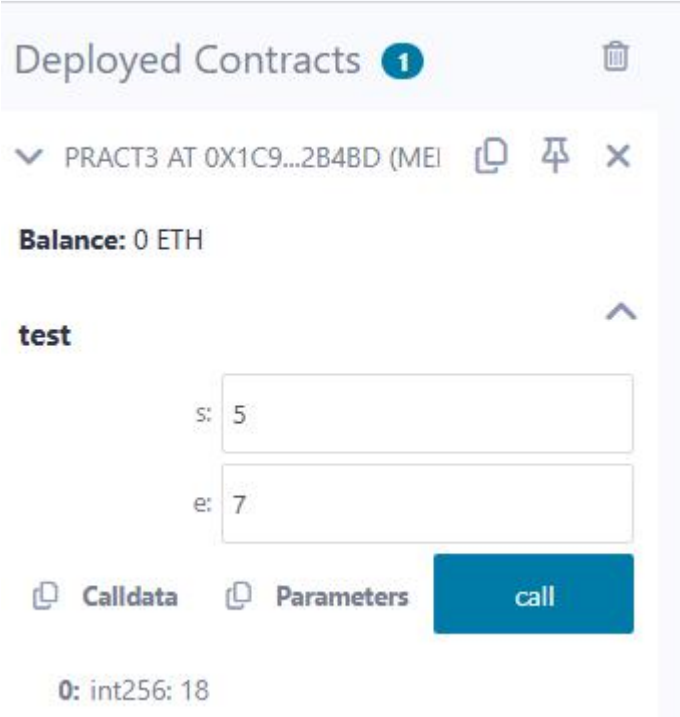
The **test statement** which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.

The **iteration statement** where you can increase or decrease your counter.

4.loop control: Solidity provides full control to handle loops and switch statements. There may be a situation when you need to come out of a loop without reaching its bottom. There may also be a situation when you want to skip a part of your code block and start the next iteration of the loop.To handle all such situations, Solidity provides **break** and **continue** statements. These statements are used to immediately come out of any loop or to start the next iteration of any loop respectively.

1.While Loop

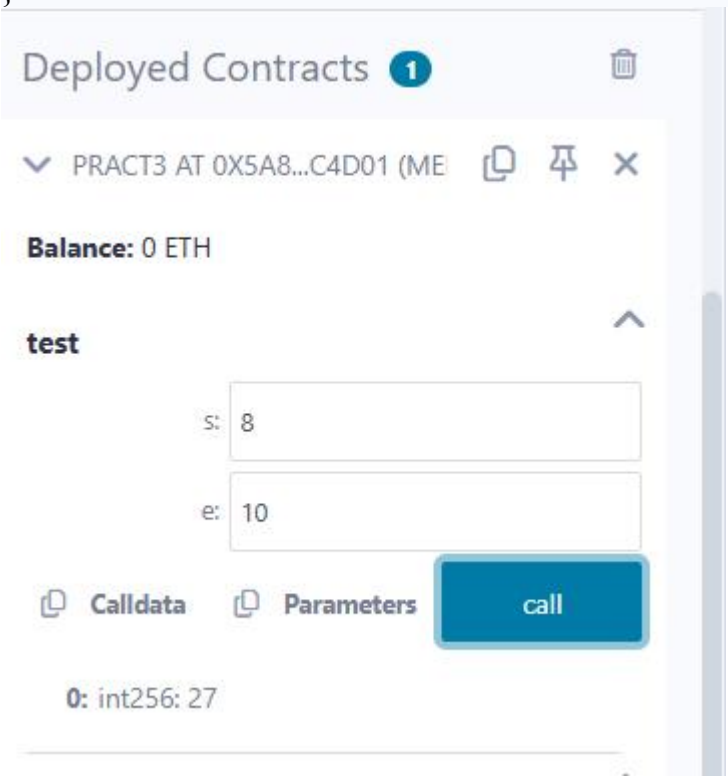
```
pragma solidity ^0.5.0;
contract Pract3{
function test(int s, int e) public view returns(int)
{
int i;
int sum=0;
i=s;
while(i<=e)
{
sum+=i;  //sum=sum+i;
i++;
}
return sum;
}
}
```



2.Do-while loop:

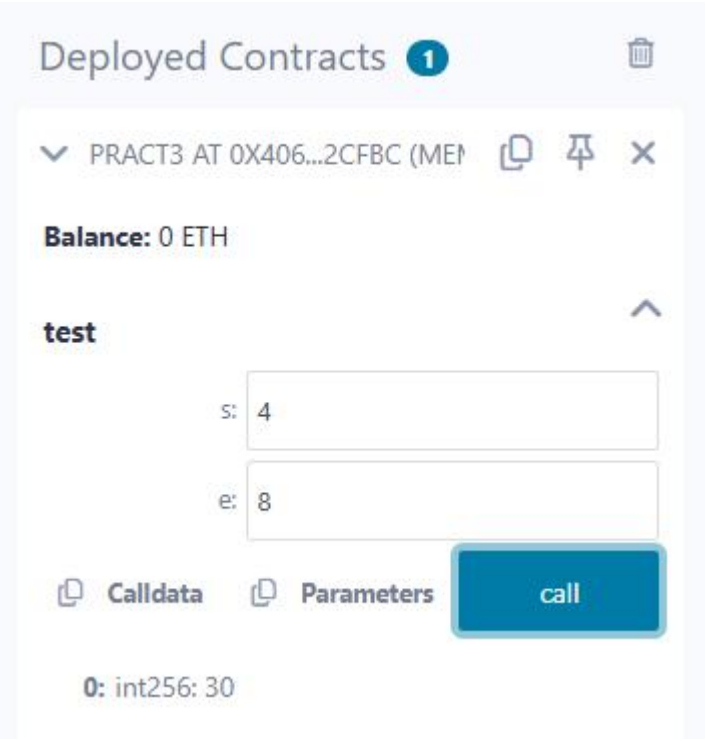
```
pragma solidity ^0.5.0;
contract Pract3{
function test(int s, int e) public view returns(int)
{
int i;
int sum=0;
i=s;
do
{
sum+=i;  //sum=sum+i;
i++;
```

```
}while(i<=e);
return sum;
}
}
```



3.For Loop:

```
contract Pract3{
function test(int s, int e) public view returns(int)
{
int i;
int sum=0;
for(i=s;i<=e;i++)
{
sum+=i;  //sum=sum+i;
}
return sum;
}
}
```

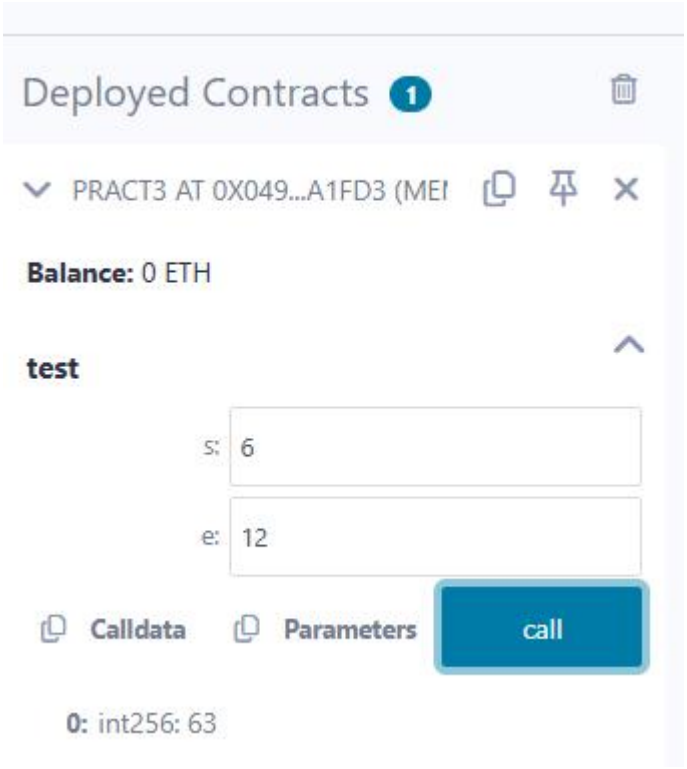


4.loop Control: (Break statement)

```
pragma solidity ^0.5.0;

contract SolidityTest {
uint storedData;
constructor() public{
storedData = 10;
}
function getResult() public view returns(string memory){
uint a = 1;
uint b = 2;
uint result = a + b;
return integerToString(result);
}
function integerToString(uint _i) internal pure
returns (string memory) {

if (_i == 0) {
return "0";
}
uint j = _i;
uint len;

while (true) {
len++;
j /= 10;
if(j==0){
break;   //using break statement
}
}
bytes memory bstr = new bytes(len);
uint k = len - 1;

while (_i != 0) {
bstr[k--] = byte(uint8(48 + _i % 10));
```

```
_i /= 10;
}
return string(bstr);
}
}
```

(continue statement)




```
pragma solidity ^0.5.0;
contract SolidityTest {
uint storedData;
constructor() public{
storedData = 10;
}
function getResult() public view returns(string memory){
uint n = 1;
uint sum = 0;

while( n < 10){
n++;
if(n == 5){
continue; // skip n in sum when it is 5.
}
sum = sum + n;
}
return integerToString(sum);
}
function integerToString(uint _i) internal pure
returns (string memory) {

if (_i == 0) {
return "0";
}
uint j = _i;
uint len;
```

```
while (true) {
len++;
j /= 10;
if(j==0){
break;   //using break statement
}
}
bytes memory bstr = new bytes(len);
uint k = len - 1;

while (_i != 0) {
bstr[k--] = byte(uint8(48 + _i % 10));
_i /= 10;
}
return string(bstr);
}
}
```

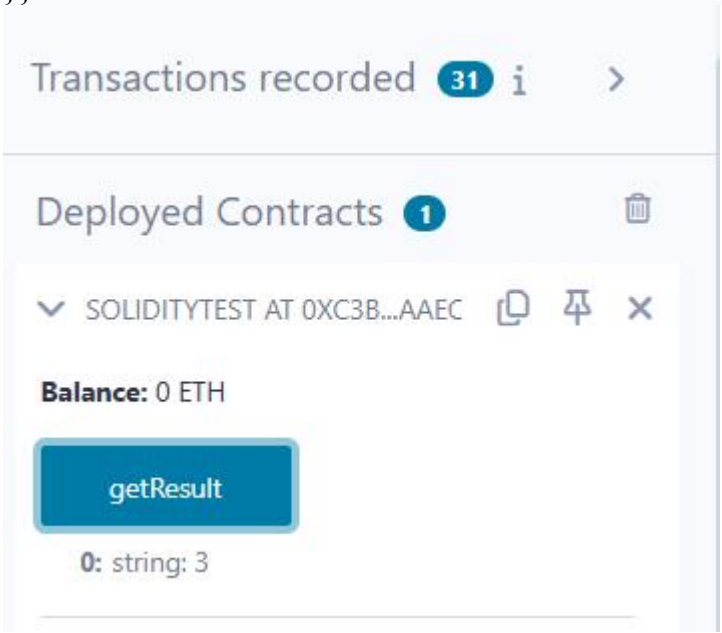
D) Decision Making:

While writing a program, there may be a situation when you need to adopt one out of a given set of paths. In such cases, you need to use conditional statements that allow your program to make correct decisions and perform right actions.Solidity supports conditional statements which are used to perform different actions based on different conditions. Here we will explain the **if..else** statement.

1. if statement: The **if** statement is the fundamental control statement that allows Solidity to make decisions and execute statements conditionally.

```
pragma solidity ^0.5.0;

contract SolidityTest {
uint storedData;
constructor() public {
storedData = 10;
}
function getResult() public view returns(string memory){
uint a = 1;
uint b = 2;
uint result = a + b;
return integerToString(result);
}
function integerToString(uint _i) internal pure
```

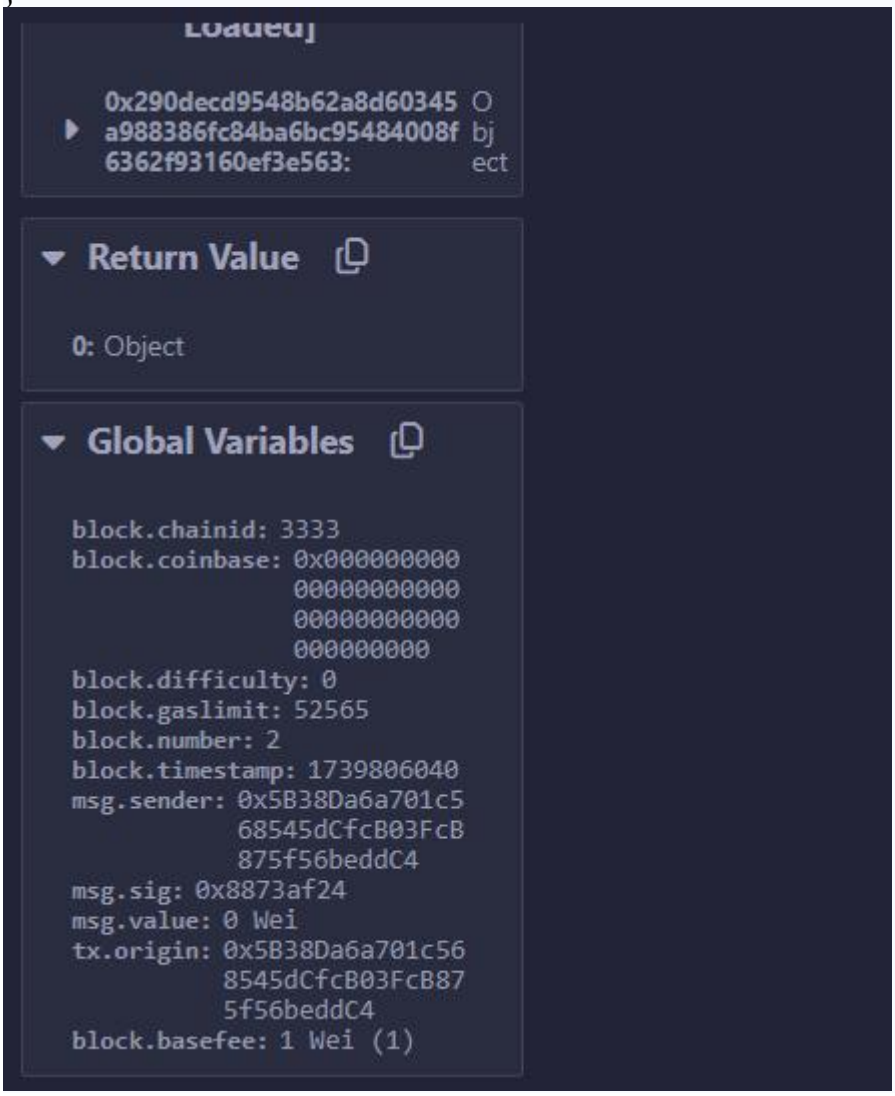
```
returns (string memory) {
if (_i == 0) {   // if statement
return "0";
}
uint j = _i;
uint len;

while (j != 0) {
len++;
j /= 10;
}
bytes memory bstr = new bytes(len);
uint k = len - 1;

while (_i != 0) {
bstr[k--] = byte(uint8(48 + _i % 10));
_i /= 10;
}
return string(bstr);//access local variable
}}
```



2.if-else statement: The **'if...else'** statement is the next form of control statement that allows Solidity to execute statements in a more controlled way.

```
pragma solidity ^0.5.0;

// Creating a contract
contract Types {
// Declaring state variables
uint i = 10;
bool even;

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

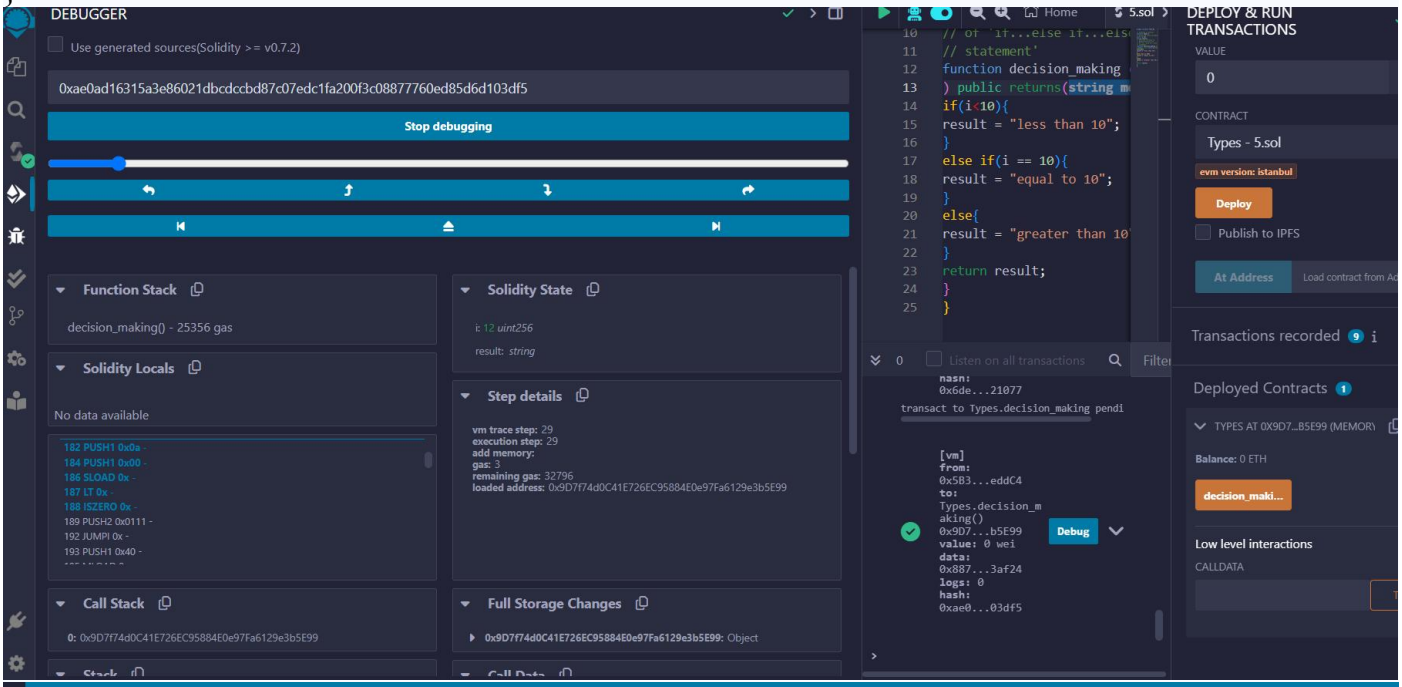
```
}
else{
even = false;
}
return even;
}
}
```

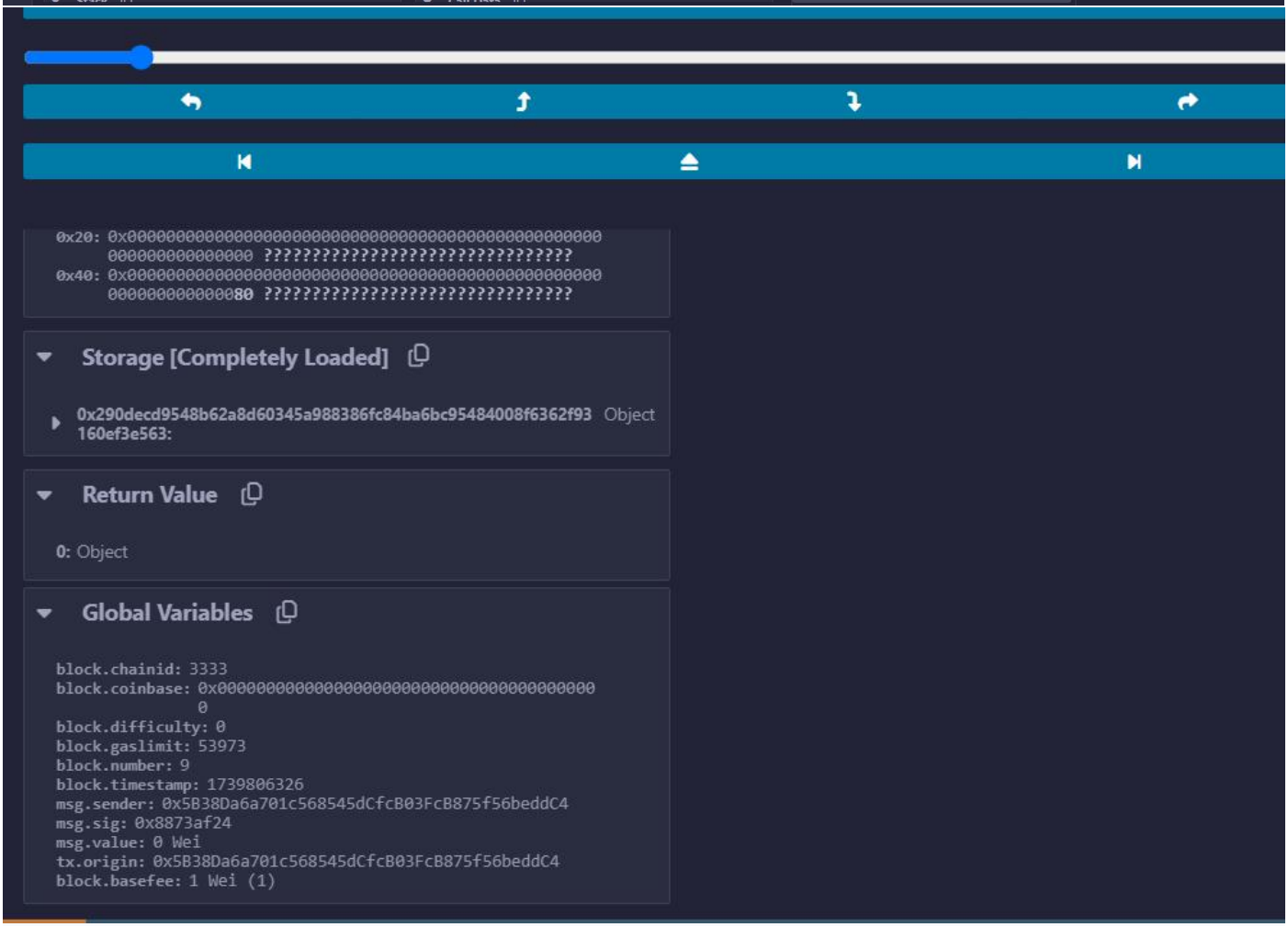


3.if-else..if statement: The **if...else if...** statement is an advanced form of **if...else** that allows Solidity to make a correct decision out of several conditions.

