CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution, Affiliated to Osmania University, Approved by AICTE, Accredited by NAAC with A++ Grade and Programs Accredited by NBA)
Chaitanya Bharathi Post, Gandipet, Kokapet (Vill.), Hyderabad, Ranga Reddy - 500 075, Telangana
www.cbit.ac.in

DEPARTMENT OF Information Technology

INDEX

Exp.No	Name of the Experiment	Date of Experiment	Date of Submission	Page No.	Record Marks Grade	Signature of Faculty
1.	Understanding Blockchain Foundations: Elements of Distributed Computing, Cryptography, Digital Signature.	26.07.23	27.08.23	01	A+	
2.	Etherium platform & Etherium virtual machine	02.08.23	27.08.23	04	A+	
3.	Solidity program structure, compilation and deployment environment.	09.08.23	27.08.23	08	A+	
4.	Compiling and deploying simple smart contract to store and get "Hello World".	16.08.23	27.08.23	11	A+	
5.	Smart contract to create a function setter and getter to set & get a value.	23.08.23	27.08.23	13	A+	

INDEX

Exp.No	Name of the Experiment	Date of Experiment	Date of Submission	Page No.	Record Marks Grade	Signature of Faculty
6.	Develop Solidity contracts to illustrate inheritance & polymorphism.	30.08.23	04.10.23	14	A+	
7.	Familiarize with the working of Remix Etherium Tool.	06.09.23	04.10.23	18	A	
8.	Smart Contract to print the array of integers and its length.	13.09.23	04.10.23	19	A+	
9.	Setup a Simple Etherium wallet and use it to send and receive Ethers.	04.10.23	01.11.23	20	A+	
10.	Hyperledger Fabric Demo	11.10.23	01.11.23	22	A+	

Laboratory Record

of Fundamentals of Blockchain Technology

Roll No.: 160120737157

Experiment No.: 01

Sheet No.: 01

Date: 26/07/2023

Aim:- To understand Blockchain Foundations:

Elements of Distributed Computing

Elements of Cryptography

Digital Signature

Blockchain Definition:-

Blockchain is peer-to-peer, distributed ledger, cryptographically secure, append and it is immutable via consensus.

Elements of Distributed Computing:-

Distributed computing is a field of computer science that deals with designing and implementing systems that involve multiple interconnections of computers working together to achieve a common goal. There are several key components/elements in distributed computing:-

(i) Confidentiality:-

Confidentiality is the assurance that the information is accessible only to the authorized. Confidentiality breaches may occur due to improper data handling or a hacking attempt.

(ii) Integrity:-

Integrity is the trustworthiness of data or resources in the prevention of improper and unauthorized changes.

(iii) Authentication:-

It refers to the characteristic of a communication, document or any data that ensures the quality of being genuine.

(iv) Non-Repudiation:-

It is the guarantee that the sender of a message cannot later deny having sent the message and that the recipient cannot deny having received the message.

Elements of Cryptography:-

Cryptography is the process of hiding or coding information so that only the person a message was intended for can read it.

There are three types of cryptography. They are:-

1. Symmetric Encryption
2. Asymmetric Encryption
3. Hashing

① Symmetric Encryption:-

Symmetric encryption uses the same key for encryption as it does for decryption.

Ex:- Data Encryption Standard (DES), Advanced Encryption Standard (AES)

② Asymmetric Encryption:-

Asymmetric Encryption uses different encryption keys for encryption and decryption. These keys are known as public and private keys. Ex:- Rivest Shamir Adleman (RSA), Diffie-Hellman

③ Hashing:-

Hashing refers to the process of taking an input and applying a mathematical algorithm to generate a fixed size output, known as the hash value or hash code.