```
pragma solidity ^0.5.0;

// Creating a contract
contract Types {
// Declaring state variables
uint i = 12;
string result;
// Defining function to
// demonstrate the use
// of 'if...else if...else
// statement'
function decision_making (
) public returns(string memory){
if(i<10){
result = "less than 10";
}
```

```
else if(i == 10){
result = "equal to 10";
}
else{
result = "greater than 10";
}
return result;
}
}
```





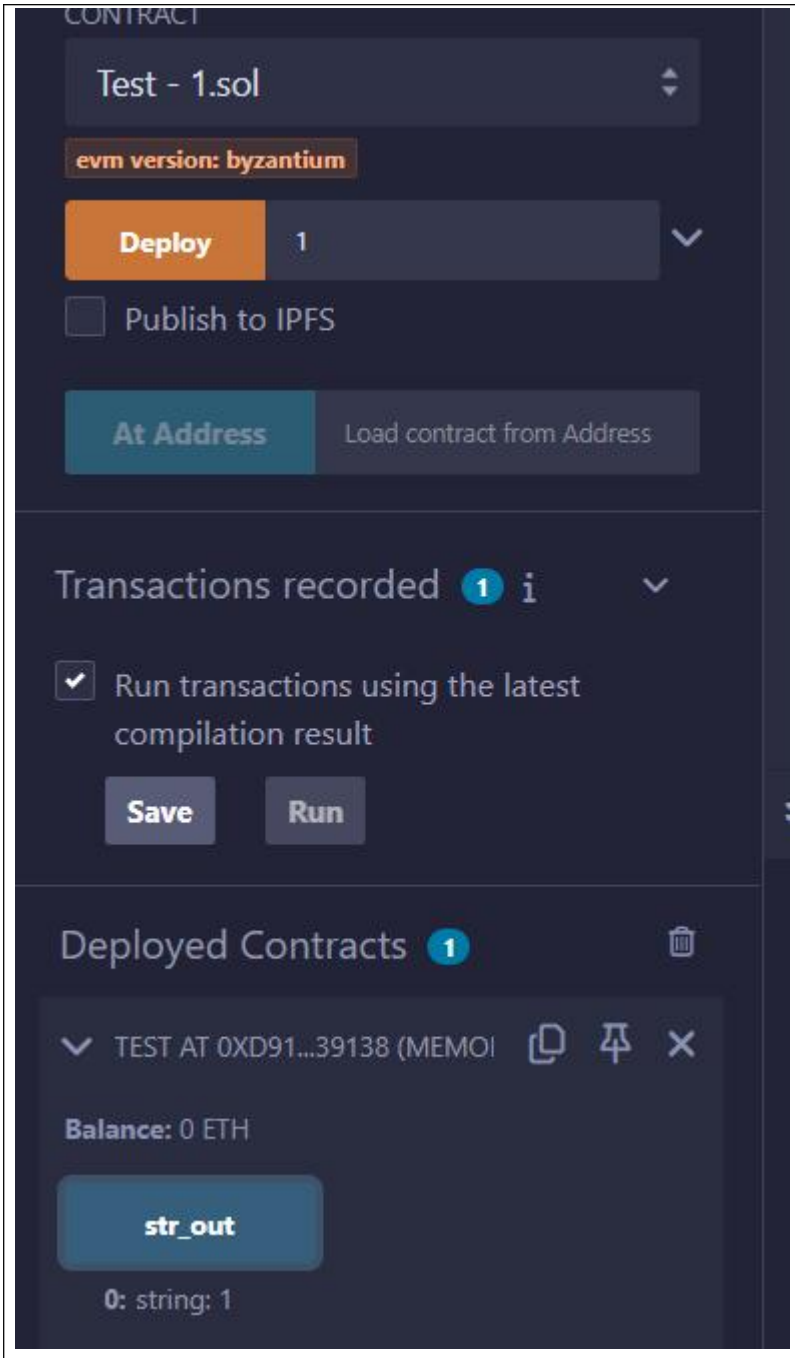
String:

// Solidity program to demonstrate

// how to create a contract

```
pragma solidity ^0.4.23;

// Creating a contract
contract Test {
// Declaring variable
string  str;

// Defining a constructor
constructor(string str_in){
str = str_in;
}
// Defining a function to
// return value of variable 'str'
function str_out() public view returns(string memory){
return str;
}
}
```

Note: after deploy it asked u to enter string then enter string over there and then see the output after clicking on str_out button


CONTRACT
Test - 1.sol
evm version: byzantium
Deploy
1
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 1
Run transactions using the latest compilation result
Save
Run
Deployed Contracts 1
TEST AT 0XD91...39138 (MEMOI
Balance: 0 ETH
str_out
0: string: 1

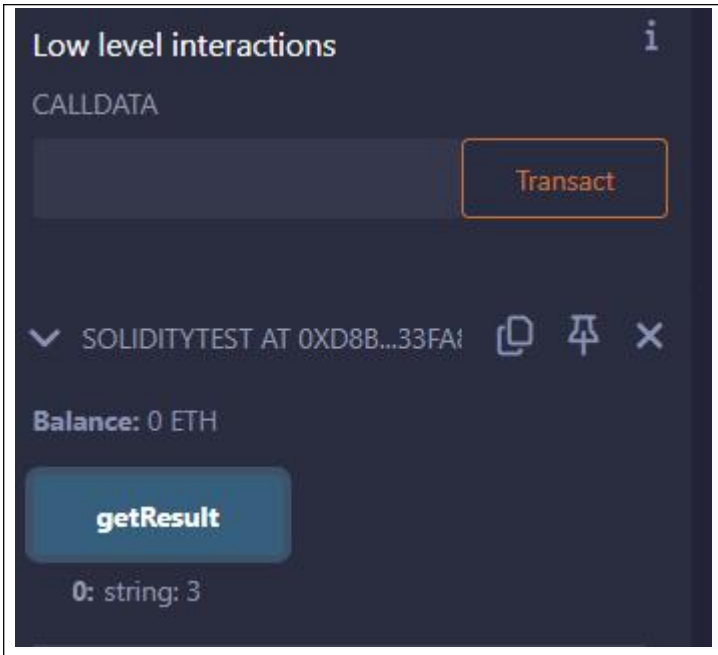
PRACTICAL 5a

Aim:- WRITE A SOLIDITY PROGRAM FOR STRING, ARRAYS, ENUMS, STRUCTURE & MAPPINGS.

A) String:

Solidity supports String literal using both double quote (") and single quote ('). It provides string as a data type to declare a variable of type String.(Int to str)

```
pragma solidity ^0.5.0;

contract SolidityTest {
constructor() public{
}
function getResult() public view returns(string memory){
uint a = 1;
uint b = 2;
uint result = a + b;
return integerToString(result);
}
function integerToString(uint _i) internal pure
returns (string memory) {

if (_i == 0) {
return "0";
}
uint j = _i;
uint len;

while (j != 0) {
len++;
j /= 10;
}
bytes memory bstr = new bytes(len);
uint k = len - 1;

while (_i != 0) {
bstr[k--] = byte(uint8(48 + _i % 10));
_i /= 10;
}
return string(bstr);
}
}
```



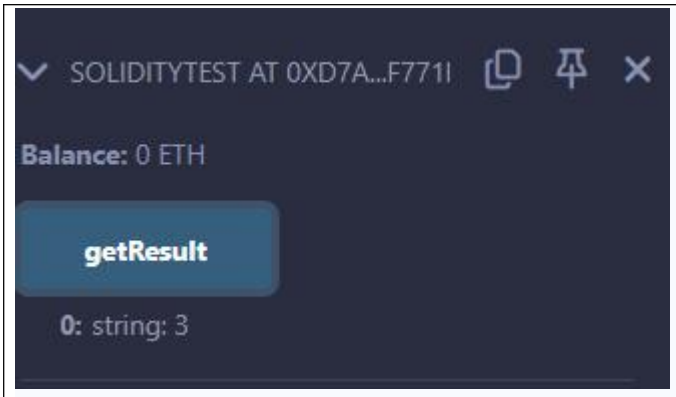
B)Array:

Array is a data structure, which stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

// Solidity program to demonstrate

// accessing elements of an array

```
pragma solidity ^0.5.0;
function
// Creating a contract
contract Types {

// Declaring an array
uint[6] data;
uint x;

// Defining function to
// assign values to array
function array_example() public returns (uint[6] memory)
{

data  = [uint(10), 20, 30, 40, 50, 60];
}
function result() public view returns(uint[6] memory){
return data;
}
// Defining function to access
// values from the array
// from a specific index
function array_element() public view returns (uint){
uint x = data[2];
return x;
}
}
```



C)Enums:

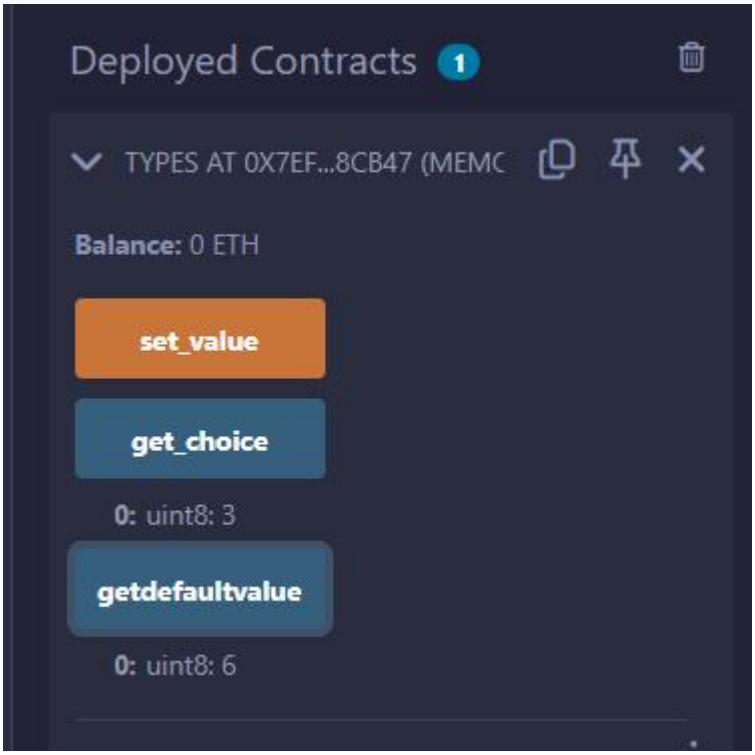
Enums restrict a variable to have one of only a few predefined values. The values in this enumerated list are called enums. With the use of enums it is possible to reduce the number of bugs in your code.

// Solidity program to demonstrate

// how to use 'enumerator'

```
pragma solidity ^0.5.0;

// Creating a contract
contract Types {

// Creating an enumerator
enum week_days
{
Monday,
Tuesday,
Wednesday,
Thursday,
Friday,
Saturday,
Sunday
}
// Declaring variables of
// type enumerator
week_days week;

week_days choice;

// Setting a default value
week_days constant default_value
= week_days.Sunday;

// Defining a function to
// set value of choice
function set_value() public {
choice = week_days.Thursday;
}
// Defining a function to
// return value of choice
function get_choice(
) public view returns (week_days) {
return choice;
}
```

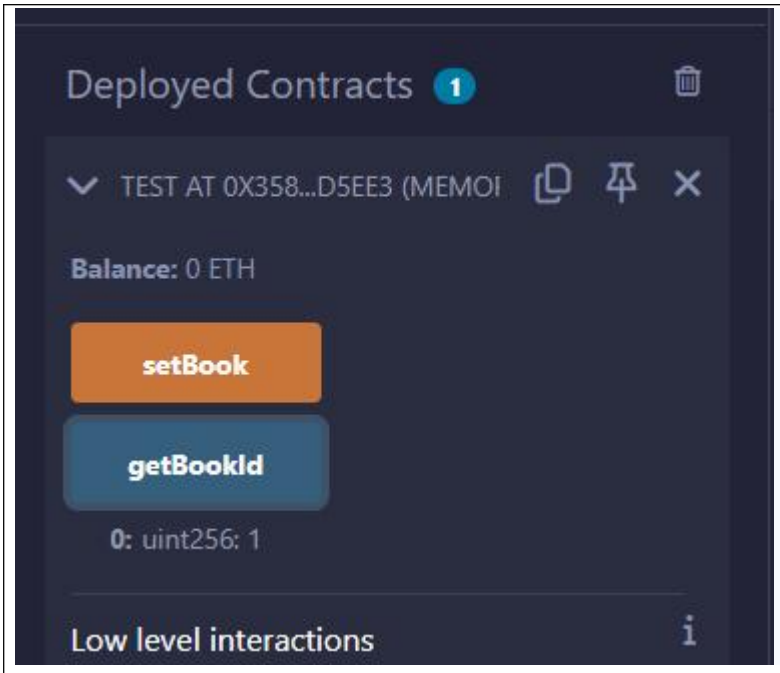
```
// Defining function to
// return default value
function getdefaultvalue(
) public pure returns(week_days) {
return default_value;
}
}
```



D)Structure:

Struct types are used to represent a record.

```
pragma solidity ^0.5.0;

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

function setBook() public {
book = Book('Learn Java', 'TP', 1);
}
function getBookId() public view returns (uint) {
return book.book_id;
}
}
```



E)Mappings:

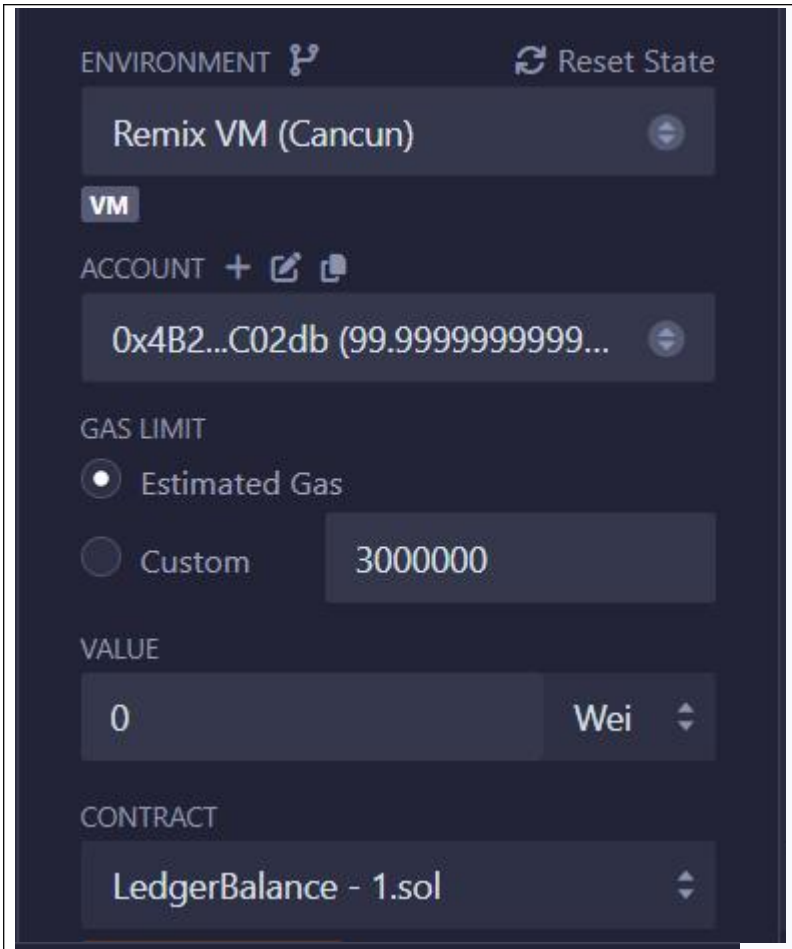
Mapping is a reference type as arrays and structs. Following is the syntax to declare a mapping type.

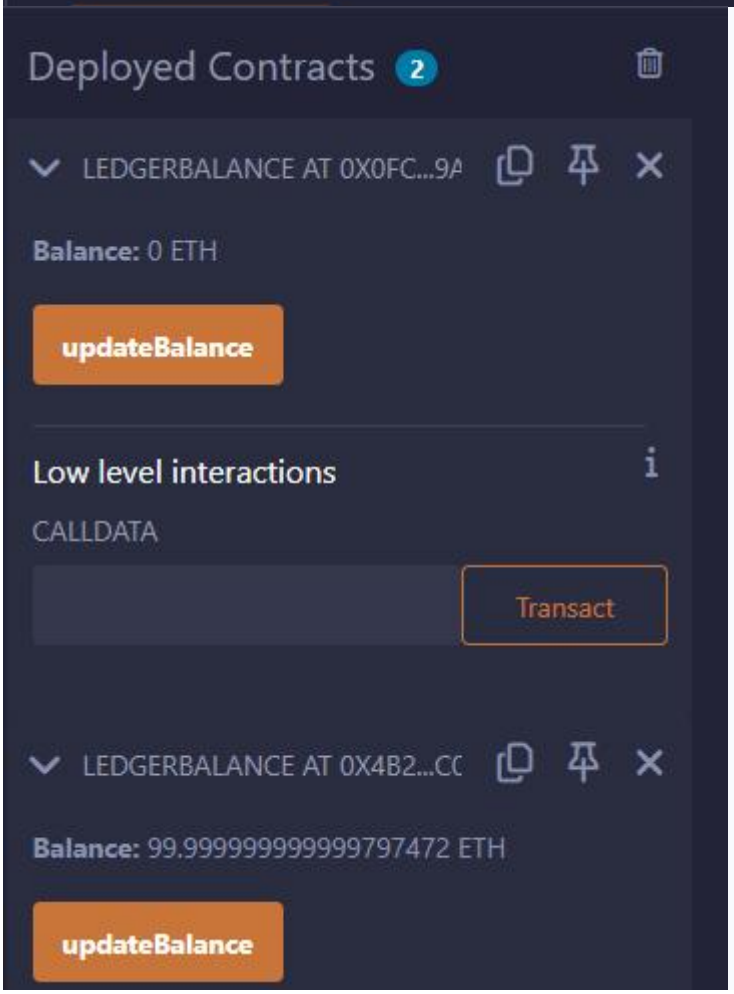
mapping(_KeyType => _ValueType) where ,

_KeyType − can be any built-in types plus bytes and string. No reference type or complex objects are allowed.

_ValueType − can be any type.