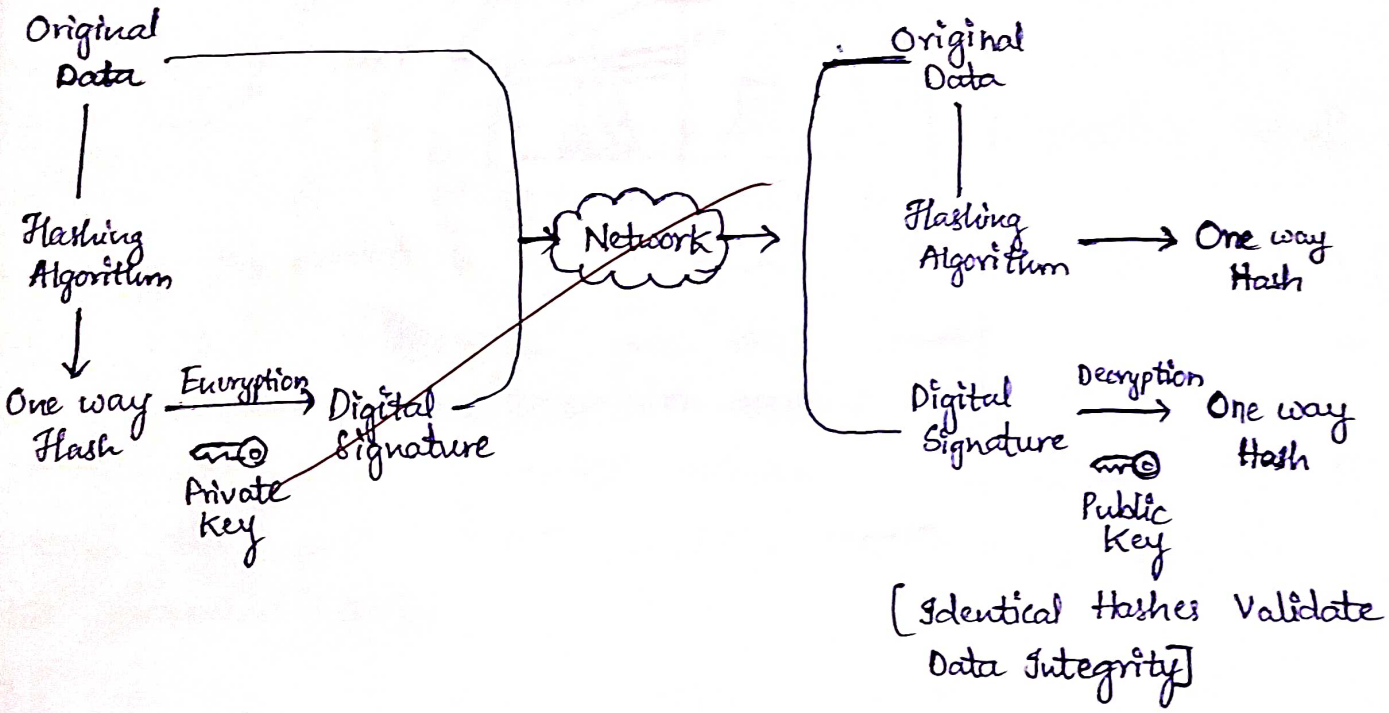
Laboratory Record
of FBCT

Roll No.: 16012073157
Experiment No.: 01
Sheet No.: 03
Date: 26/9/2023

Digital Signature:-

A digital signature is a cryptographic technique that verifies the authenticity and integrity of digital messages. It involves creating a unique "signature" using a private key to validate the sender's identity and ensure that the content hasn't been altered.

Others can verify the signature using the sender's public key, providing non-repudation and tamper detection.



Laboratory Record
of FBCT

Roll No.: 160120737154
Experiment No.: 02
Sheet No.: 04
Date: 09/08/2023

Aim: Getting familiar with the Etherium platform and Etherium virtual machine.

Description:

→ Etherium is a Blockchain network that introduced a built-in Turing-Complete programming language that can be used for creating various decentralized applications (also called Dapps). The Etherium network is fueled by its own cryptocurrency called "ether".

→ The etherium network is currently famous for allowing the implementation of Smart contracts. Smart contracts can be thought as "cryptographic bank lockers" which contain certain values.

→ These cryptographic lockers can only be unlocked when certain conditions are met. Unlike bitcoin, Etherium is a network that can be applied to various other sectors. Etherium is often called Blockchain 2.0, since it proved the potential of blockchain technology beyond the financial sector.

→ The consensus mechanism used in Etherium is Proof-of-Stakes (POS), which is more energy efficient when compared to that used in the Bitcoin network, that is, Proof of Work (POW). POS depends on the amount of stake a node holds.

Features of Etherium:-

(i) Smart Contracts:-

→ Etherium allows the creation and deployment of smart contracts which are created mainly using a programming language called solidity. Solidity is OOP language that is compatitively easy to learn.

(ii) Etherium Virtual Machine (EVM):-

→ Designed to operate as a runtime environment for compiling and deploying Etherium-based smart contracts.

(iii) Ether:-

→ It is the cryptocurrency and the only acceptable form of payment for transaction fees on the etherium network.

(iv) Decentralized application (Daaps):-

→ Daap has its backend code running on a decentralized peer-to-peer network. It can have a frontend and user interface written in any language to make calls and query data from its backend. They operate on Etherium and perform the same function irrespective of the environment in which they get executed.

(v) Decentralized autonomous organizations (DAOs):-

→ It is a decentralized organization that works in a democratic and decentralized fashion. DAO relies on smart contracts for decision-making or decentralized voting systems within the organization.

Laboratory Record
of EECT

Roll No.: 160120733152
Experiment No.: 02
Sheet No.: 06
Date: 09/02/2023

Type of Ethereum Accounts:

(i) Externally owned Account (EOA):-

→ Controlled by private keys.

→ Each EOA has a public-private key pair, the users can send message by creating and signing transactions.

(ii) Contract Account:-

→ Controlled by contract codes, which gets activated everytime a transaction form an EOA or a message from another contract is received by it.

→ When the contract code activates, it allows to read/write the message to the local storage, send messages and create contracts.

How does Etherium Work?

Etherium implements an execution environment called Etherium Virtual Machine (EVM). When a transaction triggers a smart contract all the nodes of the network will execute every instruction.

All the nodes will run the EVM as part of the block verification, where the nodes will go through the transactions listed in the block and runs the code as triggered by the transaction in the EVM.

Every transaction must include:-

→ Gas limit

→ Transaction Fee that the sender is willing to pay for the transaction.

Laboratory Record
of FBCT

Roll No.: 160120737157
Experiment No.: 02
Sheet No.: 07
Date: 09/08/2023

Real world applications of Etherium:-

* Voting
* Agreements
* Banking Systems
* Shipping
* Crowd funding
* Domain names

Benefits of Etherium:-

* Availability
* Privacy
* Security
* Less ambiguity
* Rapid deployments
* Network size
* Data coordination

Drawbacks of Etherium:-

* Complicated programming language
* Volatile cryptocurrency
* Low transaction rate

Laboratory Record
of FBCT

Roll No.: 160120737157
Experiment No.: 03
Sheet No.: 08
Date: 09/08/2022

Aim:- Introduction to solidity Program Structure, Compilation and Deployment.

Description:-

Basic structure of a solidity Program, the compilation process and Deployment steps.

Solidity Program Structure:-

A solidity Program is typically organized into the following components:

Pragma Directive:- Specifies the version of the solidity compiler

Import Statement:- Used to import other solidity files or external dependencies.