```
pragma solidity ^0.5.0;

contract LedgerBalance {
mapping(address => uint)  balance;

function updateBalance() public  returns(uint) {
balance[msg.sender]=30;
return balance[msg.sender];
}
}
```

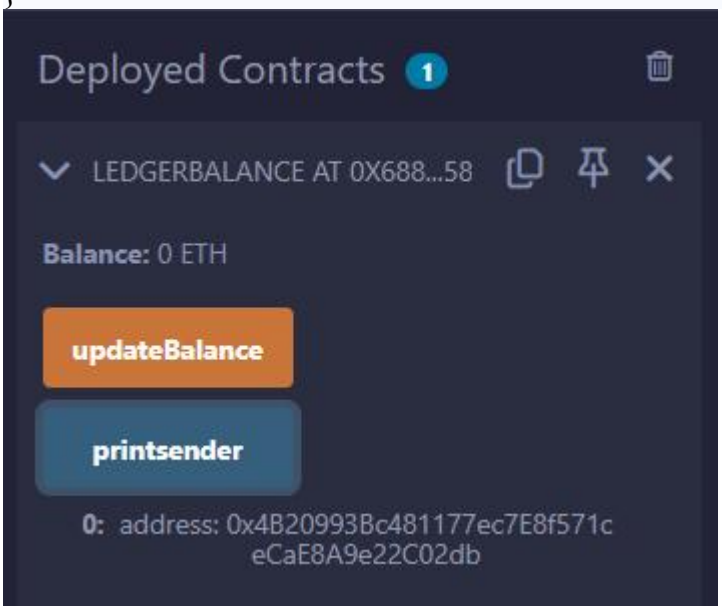




Mapping program for String.

```
pragma solidity ^0.5.0;
```

```
contract LedgerBalance {
mapping(address => string)  name;

function updateBalance() public returns(string memory){
name[msg.sender] = "Mrunali";
return name[msg.sender];
}
function printsender() public view returns(address) {
return msg.sender;
}
}
```



PRACTICAL 5B

Aim: WRITE A SOLIDITY PROGRAM FOR FUNCTION OVERLOADING, MATHEMATICAL FUNCTION & CRYPTOGRAPHIC FUNCTIONS.

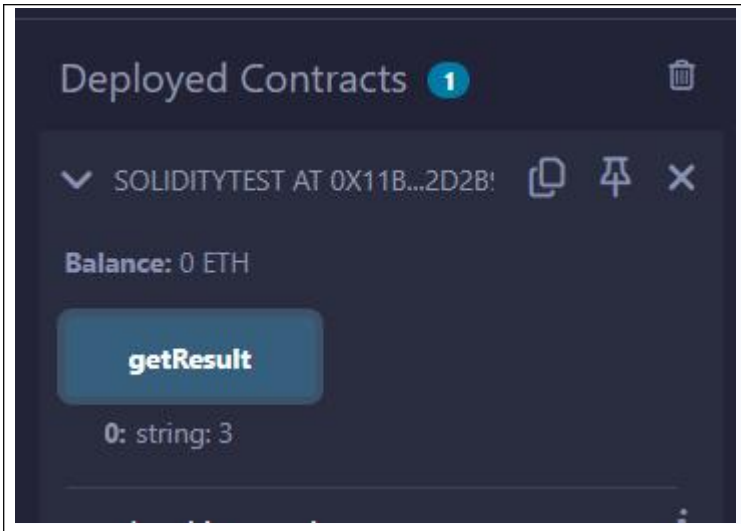
Functions, Function Modifiers, View functions, Pure Functions, Fallback Function, Function Overloading, Mathematical functions, Cryptographic functions.

AIM: WRITE A SOLIDITY PROGRAM FOR FUNCTION, VIEW FUNCTION, PURE FUNCTION & FALLBACK FUNCTION.

A)Function:

A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. It helps programmers in writing modular codes. Functions allow a programmer to divide a big program into a number of small and manageable functions.

```
pragma solidity ^0.5.0;

contract SolidityTest {
constructor() public{
}
function getResult() public view returns(string memory){
uint a = 1;
uint b = 2;
uint result = a + b;
return integerToString(result);
}
function integerToString(uint _i) internal pure
returns (string memory) {

if (_i == 0) {
return "0";
}
uint j = _i;
uint len;

while (j != 0) {
len++;
j /= 10;
}
bytes memory bstr = new bytes(len);
uint k = len - 1;

while (_i != 0) {
bstr[k--] = byte(uint8(48 + _i % 10));
_i /= 10;
}
return string(bstr);//access local variable

}
}
```



B)View Function:

View functions ensure that they will not modify the state. A function can be declared as **view**. Getter method are by default view functions.

C)Pure Function:

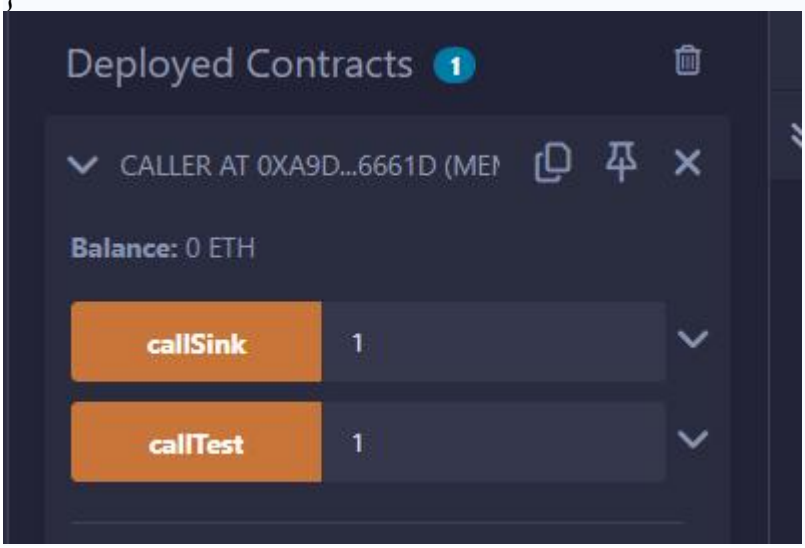
Pure functions ensure that they not read or modify the state. A function can be declared as **pure**. Pure functions can use the revert() and require() functions to revert potential state changes if an error occurs.

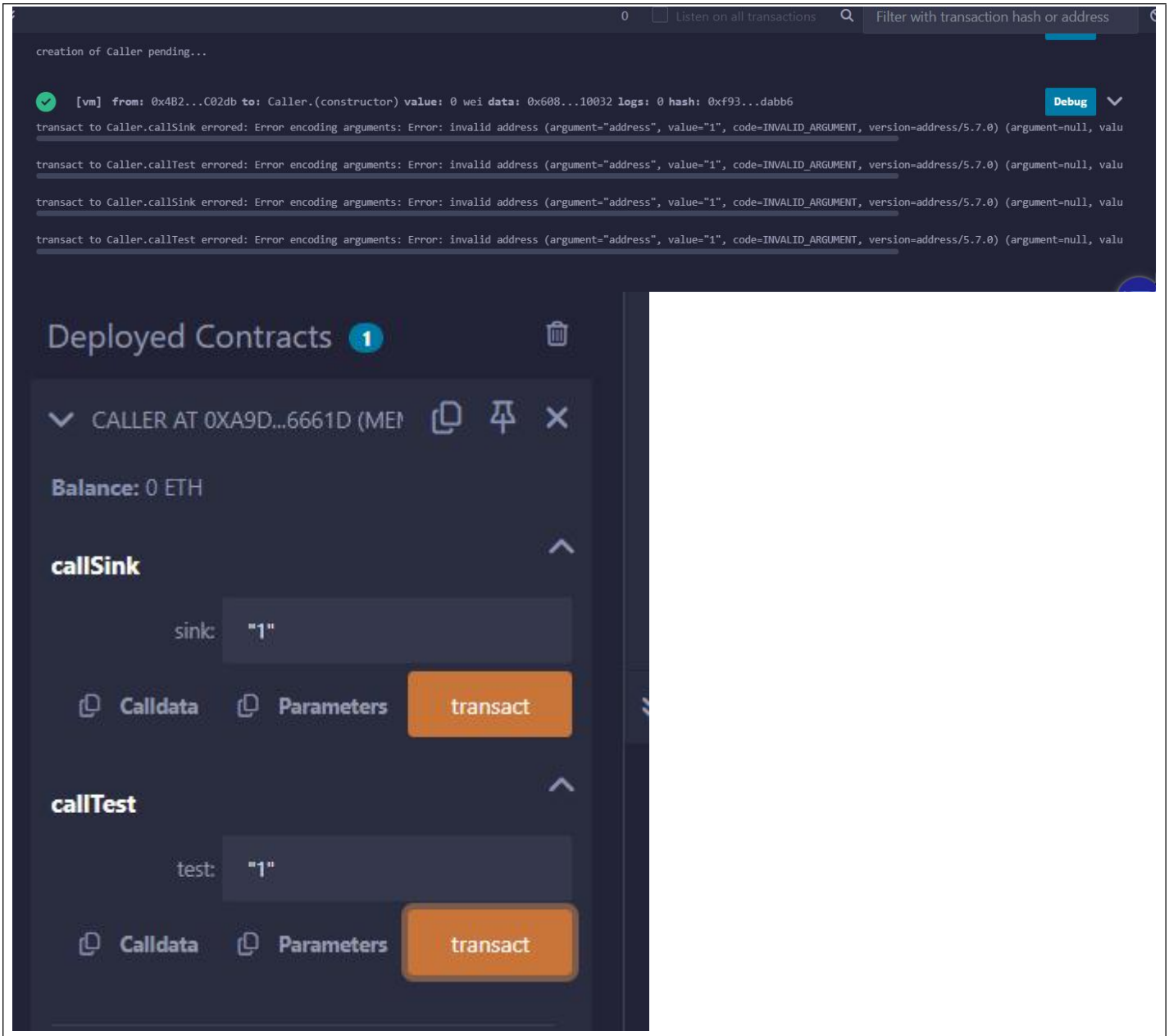
```
pragma solidity ^0.5.0;
contract Test {
int public x=10; //global
int y=90;//state
function f1() public returns(int){
   //read and update is allowed
   x=100;
return x;
}
function f2() public view returns(int){
  //  x=100; //erro beacuse x is global/state
  //we can access but we cannot update state or global variable int view function
return x;
}
function f3() public pure returns(int){
   //we cannot access or update state or global variable in pure function
   int z=80;
return z;
}


}
```

D)Fallback Function:

Fallback function is a special function available to a contract.

```
pragma solidity ^0.5.0;
contract Test {
uint public x ;
function() external { x = 1; }
}
contract Sink {
function() external payable { }
}
contract Caller {
function callTest(Test test) public returns (bool) {
(bool success,) = address(test).call(abi.encodeWithSignature("nonExistingFunction()"));
require(success);
// test.x is now 1
address payable testPayable = address(uint160(address(test)));
// Sending ether to Test contract,
// the transfer will fail, i.e. this returns false here.
return (testPayable.send(2 ether));
}
function callSink(Sink sink) public returns (bool) {
address payable sinkPayable = address(sink);
return (sinkPayable.send(2 ether));
}
}
```




0
Listen on all transactions
Filter with transaction hash or address
creation of Caller pending...
[vm] from: 0x4B2...C02db to: Caller.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0xf93...dabb6
Debug
transact to Caller.callSink errored: Error encoding arguments: Error: invalid address (argument="address", value="1", code=INVALID_ARGUMENT, version=address/5.7.0) (argument=null, valu
transact to Caller.callTest errored: Error encoding arguments: Error: invalid address (argument="address", value="1", code=INVALID_ARGUMENT, version=address/5.7.0) (argument=null, valu
transact to Caller.callSink errored: Error encoding arguments: Error: invalid address (argument="address", value="1", code=INVALID_ARGUMENT, version=address/5.7.0) (argument=null, valu
transact to Caller.callTest errored: Error encoding arguments: Error: invalid address (argument="address", value="1", code=INVALID_ARGUMENT, version=address/5.7.0) (argument=null, valu
Deployed Contracts 1
CALLER AT 0XA9D...6661D (MEI
Balance: 0 ETH
callSink
sink: "1"
Calldata
Parameters
transact
callTest
test: "1"
Calldata
Parameters
transact

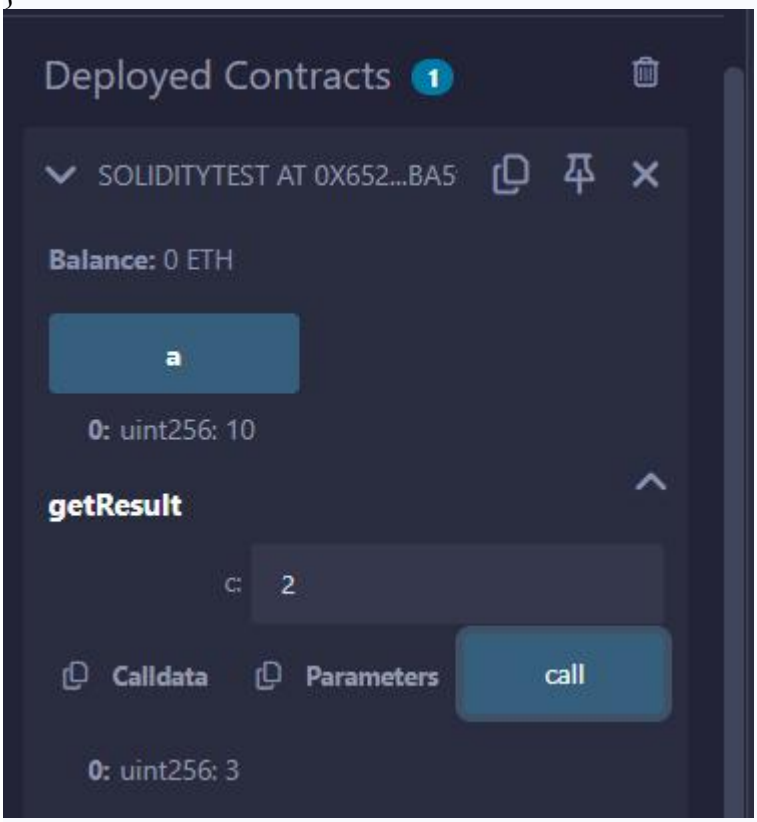
PRACTICAL 6

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

PRACTICAL 6a

6a. Withdrawal Pattern, Restricted Access.
Solidity
6a

```
pragma solidity ^0.5.0;
contract SolidityTest {
uint storedData; // State variable
uint public a=10;
constructor() public {
storedData = 10;
}
function getResult(uint c) public view returns(uint){
uint a = 1; // local variable
uint b = 2;
uint result = a + b;
return result; //access the state variable
}
}
```

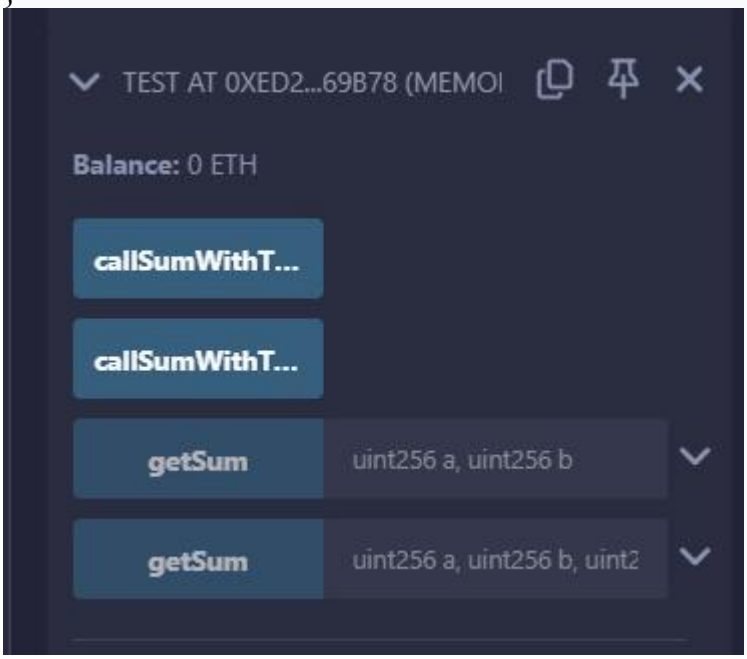


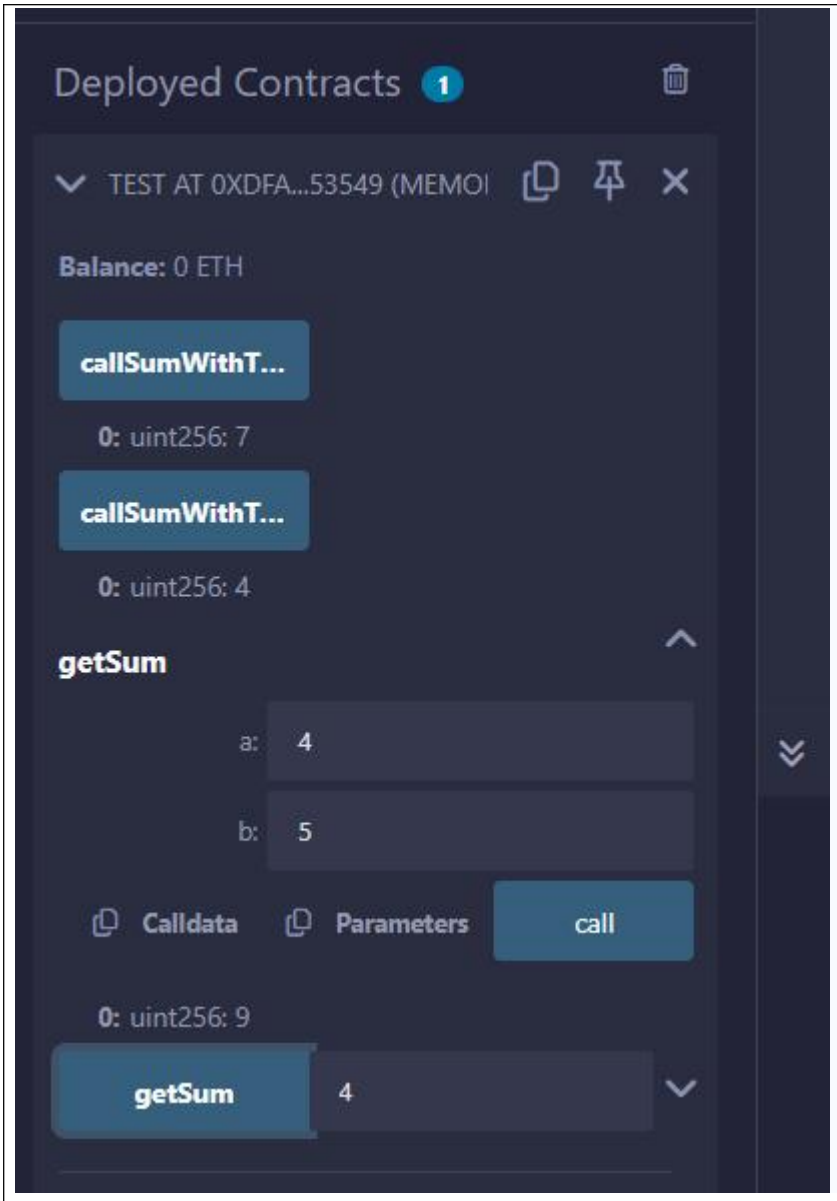
Function Overloading:

The definition of the function must differ from each other by the types and/or the number of arguments in the argument list. You cannot overload function declarations that differ only by return type.

```
pragma solidity ^0.5.0;

contract Test {
function getSum(uint a, uint b) public pure returns(uint){
```

```
return a + b;
}
function getSum(uint a, uint b, uint c ) public pure returns(uint){
return a + b + c;
}
function callSumWithTwoArguments() public pure returns(uint){
return getSum(2,2);
}
function callSumWithThreeArguments() public pure returns(uint){
return getSum(1,2,4);
}

}
```

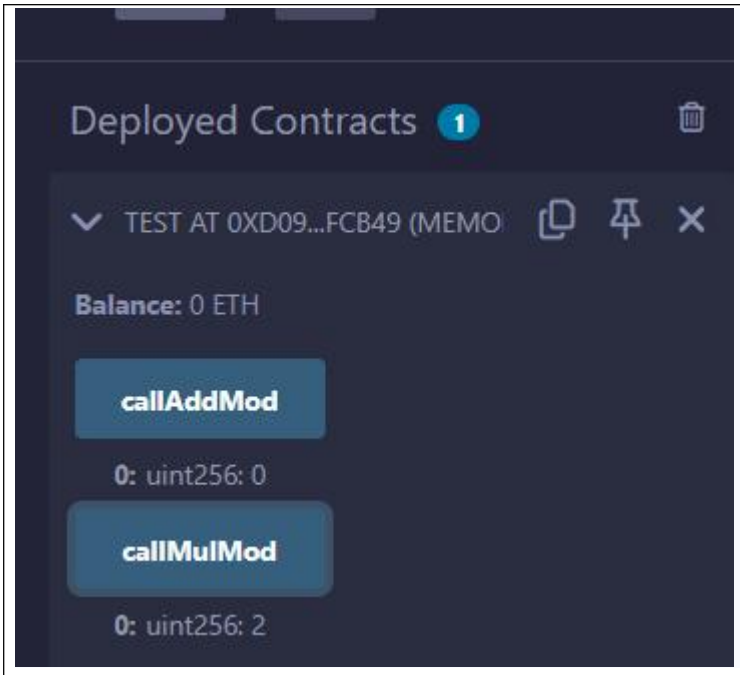




Mathematical Function:

Solidity provides inbuilt mathematical functions as well.

```
pragma solidity ^0.5.0;

contract Test {
function callAddMod() public pure returns(uint){
return addmod(4, 5, 3);
}
function callMulMod() public pure returns(uint){
return mulmod(4, 5, 3);
}
}
```



Cryptographic Function:

Solidity provides inbuilt cryptographic functions as well.

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

PRACTICAL 6B

Aim:- WRITE A SOLIDITY PROGRAM FOR CONTRACT, INHERITANCE, CONSTRUCTORS, ABSTRACT CONTRACTS, INTERFACES, LIBRARIES, ASSEMBLY, EVENTS, ERROR HANDLING.

A)Contract:

Contract in Solidity is similar to a Class in C++. A Contract have following properties.

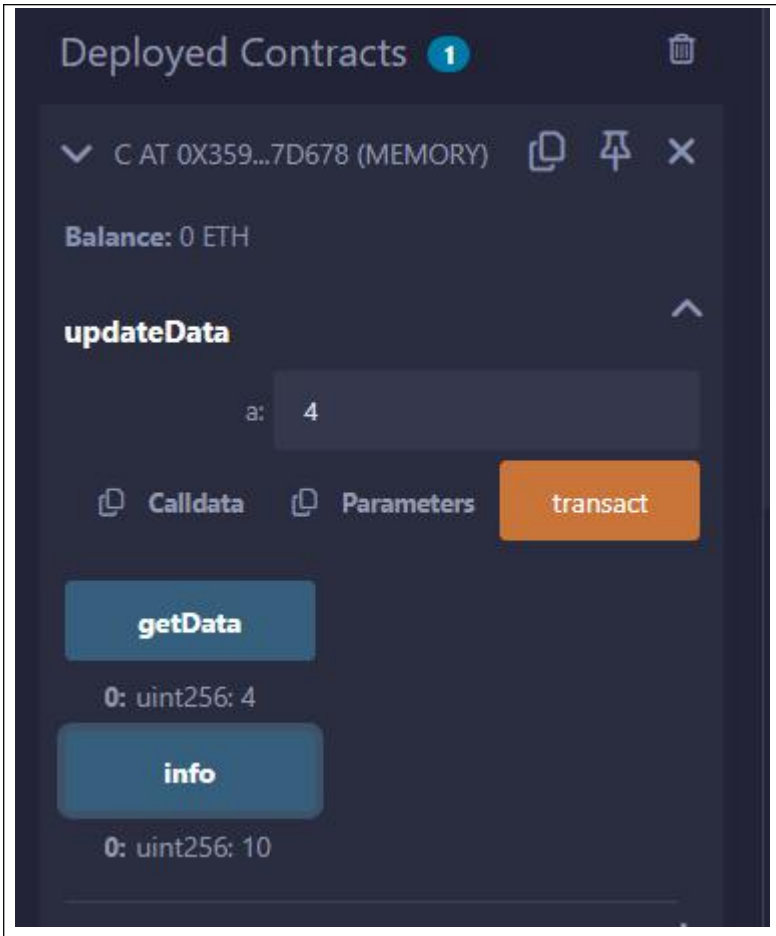
Constructor − A special function declared with constructor keyword which will be executed once per contract and is invoked when a contract is created.

State Variables − Variables per Contract to store the state of the contract.

Functions − Functions per Contract which can modify the state variables to alter the state of a contract.

// Calling function from external contract

```
pragma solidity ^0.5.0;
contract C {
//private state variable
uint private data;

//public state variable
uint public info;

//constructor
constructor() public {
info = 10;
}
//private function
function increment(uint a) private pure returns(uint) { return a + 1; }
//public function
function updateData(uint a) public { data = a; }
function getData() public view returns(uint) { return data; }
function compute(uint a, uint b) internal pure returns (uint) { return a + b; }
}
//Derived Contract
contract E is C {
uint private result;
C private c;

constructor() public {
c = new C();
}
function getComputedResult() public {
result = compute(3, 5);
}
function getResult() public view returns(uint) { return result; }
function getData() public view returns(uint) { return c.info(); }
}
```



B)Inheritance:

Inheritance is a way to extend functionality of a contract. Solidity supports both single as well as multiple inheritance.

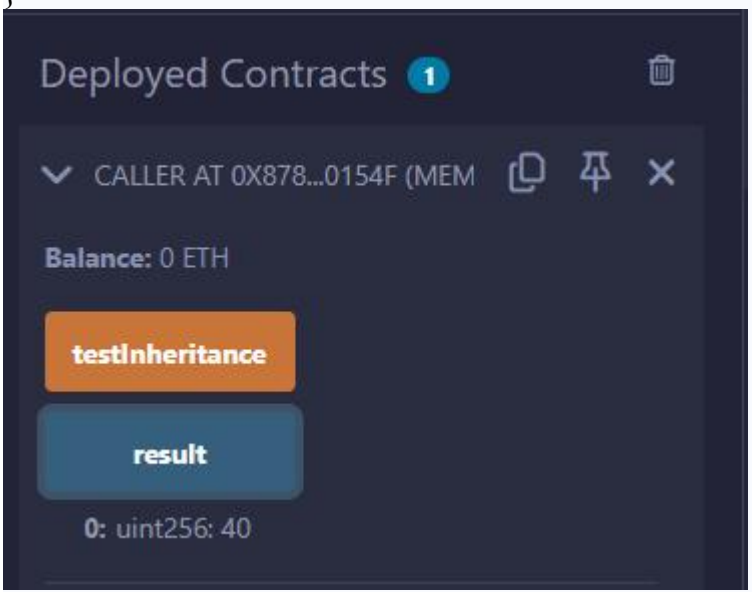
// Solidity program to

// demonstrate

// Single Inheritance

```
pragma solidity >=0.4.22 <0.6.0;

// Defining contract
contract parent{
// Declaring internal
// state variable
uint internal sum;

// Defining external function
// to set value of internal
// state variable sum
function setValue() external {
uint a = 20;
uint b = 20;
sum = a + b;
}
}
// Defining child contract
contract child is parent{

// Defining external function
```

```
// to return value of
// internal state variable sum
function getValue() external view returns(uint) {
return sum;
}
}
// Defining calling contract
contract caller {

// Creating child contract object
child cc = new child();

// Defining function to call
// setValue and getValue functions
function testInheritance() public {
cc.setValue();
}
function result() public view returns(uint ){
return cc.getValue();
}
}
```



C)Constructors:

Constructor is a special function declared using constructor keyword. It is an optional function and is used to initialize state variables of a contract. Following are the key characteristics of a constructor.

A contract can have only one constructor.

A constructor code is executed once when a contract is created and it is used to initialize contract state.

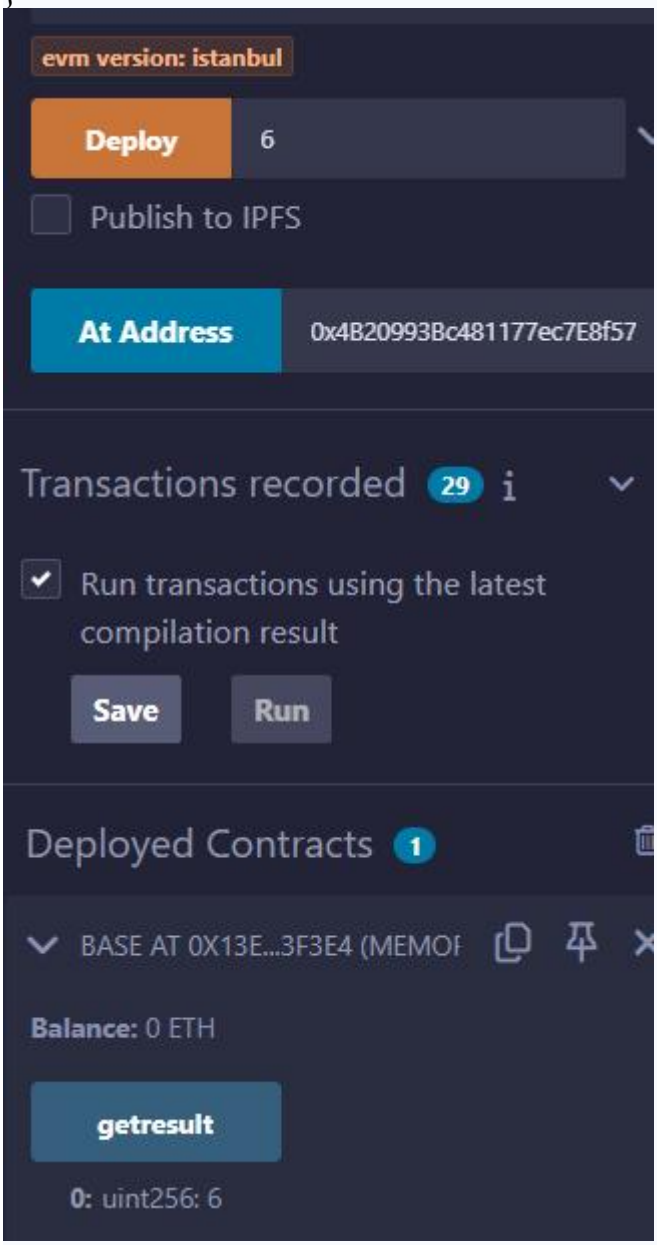
A constructor can be either public or internal.

An internal constructor marks the contract as abstract.

In case, no constructor is defined, a default constructor is present in the contract.

```
pragma solidity ^0.5.0;
contract Base {
uint data;
constructor(uint _data) public {
data = _data;
}
function getresult()public view returns(uint){
```

```
return data;
}
}
contract Derived is Base (5) {
constructor() public {}
}
```



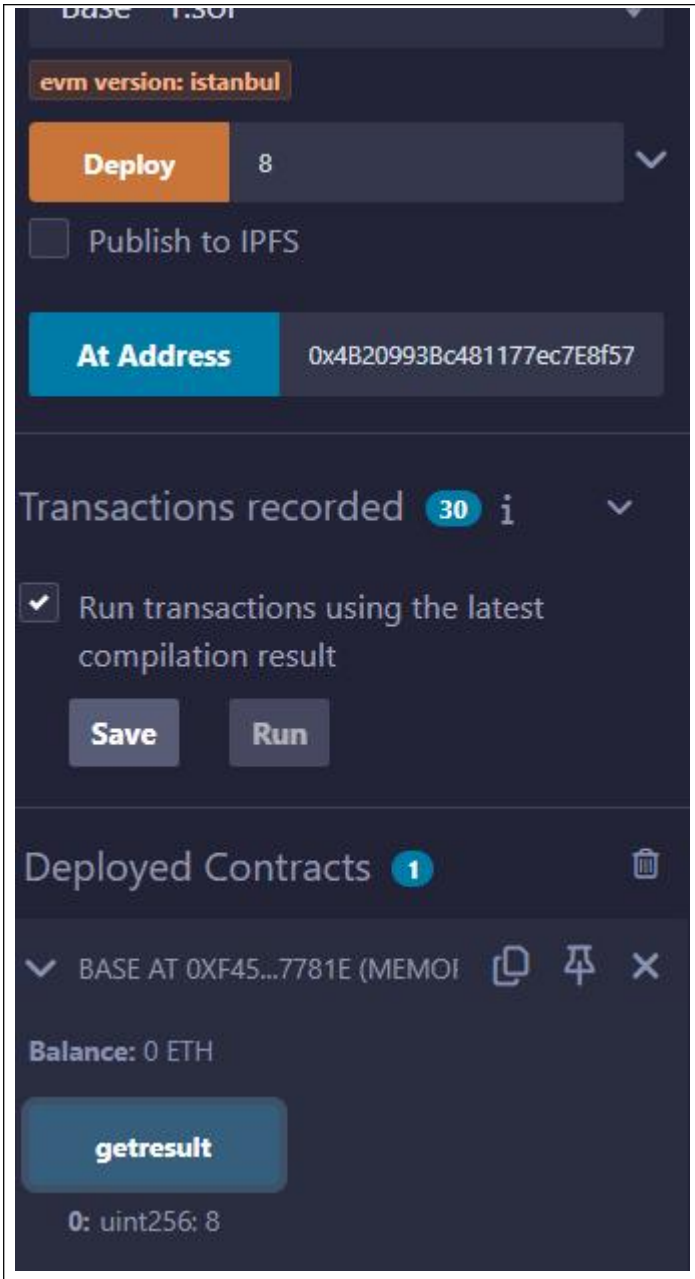
// Indirect Initialization of Base Constructor

```
pragma solidity ^0.5.0;

contract Base {
uint data;
constructor(uint _data) public {
data = _data;
}
function getresult()public view returns(uint){
return data;
}
}
contract Derived is Base {
constructor(uint _info) Base(_info * _info) public {}
}
```

D)Abstract Contracts:

Abstract Contract is one which contains at least one function without any implementation. Such a contract is used as a base contract. Generally an abstract contract contains both implemented as well as abstract functions. Derived contract will implement the abstract function and use the existing functions as and when required.

```
pragma solidity ^0.5.0;

contract Calculator {
function getResult() public view returns(uint);
}
contract Test is Calculator {
function getResult() public view returns(uint) {
uint a = 4;
uint b = 2;
uint result = a + b;
return result;
}
}
```



E)Interfaces:

Interfaces are similar to abstract contracts and are created using interface keyword. Following are the key characteristics of an interface.

Interface can not have any function with implementation.

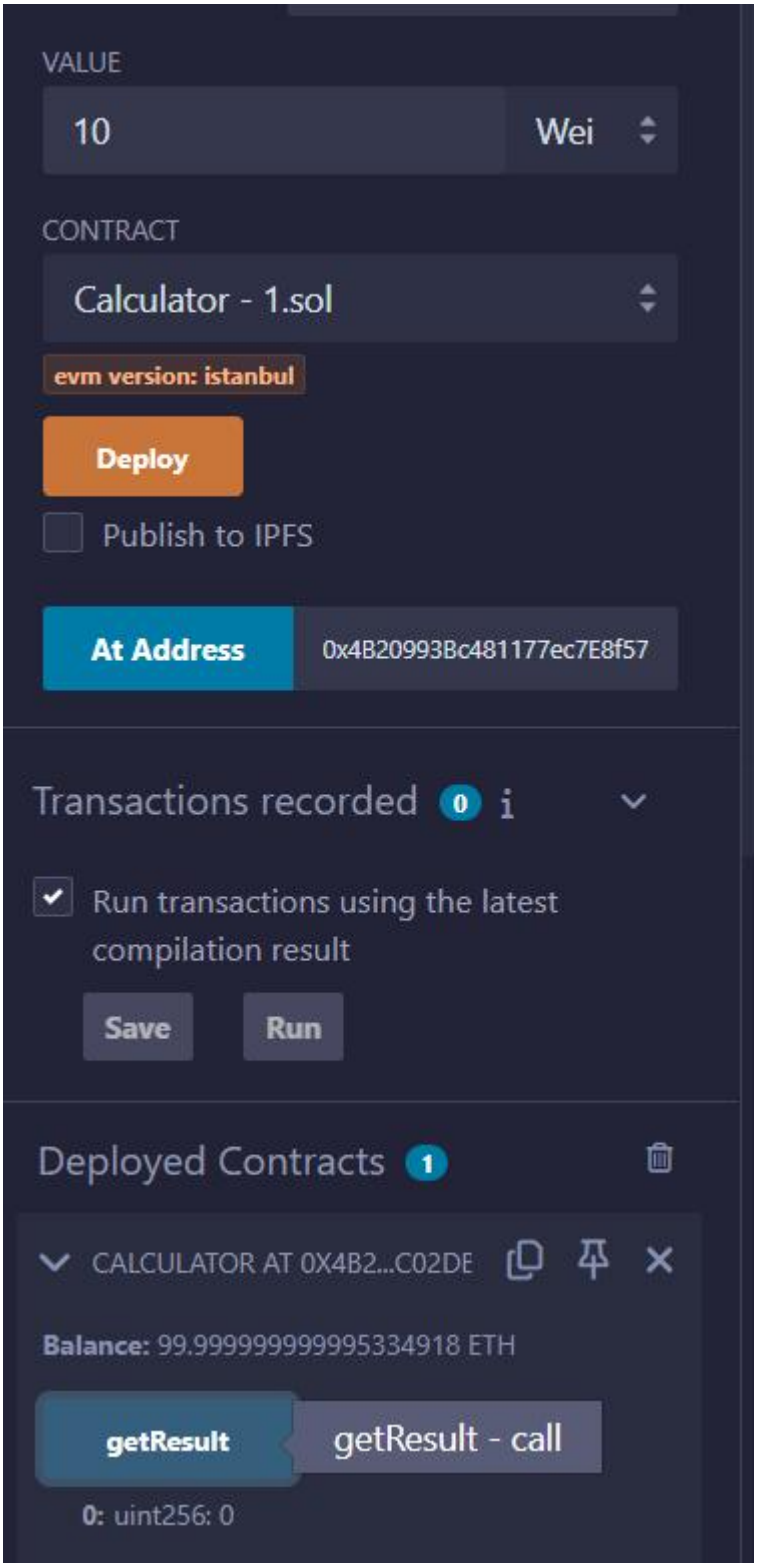
Functions of an interface can be only of type external.

Interface can not have constructor.

Interface can not have state variables.

```
pragma solidity ^0.5.0;

interface Calculator {
function getResult() external view returns(uint);
}
contract Test is Calculator {
constructor() public {}
function getResult() external view returns(uint){
uint a = 5;
uint b = 2;
```

```
uint result = a + b;
return result;
}
}
```



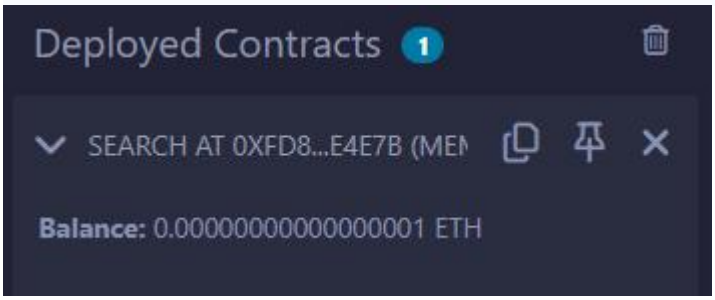
PRACTICAL 6c

Aim:-Libraries,Assembly, Events, Error handling.

F)Libraries:

Libraries are similar to Contracts but are mainly intended for reuse. A Library contains functions which other contracts can call. Solidity have certain restrictions on use of a Library.