Contract definition:- Used to define state variables, functions, events, modifiers.

State variables:- Store the persistent data of the contract such as of types uint, string, address or custom types.

Functions:- Defines the behaviour of contract.

Events:- Used to emit information from the contract that can be observed by external entities.

Modifiers:- Used to modify the behaviour of functions.

Compilation Process:-

→ Solidity compiler is used to compile a solidity program which include solc (command-line compiler), Remix IDE, and solidity plugin for various integrated development environment (IDEs)

→ The compilation process involves the following steps:

(i) Use the solidity compiler (solc) to compile your solidity program, specify the desired version using pragma directive.

(ii) Ensure that the solidity compiler version matches the pragma version specified in your code.

(iii) Check for any compilation errors or warnings. Resolve them before proceeding to the deployment step.

(iv) If the compilation is successful, you will obtain the compiled bytecode and Application Binary Interface (ABI) of the contract. These will be used for deployment.

Deployment steps:-

Deploying a solidity contract involves the following steps:

(i) Select a blockchain network or Ethereum-compatible network to deploy your contract.

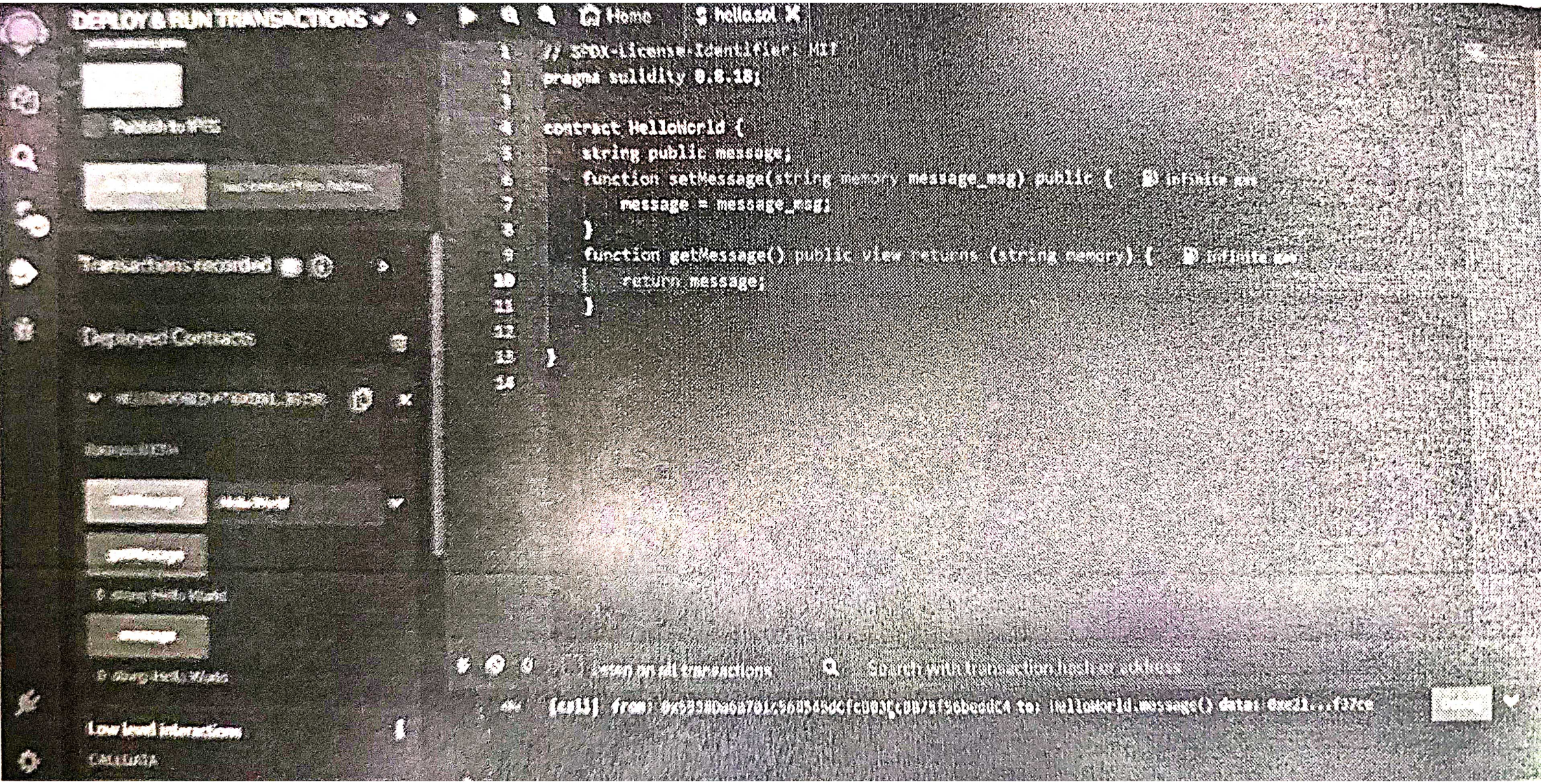
(ii) Choose a deployment tool or platform such as Remix IDE, Truffle, or Hardhat. These tools provide a convenient interface to interact with the network and deploy contracts.

(iii) configure the deployment parameters, including the selected network, account credentials, gas limits and deployment costs.

(iv) Use the deployment tool to deploy the compiled bytecode and ABI to the chosen network. This will create a new instance of your contract on the blockchain.

(v) Once deployed, you will receive a contract address, which you can use to interact with your contract from other applications or smart contracts.

(vi) It's important to note that deploying a contract to a live network incurs gas fees, which needs to be paid in cryptocurrency (such as Ether on the Etherium network).