```
pragma solidity ^0.5.0;

library Search {
function indexOf(uint[] storage self, uint value) public view returns (uint) {
for (uint i = 0; i < self.length; i++)
if (self[i] == value) return i;
return uint(-1);}
}
contract Test {
uint[] data;
uint value;
uint index;
constructor() public {
data.push(6);
data.push(7);
data.push(8);
data.push(9);
data.push(10);
}
function isValuePresent() external {
value = 9;
//search if value is present in the array using Library function
index = Search.indexOf(data, value);
}
function getresult() public view returns(uint){
return index;
}}
```

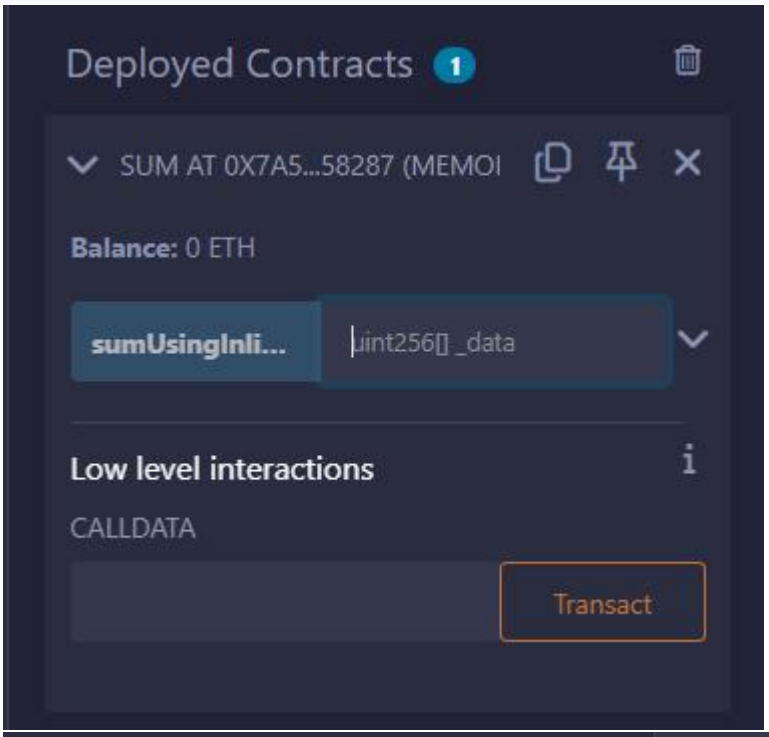


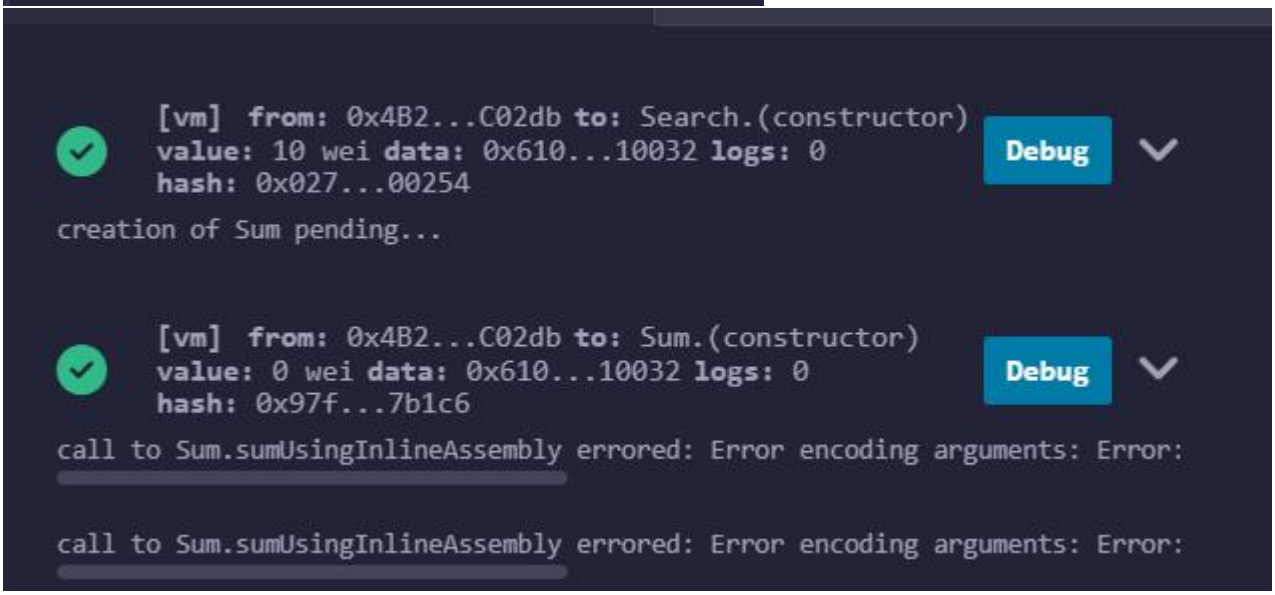
G)Assembly:

Solidity provides an option to use assembly language to write inline assembly within Solidity source code. We can also write a standalone assembly code which then be converted to bytecode. Standalone Assembly is an intermediate language for a Solidity compiler and it converts the Solidity code into a Standalone Assembly and then to byte code. We can used the same language used in Inline Assembly to write code in a Standalone assembly.

```
pragma solidity ^0.5.0;

library Sum {
function sumUsingInlineAssembly(uint[] memory _data) public pure returns (uint o_sum) {
for (uint i = 0; i < _data.length; ++i) {
```

```
assembly {
o_sum := add(o_sum, mload(add(add(_data, 0x20), mul(i, 0x20))))
}}
}
}
contract Test {
uint[] data;
constructor() public {
data.push(1);
data.push(2);
data.push(3);
data.push(4);
data.push(5);
}
function sum() external view returns(uint){
return Sum.sumUsingInlineAssembly(data);
}
}
```





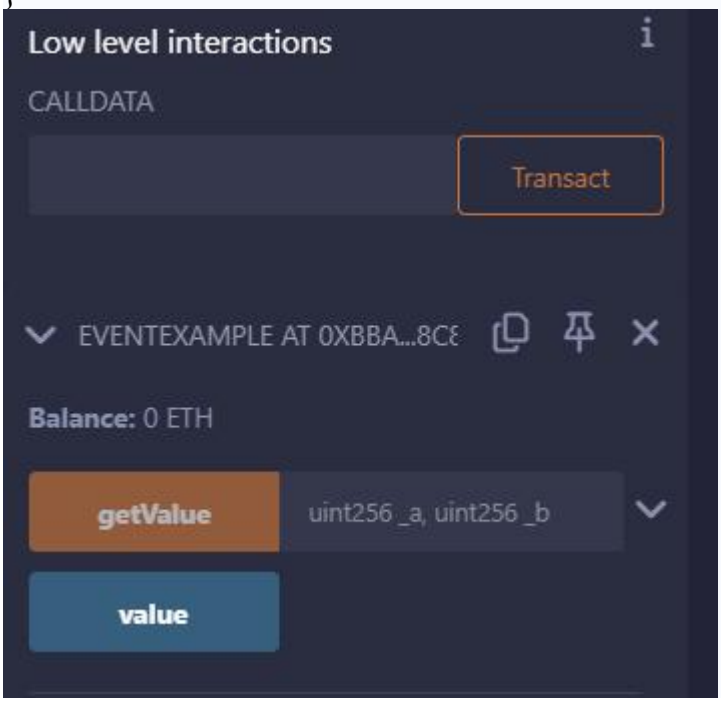
H)Events:

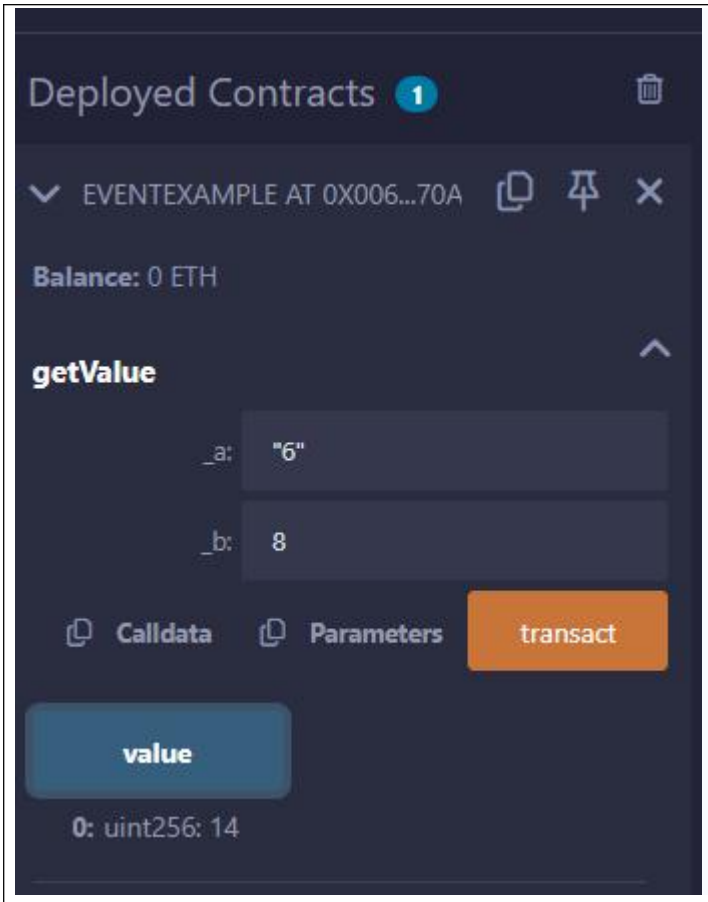
Event is an inheritable member of a contract. An event is emitted, it stores the arguments passed in transaction logs. These logs are stored on blockchain and are accessible using address of the contract till the contract is present on the blockchain. An event generated is not accessible from within contracts, not even the one which have created and emitted them.

// Solidity program to demonstrate

// creating an event

```
pragma solidity ^0.4.21;

// Creating a contract
contract eventExample {

// Declaring state variables
uint256 public value = 0;

// Declaring an event
event Increment(address owner);

// Defining a function for logging event
function getValue(uint _a, uint _b) public {
emit Increment(msg.sender);
value = _a + _b;
}
}
```





I)Error Handling:

Solidity provides various functions for error handling. Generally when an error occurs, the state is reverted back to its original state. Other checks are to prevent unauthorized code access.

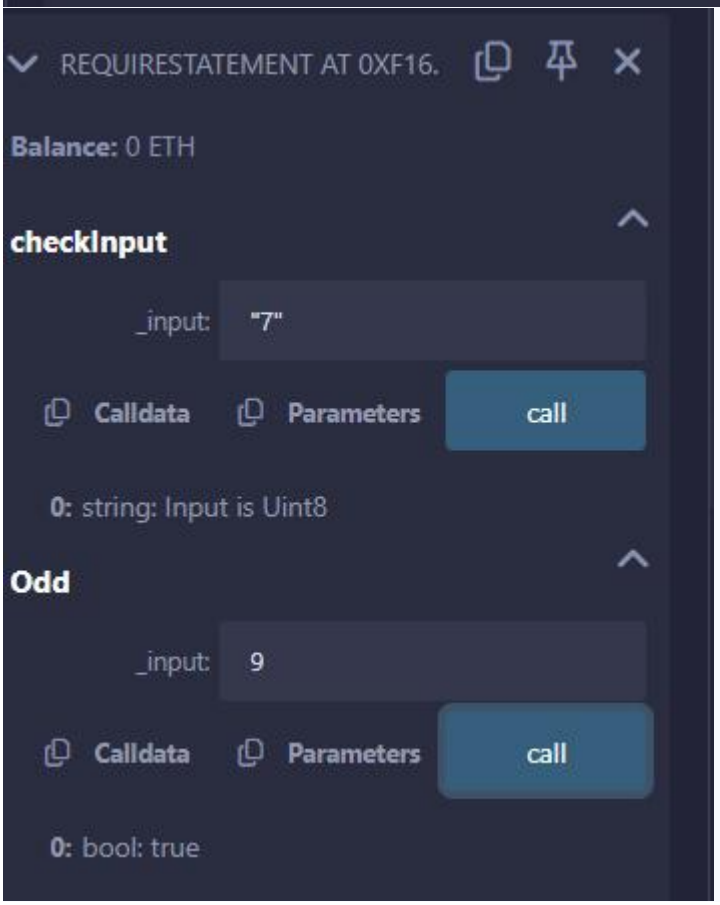
Solidity program to demonstrate require statement.

// Solidity program to

// demonstrate require

// statement

```
pragma solidity ^0.5.0;
// Creating a contract
contract requireStatement {
// Defining function to
// check input
function checkInput(uint8 _input) public view returns(string memory){
require(_input >= 0, "invalid uint");
require(_input <= 255, "invalid uint8");

return "Input is Uint8";
}
// Defining function to
// use require statement
function Odd(uint _input) public view returns(bool){
require(_input % 2 != 0);
return true;
}
}
```



Solidity program to demonstrate assert statement.

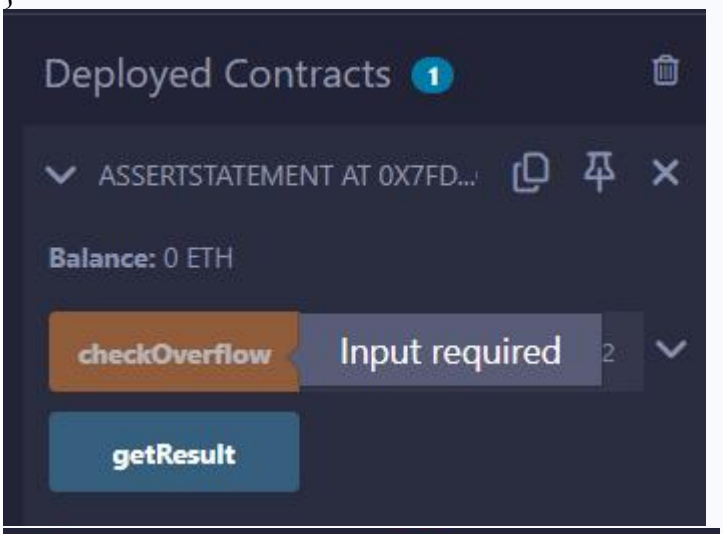
// Solidity program to

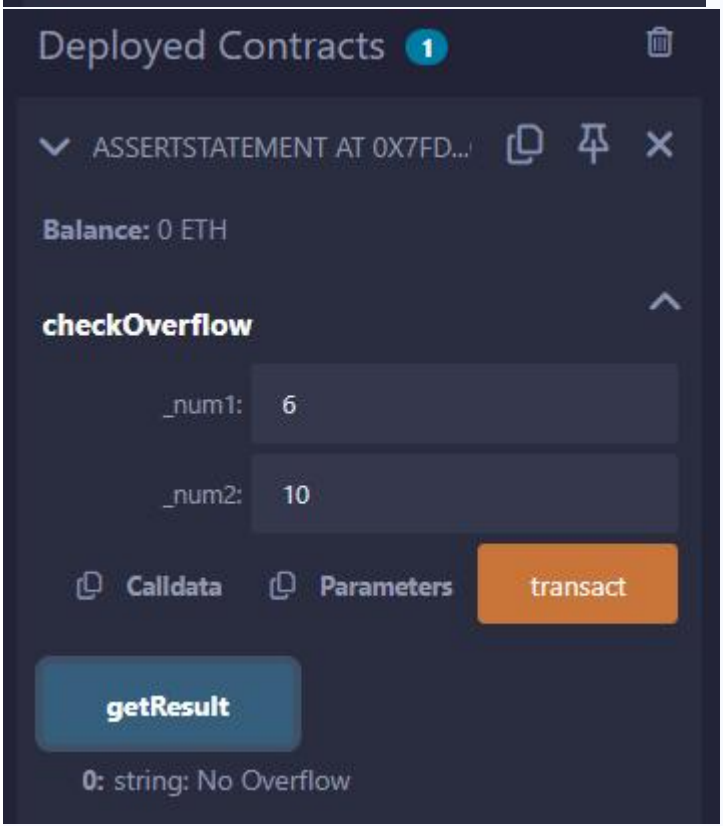
// demonstrate assert

// statement

```
pragma solidity ^0.5.0;

// Creating a contract
contract assertStatement {
// Defining a state variable
bool result;
// Defining a function
// to check condition
function checkOverflow(uint8 _num1, uint8 _num2) public {
uint8 sum = _num1 + _num2;
assert(sum<=255);
```

```
result = true;
}
// Defining a function to
// print result of assert
// statement
function getResult() public view returns(string memory){
if(result == true){
return "No Overflow";
}
else{
return "Overflow exist";
}
}
}
```





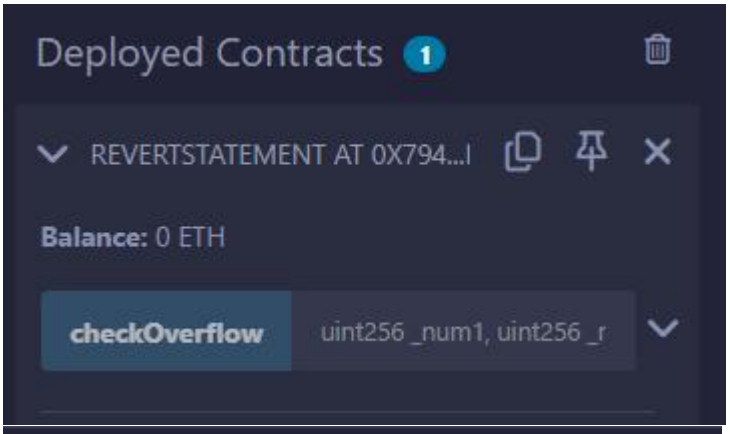
Solidity program to demonstrate revert statement.

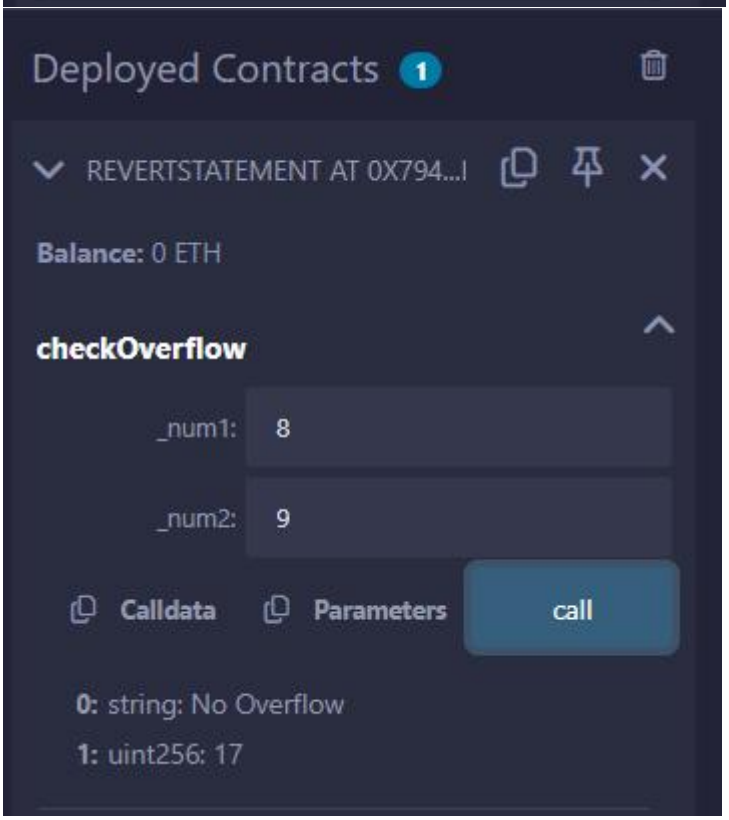
// Solidity program to

// demonstrate revert

```
pragma solidity ^0.5.0;
// Creating a contract
contract revertStatement {
// Defining a function
// to check condition
function checkOverflow(uint _num1, uint _num2) public view returns(
string memory, uint) {
uint sum = _num1 + _num2;
if(sum < 0 || sum > 255){
revert(" Overflow Exist");
}
else{
return ("No Overflow", sum);
}
}
}
```

https://www.tutorialspoint.com/solidity/index.htm





PRACTICAL 7

Aim:-

7 Deploying a contracts on an external blockchain by using Ganache and/or MyEtherwalllet, Metamask

PRACTICAL 1

Aim:-
Code:-
Output:-

8. Deploy a local private blockchain over a network with Ethereum or Rust (VM)

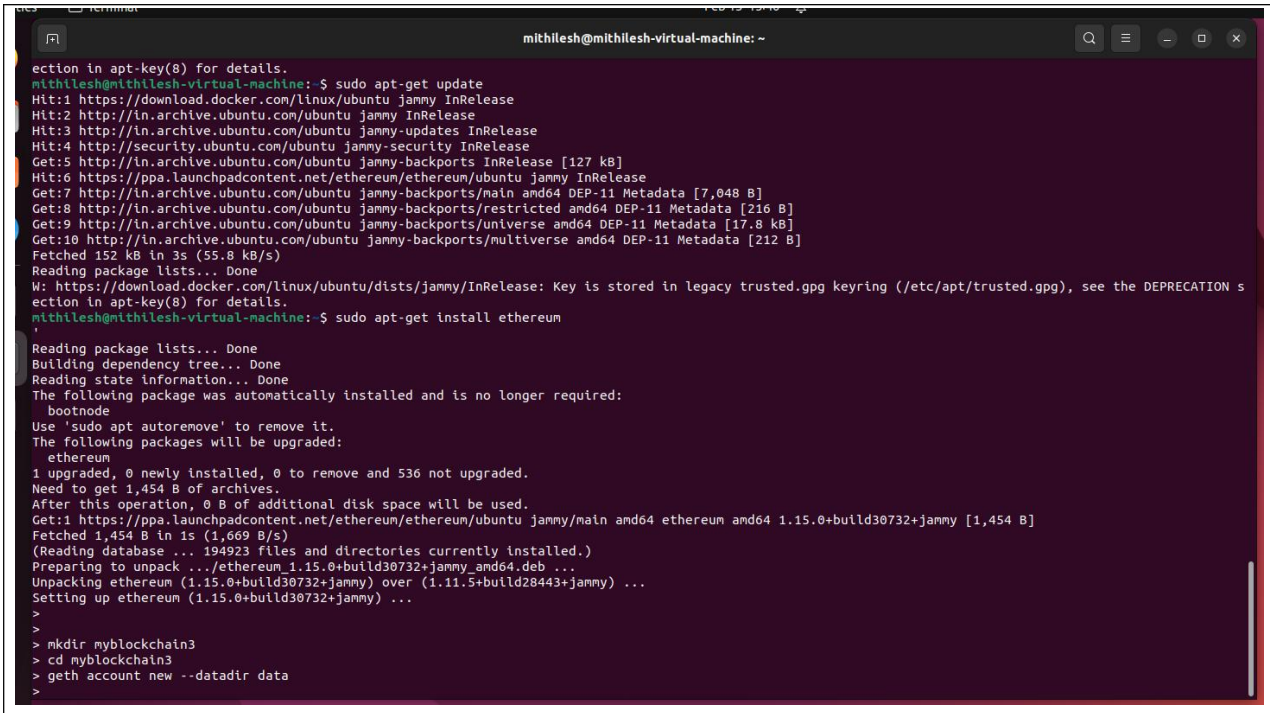
PRACTICAL 8

Aim:-Create your own blockchain and demonstrate its use.
Install on Ubuntu via PPAs
The easiest way to install go-ethereum on Ubuntu-based distributions is with the built-in launchpad PPAs (Personal Package Archives). We provide a single PPA repository that contains both our stable and development releases for Ubuntu versions trusty, xenial, zesty and artful.

linux:
To enable our launchpad repository run:
Step 1: open new terminal
Step 2: on terminal type this command
sudo add-apt-repository -y ppa:ethereum/ethereum
#if above command gives error then run
#sudo apt-get install --reinstall ca-certificates

Step 3: install the stable version of go-ethereum:
sudo apt-get update
sudo apt-get install ethereum

Step 4: create new directory for storing blockchain data
mkdir myblockchain2
cd myblockchain2
geth account new --datadir data

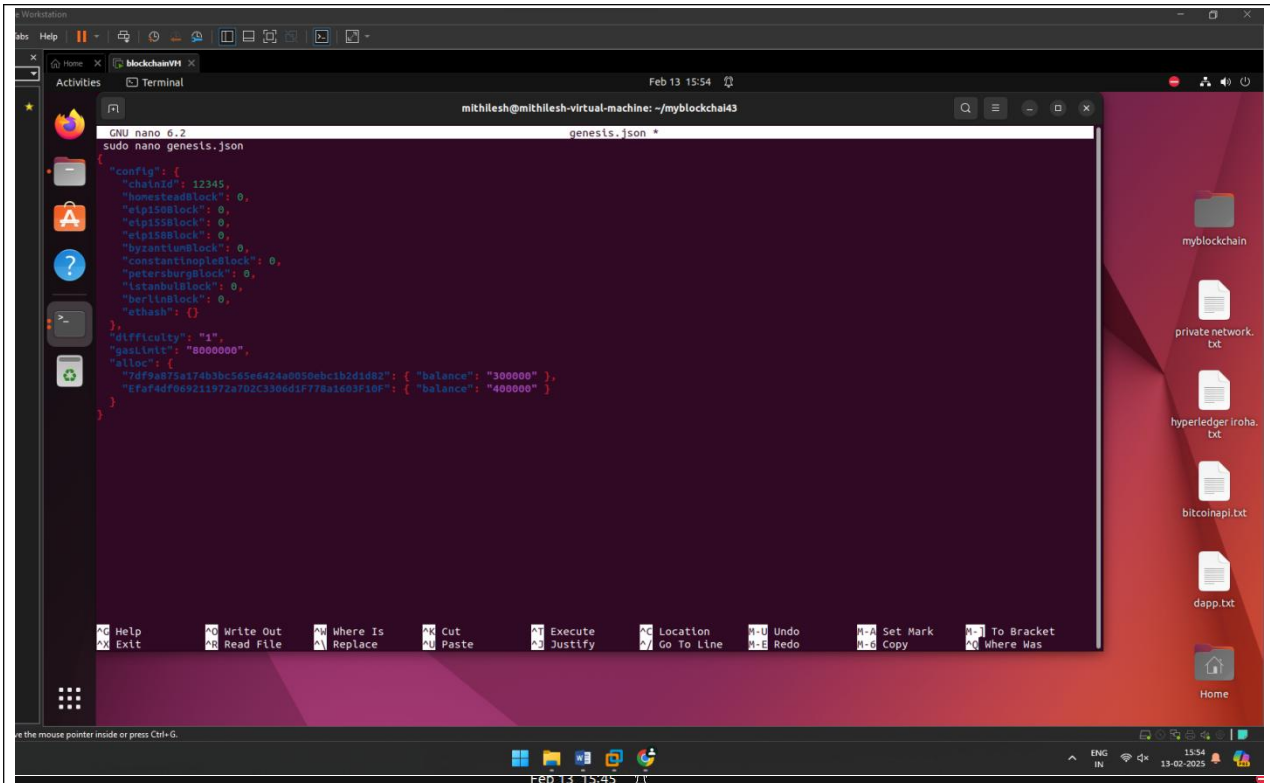


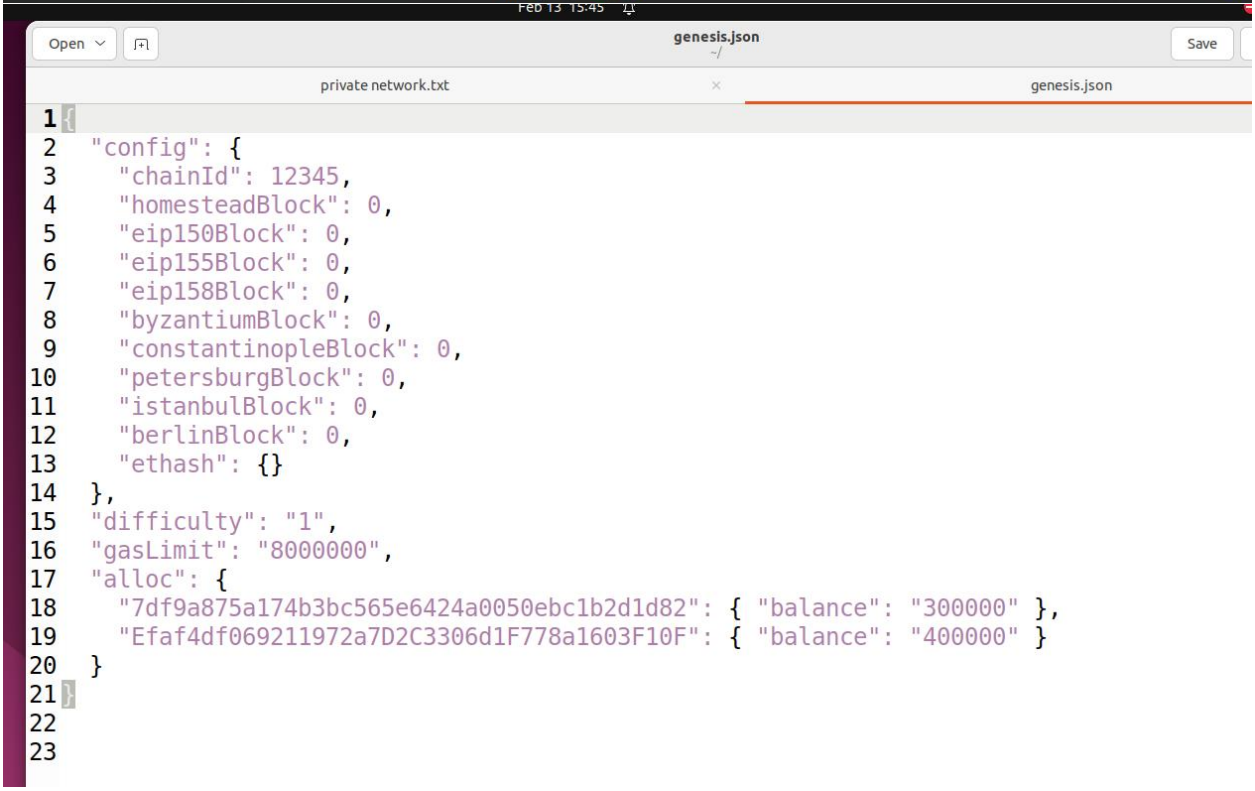
Step 5:Create genesis.json file

sudo nano genesis.json