DEPLOY & RUN TRANSACTIONS
Transactions recorded
Deployed Contracts
Low level interactions
Home
hello.sol
// SPDX-License-Identifier: MIT
pragma solidity 0.8.18;
contract HelloWorld {
string public message;
function setMessage(string memory message_msg) public {
message = message_msg;
}
function getMessage() public view returns (string memory) {
return message;
}
}
Search with transaction hash or address

Aim:- Creating & deploying the small contract and get "Hello World" on the blockchain network.

Steps:-

Step1 - Create a new solidity file.

Create a new file with '.sol' extension & open it in solidity deployment environment.

Step2:- Write the smart contract code inside '.sol' file

write the code for the smart contract.

Code:-

```
pragma solidity ^0.8.0;
contract HelloWorld{
    string public message;
    function setMessage(string memory msg) public{
        message = message;
    }
    function getMessage() public view return (string memory)
    {
      return message;
    }
}
```

Output:

set Message = Hello World

get Message

0: string : Hello World

message

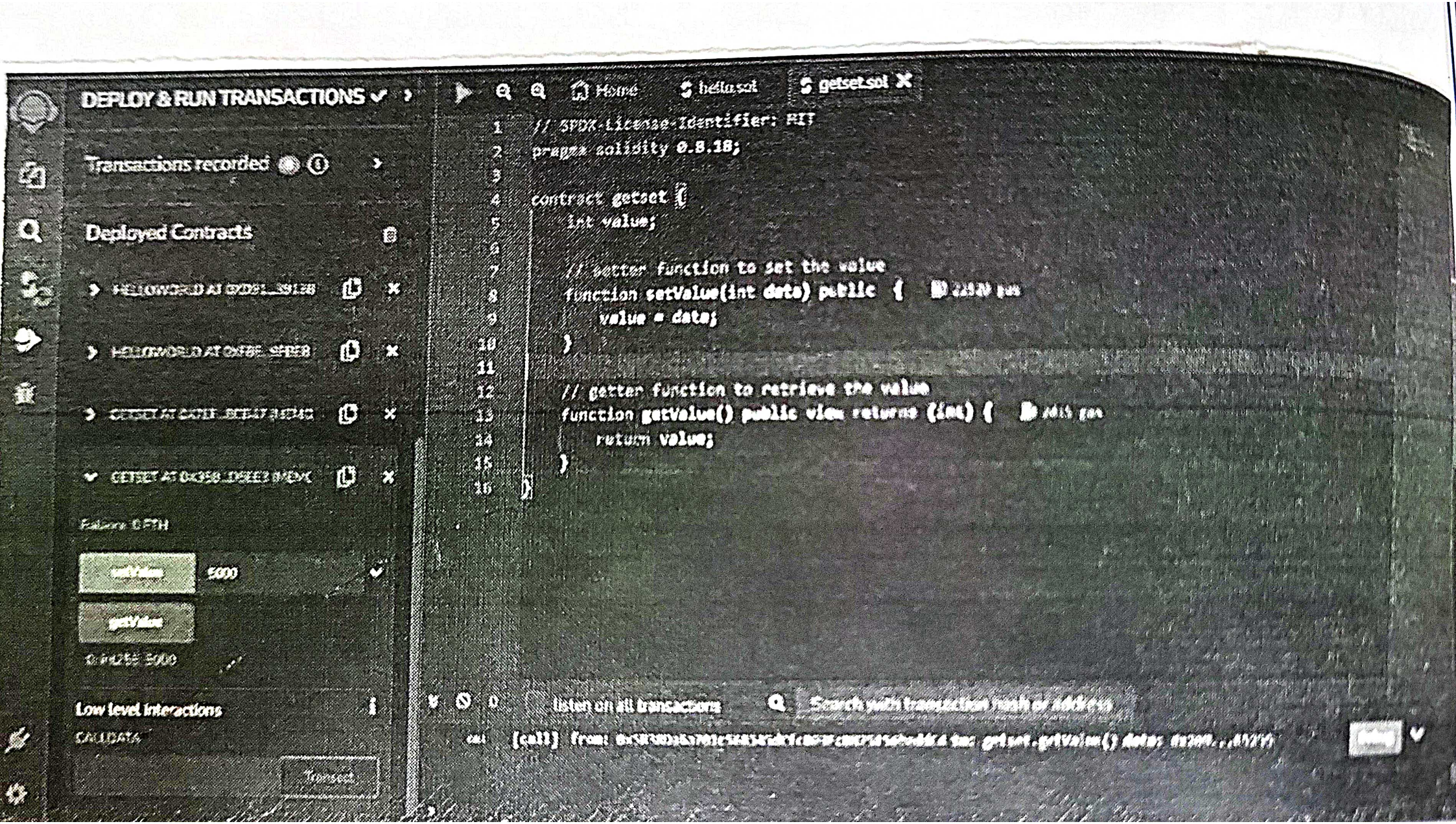
0: string : Hello World

Step 3:- Compile the smart-contract

Use solidity compiler or development tool (such as Remix IDE) to compile the smart contract.

Step 4:- Deploy the smart contract choose a block chain network or Etherium-compatible network to deploy your smart contract.

Step 5:- Interact with the smart contract Once deployed, you can interact with the contract using its function.


DEPLOY & RUN TRANSACTIONS
Transactions recorded
Deployed Contracts
Low level interactions
CALLDATA
Transact
Home
hello.sol
getset.sol
// SPDX-License-Identifier: MIT
pragma solidity 0.8.18;
contract getset {
int value;
// setter function to set the value
function setValue(int data) public {
value = data;
}
// getter function to retrieve the value
function getValue() public view returns (int) {
return value;
}
5000
listen on all transactions
Search with transaction hash or address

Aim:- Develop a smart contract to create a function setter and getter to set and get a value.