```
{
 "config": {
  "chainId": 12345,
  "homesteadBlock": 0,
  "eip150Block": 0,
  "eip155Block": 0,
  "eip158Block": 0,
  "byzantiumBlock": 0,
  "constantinopleBlock": 0,
  "petersburgBlock": 0,
  "istanbulBlock": 0,
  "berlinBlock": 0,
  "ethash": {}
 },
 "difficulty": "1",
 "gasLimit": "8000000",
 "alloc": {
  "7df9a875a174b3bc565e6424a0050ebc1b2d1d82": { "balance": "300000" },
  "Efaf4df069211972a7D2C3306d1F778a1603F10F": { "balance": "400000" }
 }
}
```

save the file -> ctrl +o to write -> {enter} save -> ctrl +x exit

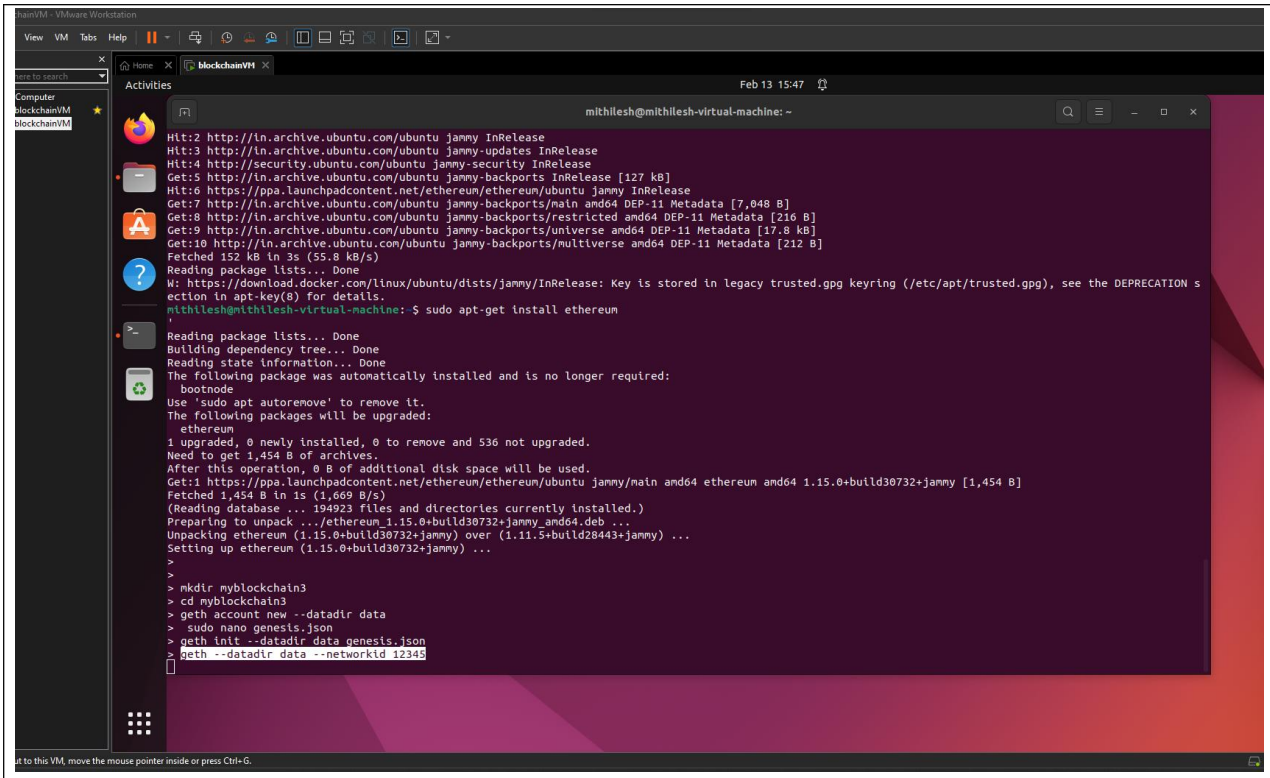




Step 6: initialize the block
geth init --datadir data genesis.json
Step 7: create network
geth --datadir data --networkid 12345

[do not close this terminal]

//



Step 8: open new tab/terminal 2:
sudo geth attach data/geth.ipc
eth.getBalance(eth.accounts[0])
miner.setEtherbase(eth.accounts[0])
miner.start()
admin.addPeer(admin.nodeInfo.enode)
eth.getBalance(eth.accounts[0])

Step 10: Wait for 10-20 minutes and check balance
eth.getBalance(eth.accounts[0])

if ether balance is 0 wait for 10-20minutes for mining process to get complete and run eth.getBalance(eth.accounts[0]) again.

After balance is updated you can check current block height
eth.blockNumber

PRACTICAL 9

Aim:-
Code:-
Output:-

9Implement the mining module of Bitcoin client . The mining module, or miner, should produce blocks that solve proof-of-work puzzle

9. Compile and test smart contracts on a testing framework using the Ethereum Virtual Machine (EVM).

PRACTICAL 10

Aim:-
Code:-
Output:-

11Demonstrate the running of the blockchain node

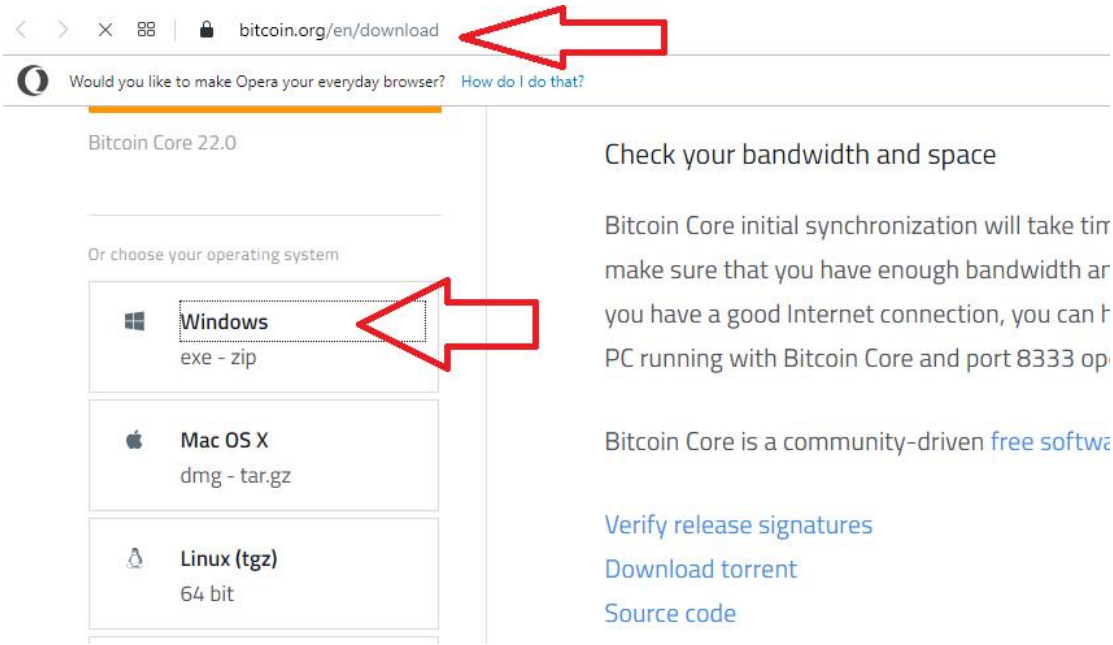
PRACTICAL 11

Aim:-
Code:-
Output:-

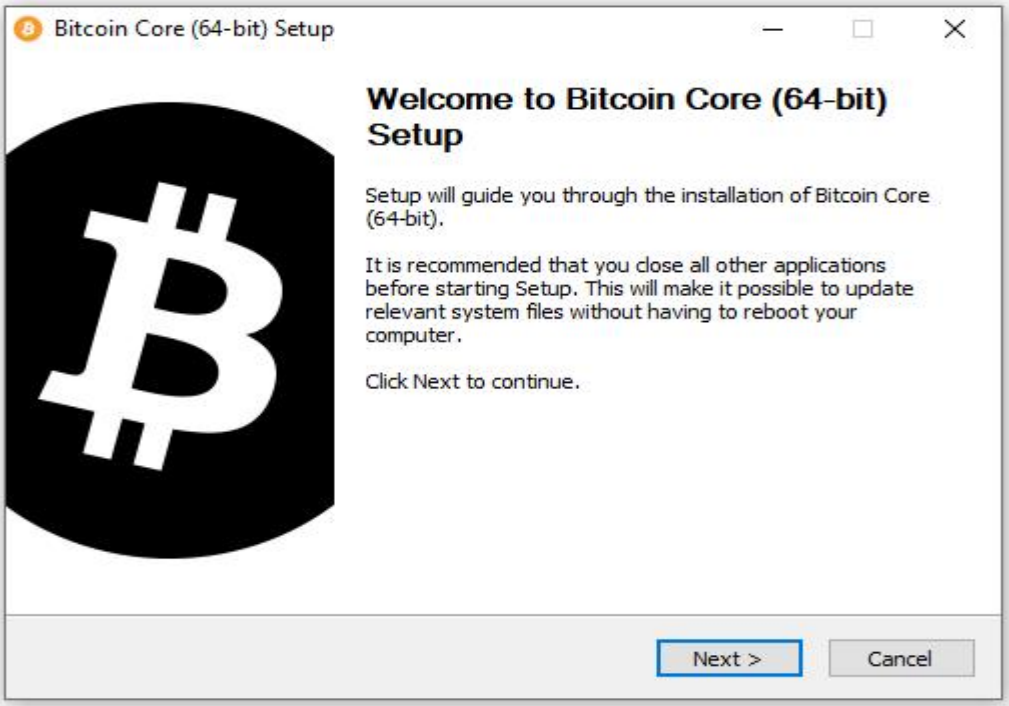
PRACTICAL 12

Aim:-**Demonstrate the use of Bitcoin Core API.**
Step 1: Visit: https://bitcoin.org/en/download

Step 2: Download windows setup [use and try with Linux version as well]

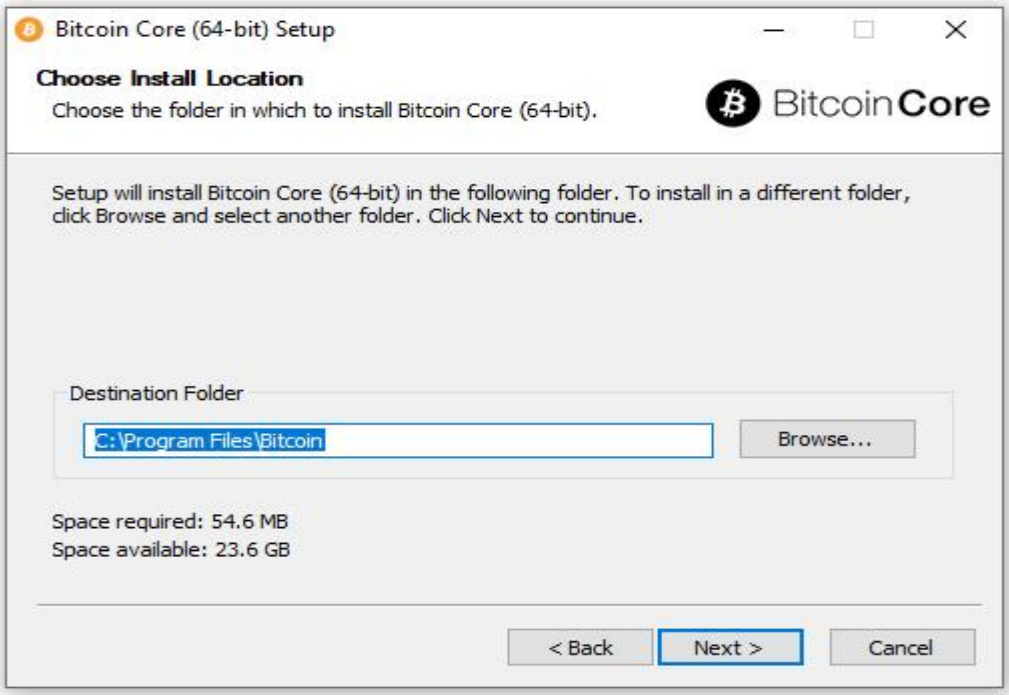


Step 3: Run the setup file-> click

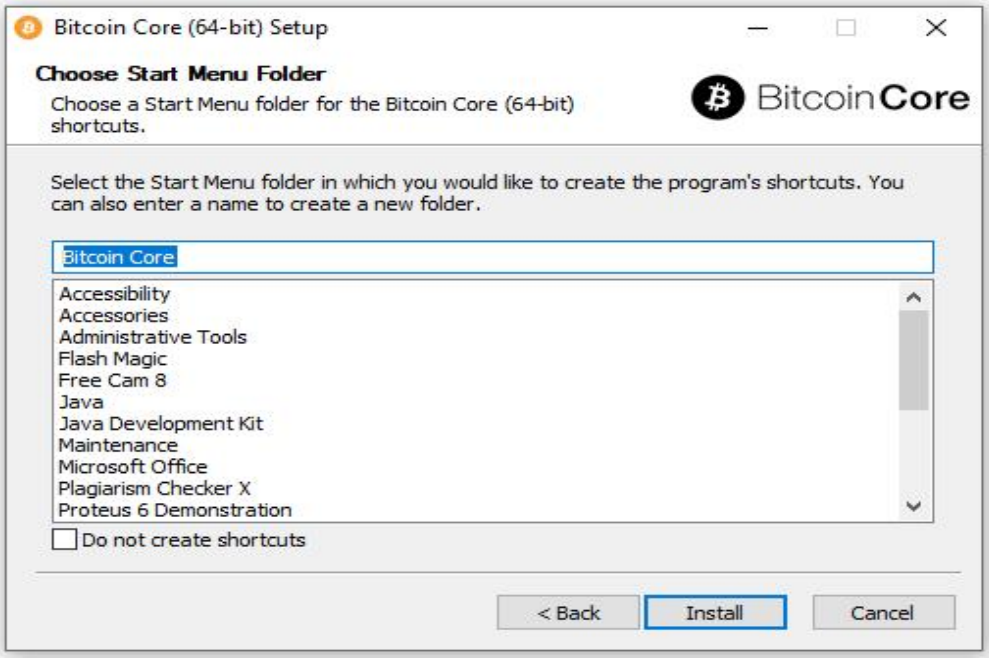


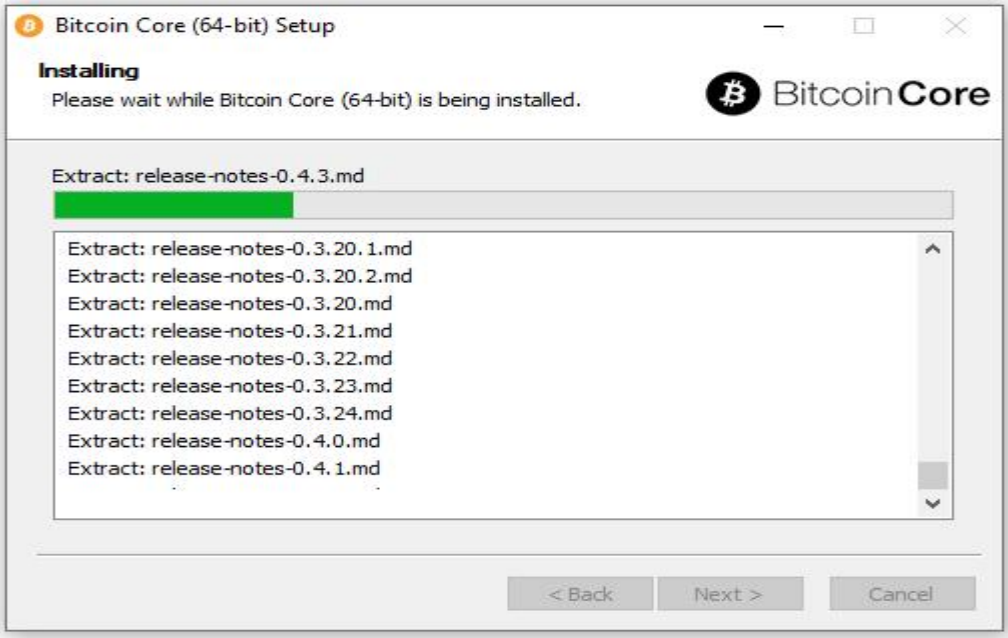
next

Step 4: Click Next

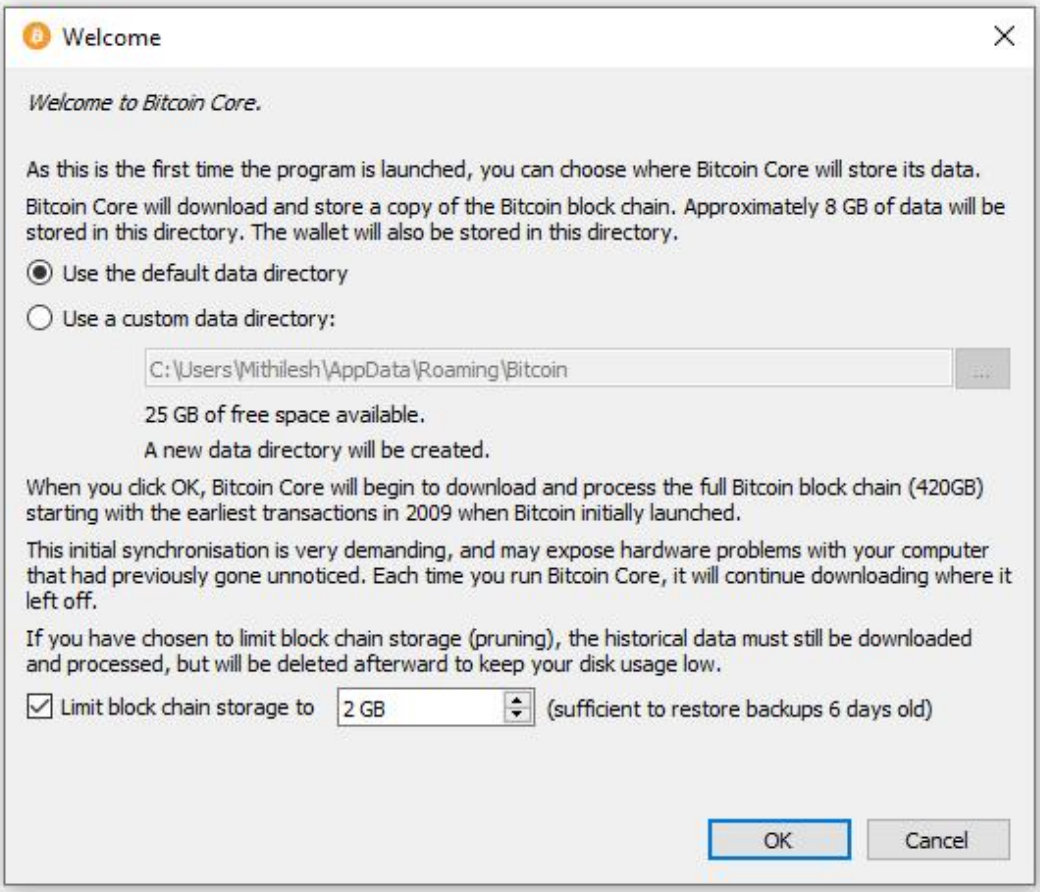


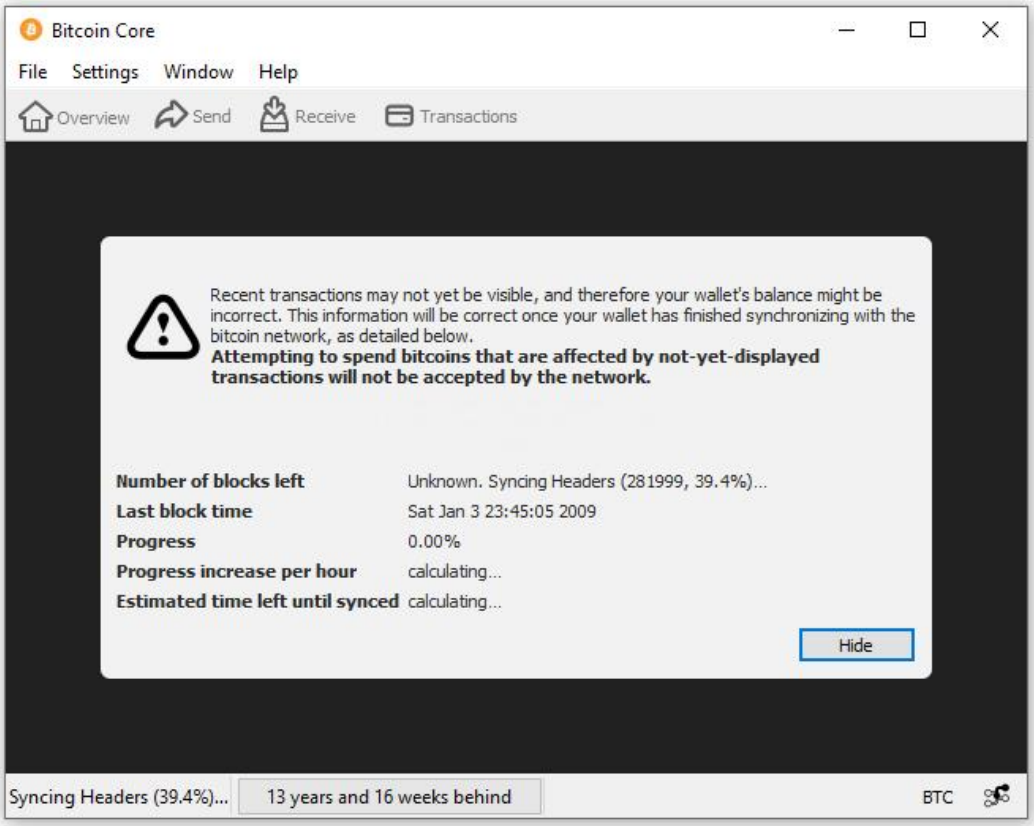
Step 5: Finally click on Install



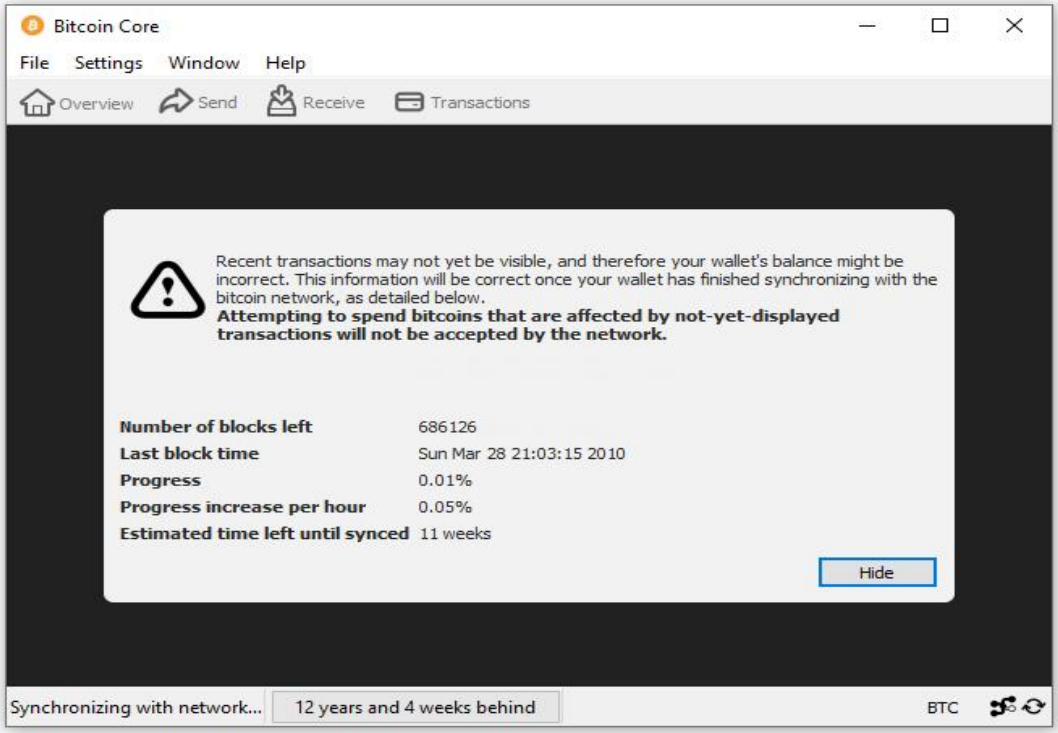


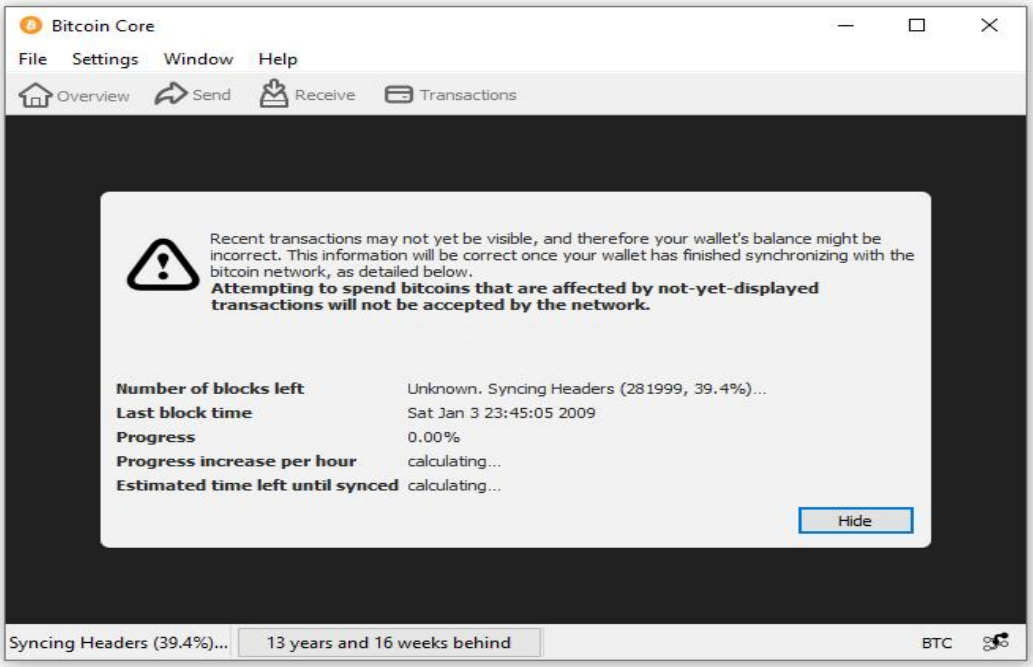
Launch Bitcoin Core-> Click OK.



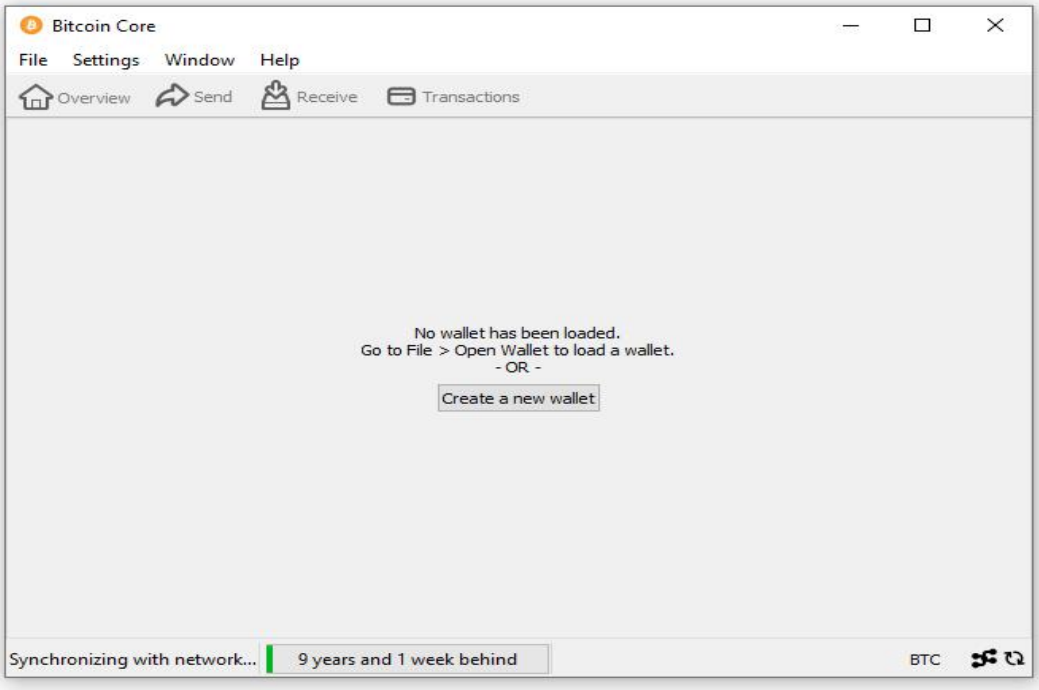


Click on Hide button [Synchronization take place in background]

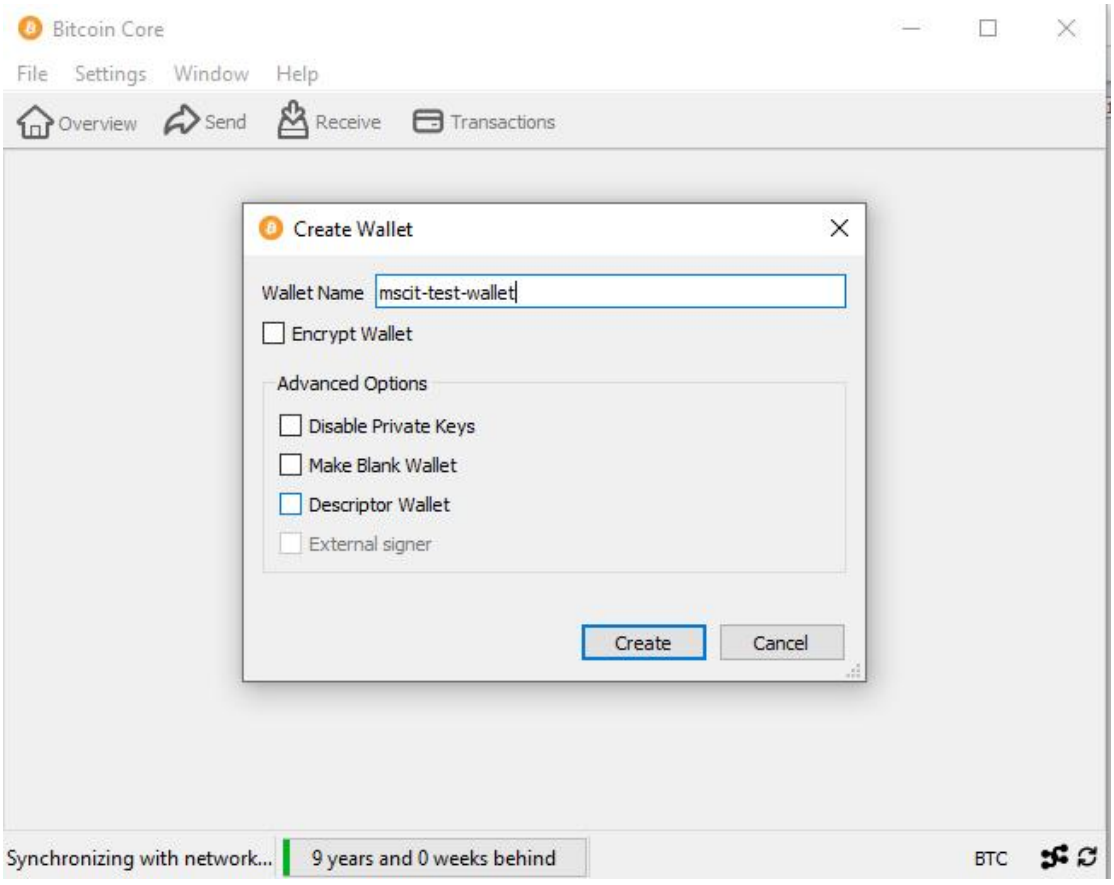




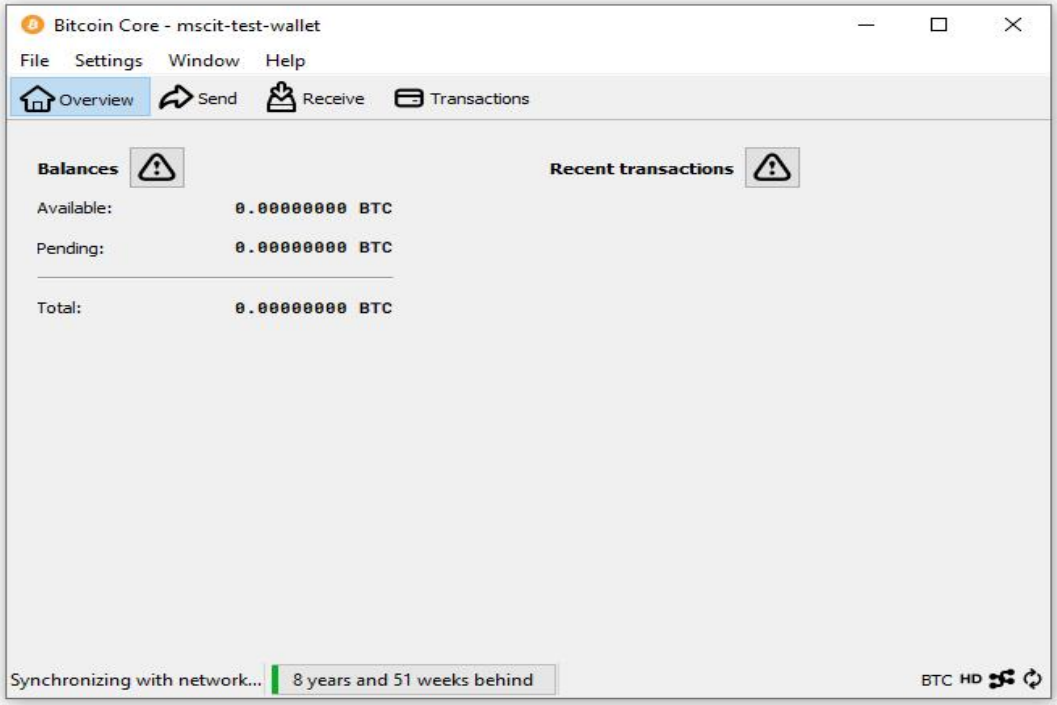
You can create a wallet -> Create a new wallet

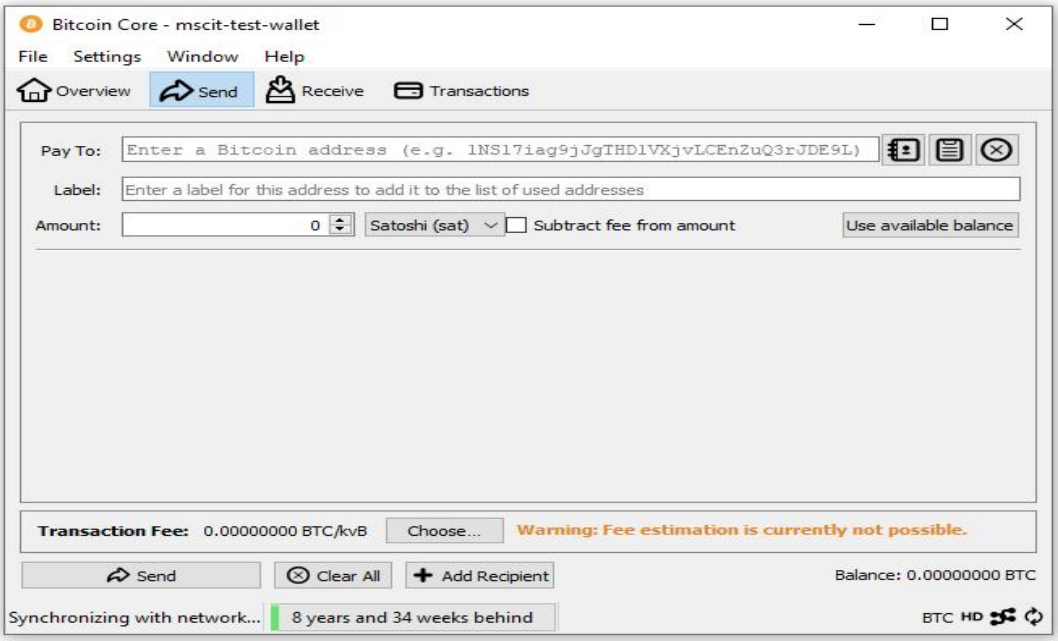


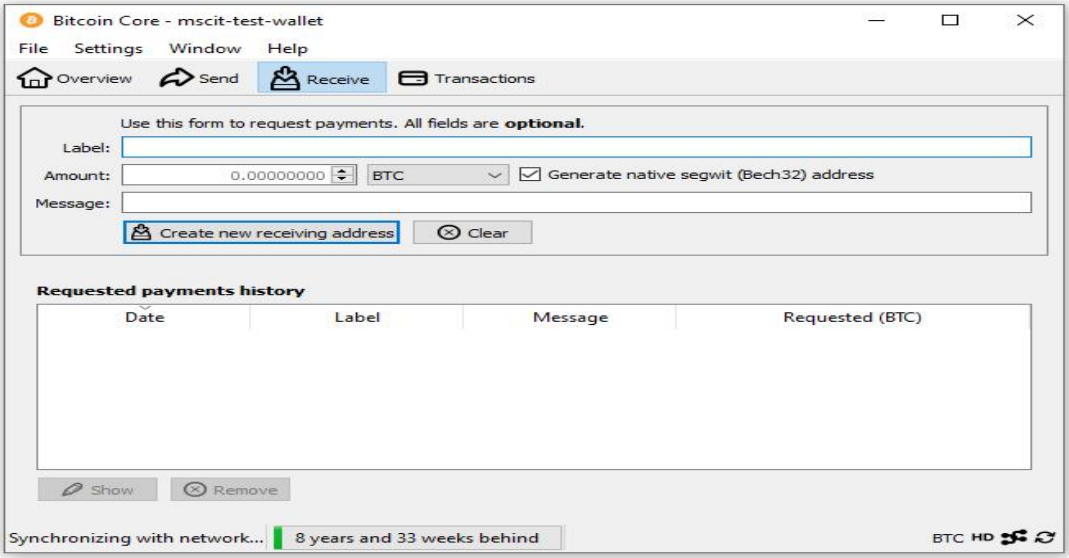
Enter Wallet name

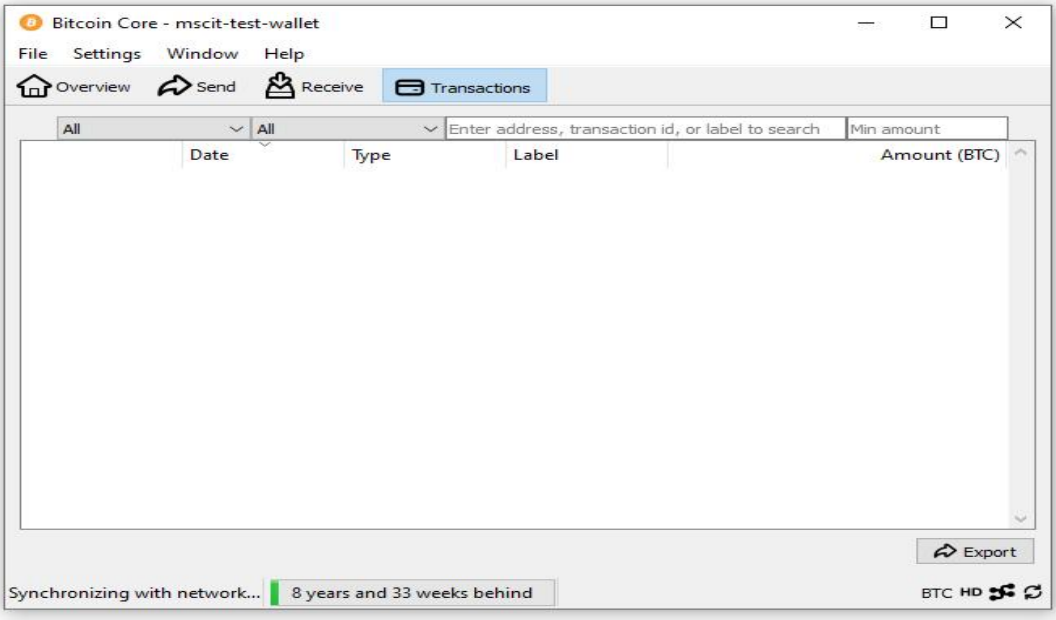


Finally Account is setup


Bitcoin Core - mscit-test-wallet
File
Settings
Window
Help
Overview
Send
Receive
Transactions
Balances
Recent transactions
Available:
0.00000000 BTC
Pending:
0.00000000 BTC
Total:
0.00000000 BTC
Synchronizing with network...
8 years and 51 weeks behind
BTC
HD


Bitcoin Core - mscit-test-wallet
File
Settings
Window
Help
Overview
Send
Receive
Transactions
Pay To:
Enter a Bitcoin address (e.g. 1NS17iag9jJgTHD1VXjvLCEnZuQ3rJDE9L)
Label:
Enter a label for this address to add it to the list of used addresses
Amount:
0
Satoshi (sat)
Subtract fee from amount
Use available balance
Transaction Fee:
0.00000000 BTC/kvB
Choose...
Warning: Fee estimation is currently not possible.
Send
Clear All
Add Recipient
Balance: 0.00000000 BTC
Synchronizing with network...
8 years and 34 weeks behind
BTC
HD





Open Python IDLE and create new Script.

##########################