Description:-

Getter- Getter function is used to retrieve the variable value.

Setter:- Setter function is used to set the variable value.

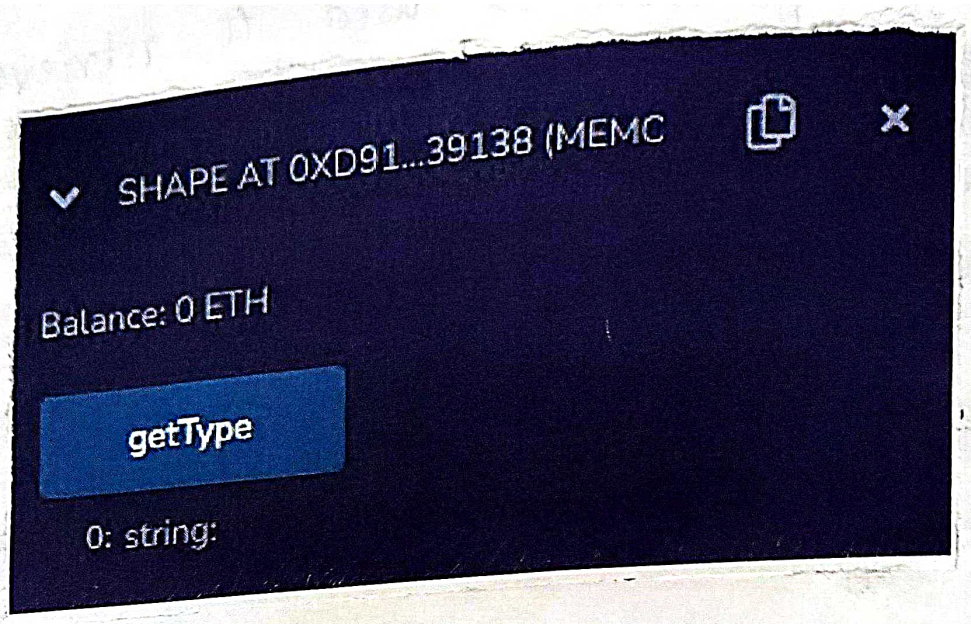
Program:-

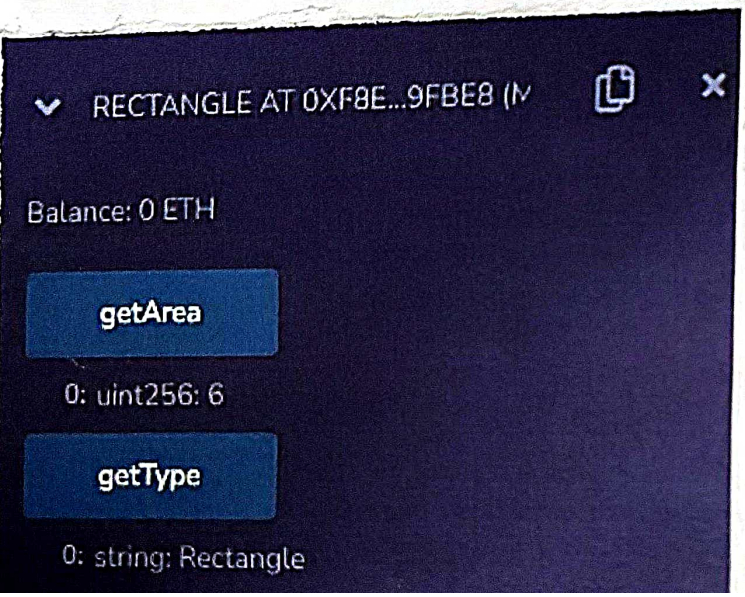
```
pragma solidity ^0.8.0;
contract ValueContract{
    uint private value; // Private variable to store the value
    // setter function to set the value
    function setValue(uint newValue) public{
        value = newValue;
    }
    // Getter function to retrieve the value
    function getValue() public view returns(uint){
        return value;
    }
}
```

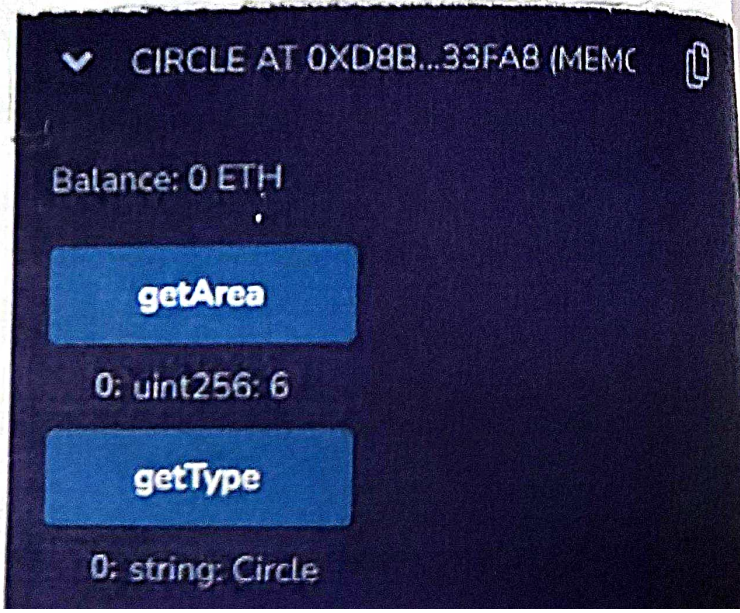
Output:-

setValue=69

getValue = 0: uint256: 69


SHAPE AT 0XD91...39138 (MEMC
Balance: 0 ETH
getType
0: string:


RECTANGLE AT 0XF8E...9FBE8 (M
Balance: 0 ETH
getArea
0: uint256: 6
getType
0: string: Rectangle


CIRCLE AT 0XD8B...33FA8 (MEMC
Balance: 0 ETH
getArea
0: uint256: 6
getType
0: string: Circle

Laboratory Record
of FBCT

Roll No.: 160120737157
Experiment No.: 06 (a)
Sheet No.: 14
Date: 23/02/2023

Aim: Develop solidity contract to illustrate inheritence.

Description:-

Inheritence is a mechanism in which one object aquires all the properties and behaviours of a parent object.

Program:-

```
pragma solidity ^0.8.0;
// parent contract
contract shape{
    string internal shapeType;
    constructor (string memory _type){
        shapeType = _type;
    }
    function getType() public view returns (string memory){
        return shapeType;
    }
}
// child contract inheriting from shape
contract Rectangle is shape{
    int private width;
    int private height;
    constructor (int _width, int _height) shape ("Rectangle"){
        width = _width;
        height = _height;
    }
```

Output:-

contract:- circle - demo.sol

Rectangle - demo.sol

Shape - demo.sol

① Circle - demo.sol

Deploy:- radius=5

getArea:-18

getType:- Circle

② Rectangle - demo.sol

Deploy:- width, height=2,

getArea:- 6

getType:-Rectangle

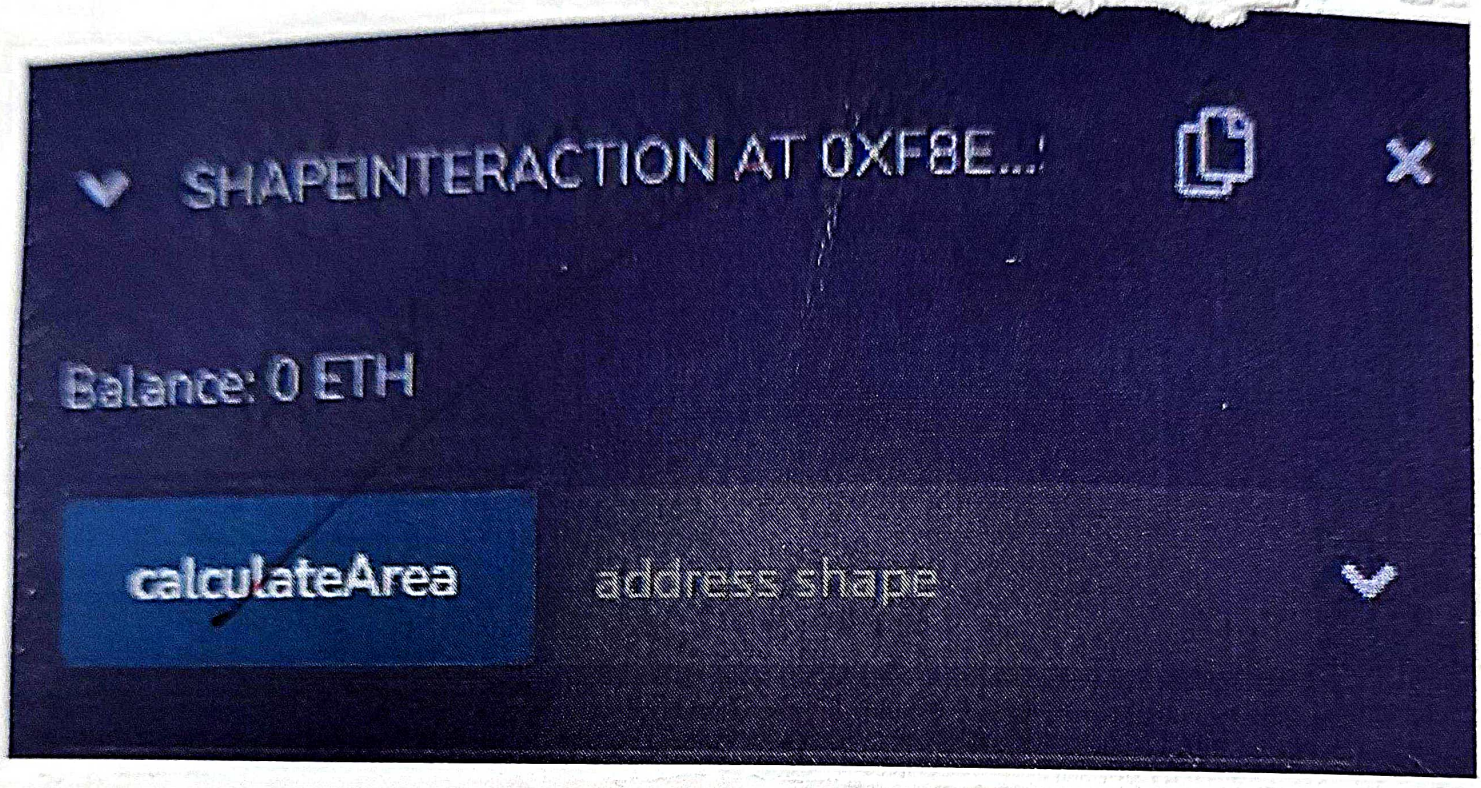
③ Shape - demo.sol

Deploy:- string -type="demo"

getType:- "demo"

```
    function getArea() public view returns (int) {
        return width * height;
    }
}
contract Circle is Shape {
    int private radius;
    constructor (int _radius) Shape ("circle")
    {
        radius = _radius;

        function getArea() public view returns (int) {
            return calculateArea();
        }
        function calculateArea() private view returns (int) {
            int pi = 3;
            return (pi * radius * radius) / 4;
        }
```