```
from bitcoinlib.wallets import Wallet
w = Wallet.create('Wallet1')
key1 = w.get_key()
print('Wallet Address:',key1.address)
w.scan()
print(w.info())
```

```
Wallet Address: bc1qppnqpg9quay7qf5hzh2ekx2cu0g8tmky666u5n
=== WALLET ===
 ID                          1
 Name                        Wallet1
 Owner
 Scheme                      bip32
 Multisig                    False
 Witness type                segwit
 Main network                bitcoin
 Latest update               2025-02-17 04:17:53.572643+00:00

= Wallet Master Key =
 ID                          1
 Private                     True
 Depth                       0

- NETWORK: bitcoin -
- - Keys
     6 m/84'/0'/0'/0/0         bc1qppnqpg9quay7qf5hzh2ekx2cu0g8tmky666u5n    address index 0          0.00000000 ฿
     7 m/84'/0'/0'/0/1         bc1qy762wx0jqp9y6psekwhg6yn2h0z2ry7gavymra    address index 1          0.00000000 ฿
     9 m/84'/0'/0'/0/2         bc1q0lvwvm045pl0q9hdvucwk8h5dmvu88ggrdeuyj    address index 2          0.00000000 ฿
    10 m/84'/0'/0'/0/3         bc1ql33pg2mjzempsyzxme740xzzsln7fc7hhry4fl    address index 3          0.00000000 ฿
    11 m/84'/0'/0'/0/4         bc1q4p3fhm8r7sjwzhhq9lxrp5wg0thqpn2k3xj5fe    address index 4          0.00000000 ฿
    13 m/84'/0'/0'/1/0         bc1qkcuslwjmd2uhgs9auwua0xgydt4sc6jrjlqesx    address index 0          0.00000000 ฿
    15 m/84'/0'/0'/1/1         bc1qs29svsg7he7l2nqpfk6y4k0x5hk5zev9r0nvv5    address index 1          0.00000000 ฿
    16 m/84'/0'/0'/1/2         bc1qeh7dpxunzd55rc35tgg7qr2c4krahn70pvg2wv    address index 2          0.00000000 ฿
    17 m/84'/0'/0'/1/3         bc1q3cwgrkmrcclprvdk9caqpfjhnyzcznhhw0pag3    address index 3          0.00000000 ฿
    18 m/84'/0'/0'/1/4         bc1q3wd8tnqsy7a5u7rugz07de54cwxd8vdswn2fxr    address index 4          0.00000000 ฿

- - Transactions Account 0 (0)

= Balance Totals (includes unconfirmed) =


None
```

######################

Open CMD and install bitcoinlib package

pip install bitcoinlib

PRACTICAL 13

Aim:-Create your own blockchain and demonstrate its use.

Code:-

```
# following imports are required by PKI
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

A = Client()
B =Client()
C =Client()
t0 = Transaction (
  "Genesis",
  A.identity,
  500.0
)

t1 = Transaction (
  A,
  B.identity,
  40.0
)
t2 = Transaction (
  A,
  C.identity,
  70.0
)
t3 = Transaction (
  B,
  C.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)


################################################################################
```

save the file -> ctrl +O to write -> {enter} save -> ctrl +x exit

Run this file

Output:

```
Number of blocks in the chain: 3
block # 0
sender: Genesis
-----
recipient: 30819f300d06092a864886f70d010101050003818d00308189028181 00b1558beccb109cbf1c223ebf6e791ea734e20c03dc8bce926f3b28c9e2a2383cf4ae87b6431211299b11
7d6e143373a6682a64e0c7de6955fbd9dc8806103d4c6a738d92d7511112e944c9d6eb51730b76e2b0b6a48069b6bd90c52549832429cbf8ba7ade362d4f3b04a5d568f54d30d6e3cb57ceaf1
8e7e7b2fb2df2d8d2530203010001
-----
value: 500.0
-----
time: 2025-02-17 10:34:26.627963
-----
--------------
======================================


======================================
block # 1
sender: 30819f300d06092a864886f70d010101050003818d00308189028181 00b1558beccb109cbf1c223ebf6e791ea734e20c03dc8bce926f3b28c9e2a2383cf4ae87b6431211299b117d6
e143373a6682a64e0c7de6955fbd9dc8806103d4c6a738d92d7511112e944c9d6eb51730b76e2b0b6a48069b6bd90c52549832429cbf8ba7ade362d4f3b04a5d568f54d30d6e3cb57ceaf18e7
e7b2fb2df2d8d2530203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d00308189028181 00bbcf0e76057d12120ce04cdd708cc9dca881bca54eaf520584faadeaa97a6b90de616b04036b465e8f1b
bb2b79c3236f59d83c818830280b108c1a114026eb93cff404b9b78d757147b83cc259e099ae9166a0afde96a06f6fdcd1c8d99aab21ded8bf89ea7eab67efdbd52370daf555eee8ceb7e56a2
ba7c37f441c6ac00aa70203010001
-----
value: 40.0
-----
time: 2025-02-17 10:34:26.627963
-----
--------------
sender: 30819f300d06092a864886f70d010101050003818d00308189028181 00b1558beccb109cbf1c223ebf6e791ea734e20c03dc8bce926f3b28c9e2a2383cf4ae87b6431211299b117d6
e143373a6682a64e0c7de6955fbd9dc8806103d4c6a738d92d7511112e944c9d6eb51730b76e2b0b6a48069b6bd90c52549832429cbf8ba7ade362d4f3b04a5d568f54d30d6e3cb57ceaf18e7
e7b2fb2df2d8d2530203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d00308189028181 00bcfacd1614746e02a0ab42490f4932f55c8a339b2e2c4661c74950d9a1a22c3217e56525342d57bc51f2
c5a97fe5bd0601c8e6cc30b5752c3c6c927add09703245b10ee469ae427d4929d947e51d4e1b8921cf2fb83f6d20aef8be123f9a30ac2d8eb602a8fdcf2dfb8150db16a31196af78de1b09028
022fd48e63f476916d50203010001
-----
value: 70.0
-----
time: 2025-02-17 10:34:26.627963
-----
--------------
======================================


======================================
block # 2
sender: 30819f300d06092a864886f70d010101050003818d00308189028181 00bbcf0e76057d12120ce04cdd708cc9dca881bca54eaf520584faadeaa97a6b90de616b04036b465e8f1bbb2
b79c3236f59d83c818830280b108c1a114026eb93cff404b9b78d757147b83cc259e099ae9166a0afde96a06f6fdcd1c8d99aab21ded8bf89ea7eab67efdbd52370daf555eee8ceb7e56a2ba7
c37f441c6ac00aa70203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d00308189028181 00bcfacd1614746e02a0ab42490f4932f55c8a339b2e2c4661c74950d9a1a22c3217e56525342d57bc51f2
c5a97fe5bd0601c8e6cc30b5752c3c6c927add09703245b10ee469ae427d4929d947e51d4e1b8921cf2fb83f6d20aef8be123f9a30ac2d8eb602a8fdcf2dfb8150db16a31196af78de1b09028
022fd48e63f476916d50203010001
-----
value: 700.0
-----
time: 2025-02-17 10:34:26.627963
-----
--------------
======================================
```

EXTRA

PRACTICAL 1

Aim:-

Code:-
Output:-

---

https://tinyurl.com/vaze123
format of practical

Practical No.

Aim: Question

Program:

Output:
Practical 1
aim: types of variable

```
pragma solidity ^0.5.0;
contract Pract1{
int x=15; //state var
int public y=10;//global
function getValue(int z) public{
y=y+z;
}

function show() public view returns (int)
{
return x;
}
}
```

Practical 2
Aim: relational operators

```
pragma solidity ^0.5.0;
contract Pract2{
bool public a=true;
bool  public b=false;
bool public  r1or=a||b;
bool public r2and=a&&b;
bool public r3not=!b;
}
```

Practical 3
aim:for loop

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

Practical 4
Aim:while loop

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

Practical 5
Aim: do while loop

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

Practical 6
Aim: if else

```
pragma solidity ^0.5.0;
contract Pract4{
function test(int x) public view returns(string memory)
{
if(x%2==0)
return "Number is even";
else
return "Number is odd";
}
}
```

Practical 7
Aim: string

```
pragma solidity ^0.5.0;
contract Pract4{
function test(int x) public view returns(string memory)
{
if(x%2==0)
return "Number is even";
else
```

```
return "Number is odd";
}
}
```

Practical 8

Aim:array

```
contract Types {
uint[5] data;
 constructor() public
{
   data = [uint(10), 20, 30, 40, 50];
}
function array_example() public view returns (uint,uint)
{

return (data[0],data[4]);
}
function array_example2() public view returns (uint [5] memory)
{

return data;
}
}
```

Practical 9

Aim: enum

```
pragma solidity ^0.5.0;
contract Types {
enum week_days
{
Monday,Tuesday,Wednesday,Thursday,Friday,Saturday,
Sunday
}
week_days choice;
function set_value() public {
choice = week_days.Thursday;
}
function get_choice() public view returns (week_days) {
return choice;
}
}
```

Practical 10

//Aim: arithmetic operations

```
pragma solidity ^0.5.0;

// Creating a contract

contract SolidityTest {

// Initializing variables

uint16 public a = 20;

uint16 public b = 10;

// Initializing a variable

// with sum
```

```
uint public sum = a + b;
// Initializing a variable
// with the difference
uint public diff = a - b;
// Initializing a variable
// with product
uint public mul = a * b;
// Initializing a variable
// with quotient
uint public div = a / b;
// Initializing a variable
// with modulus
uint public mod = a % b;
// Initializing a variable
// decrement value
uint public dec = --b;
// Initializing a variable
// with increment value
uint public inc = ++a;
}
```

Practical 11

Aim: structure

```
pragma solidity ^0.5.0;
contract test {
struct Book {
string title;
string author;
uint book_id;
}
Book book;

function setBook() public {
book = Book('Learn Java', 'TP', 1);
}
function getBookId() public view returns (uint) {
return book.book_id;
}
function getBookDetail() public view returns (string memory, string memory,uint) {
return (book.title, book.author, book.book_id);
}

}
```

Practical 12

Aim: type of function(view, pure)

```
pragma solidity ^0.5.0;
contract Test {
int public x=10; //global
int y=90;//state
function f1() public returns(int){
   //read and update is allowed
   x=100;
return x;
}
function f2() public view returns(int){
  //  x=100; //erro beacuse x is global/state
  //we can access but we cannot update state or global variable int view function
return x;
}
function f3() public pure returns(int){
   //we cannot access or update state or global variable in pure function
   int z=80;
return z;
}

}
```

Practical 13

Aim:function overloading

```
pragma solidity ^0.5.0;
contract Test {
function getSum(uint a, uint b) public pure returns(uint){
return a + b;
}
function getSum(uint a, uint b, uint c ) public pure returns(uint){
return a + b + c;
}
}
```

Practical 14

Aim: mathematical function

```
pragma solidity ^0.5.0;

contract Test {
function callAddMod() public pure returns(uint){
return addmod(4, 5, 3);
}
function callMulMod() public pure returns(uint){
return mulmod(4, 5, 3);
}
}
```

Practical 15

Aim: cryptographic function

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