SHAPEINTERACTION AT 0XF8E...
Balance: 0 ETH
calculateArea
address shape

Laboratory Record
of FBCT

Roll No.: 160120737157
Experiment No.: 6(b)
Sheet No.: 16
Date: 23/08/2022

Aim: Develop solidity contracts to illustrate a) inheritance.
b) Polymorphism

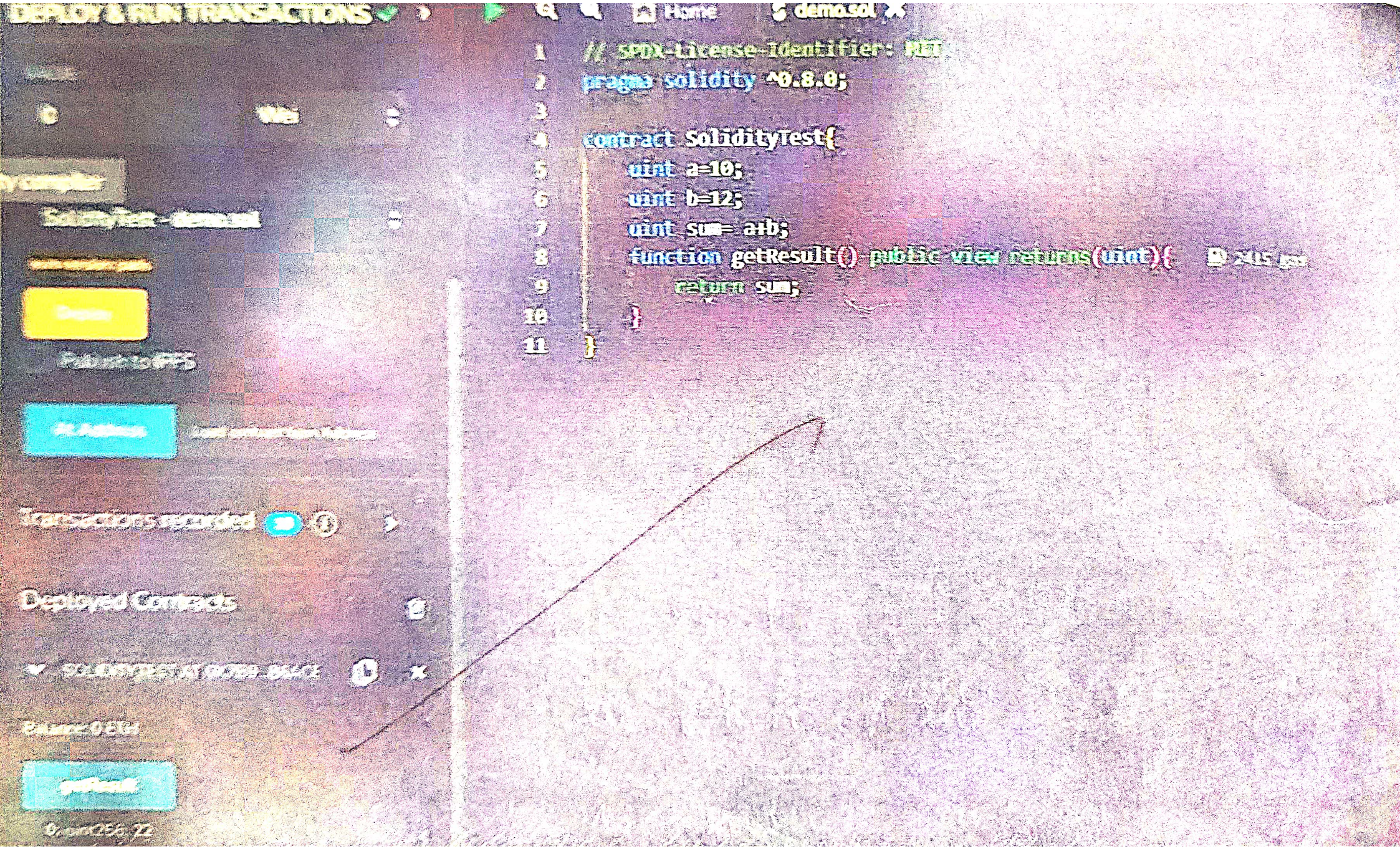
Description:-

→ Contract polymorphism means using multiple contract instances interchangeably when they are related to each other by using inheritance. This helps in calling the child contract functions using the instance of the parent contract.

Code:-

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
abstract contract Shape {
    string internal shapeType;
    constructor (string memory _type){
        shapeType=_type;
    }
    function getArea() public virtual view returns (uint);
}
contract Rectangle is Shape{
    uint private width;
    uint private height;
    constructor (uint _width, uint _height) Shape ("Rectangle"){
        width = _width;
        height=_height;
    }
```

```
        function getArea() public view virtual override returns(uint)
        {
            return width * height;
        }
}
contract Circle is Shape{
        uint private radius;
        uint private constant pi = 31415;
        constructor(uint _radius) Shape ("Circle"){
            radius = _radius;
        }
        function getAreal() public view virtual override returns
                                                    (uint)
        {
            return (pi * radius * radius)/10000;
        }
}
contract ShapeInteraction{
        function calculateArea(Shape shape) public view returns(uint)
        {
            return shape.getArea();
        }
}
```


DEPLOY & RUN TRANSACTIONS
Home
demo.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract SolidityTest{
uint a=10;
uint b=12;
uint sum= a+b;
function getResult() public view returns(uint){
return sum;
}
}
Transactions recorded
Deployed Contracts

Laboratory Record
of FDCT

Roll No.: 160120737157
Experiment No.: 07
Sheet No.: 18
Date: 13/09/2023

Aim: To familiarize with the working of Remix Etherium tool.

Description & Steps: Remix IDE is generally used to compile & run solidity smart contracts. Below are the steps for the compilation, execution and debugging of the smart contract.

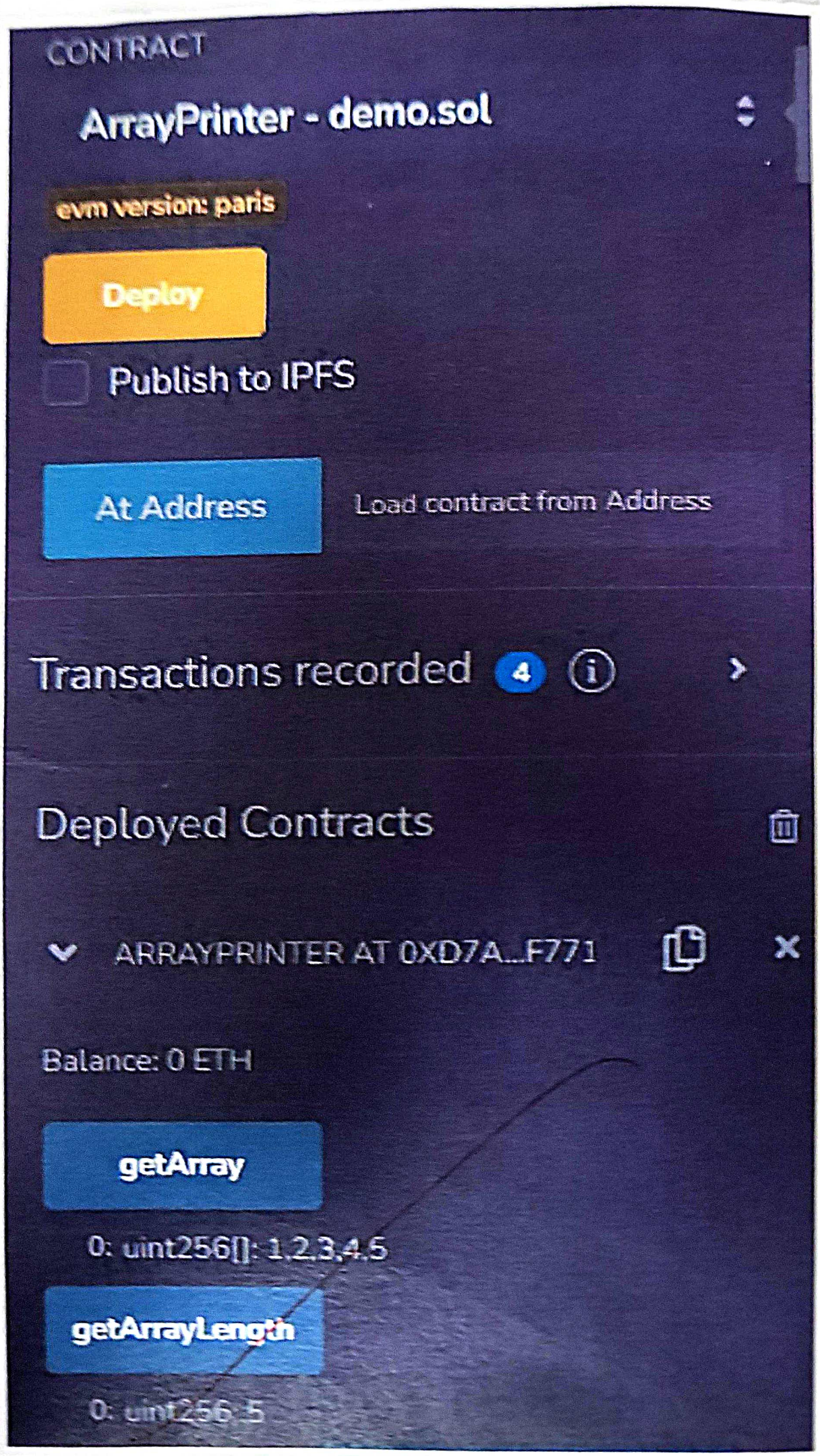
Step 1: Open Remix IDE on any of your browsers, select on New File and click on Solidity to choose the environment.

Step 2: Write the smart contract in the code section, and click the Compile button under the Compiles window to compile the contract.

Step 3: To execute the code, click on the Deploy button under Deploy and Run Transactions window.

Step 4: After deploying the code click on the method calls under the drop-down of deployed contracts to run the program, and for output, check to click on the drop-down on the console.

Step 5: For debugging click on the Debug button corresponding to the method call in the console.


CONTRACT
ArrayPrinter - demo.sol
evm version: paris
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 4
Deployed Contracts
ARRAYPRINTER AT 0XD7A...F771
Balance: 0 ETH
getArray
0: uint256[]: 1,2,3,4,5
getArrayLength
0: uint256: 5

Laboratory Record
of Fundamentals of Blockchain Technology

Roll No.: 160120737157
Experiment No.: 08
Sheet No.: 19
Date: 04/10/2023

Aim: Design a smart contract on Remix Ethereum to print array.

Description:-

→ Arrays are used to store lists or collections of data elements.

→ Role of Array is generally Iteration, Token Management, Dynamic Sizing, etc.

Code:-

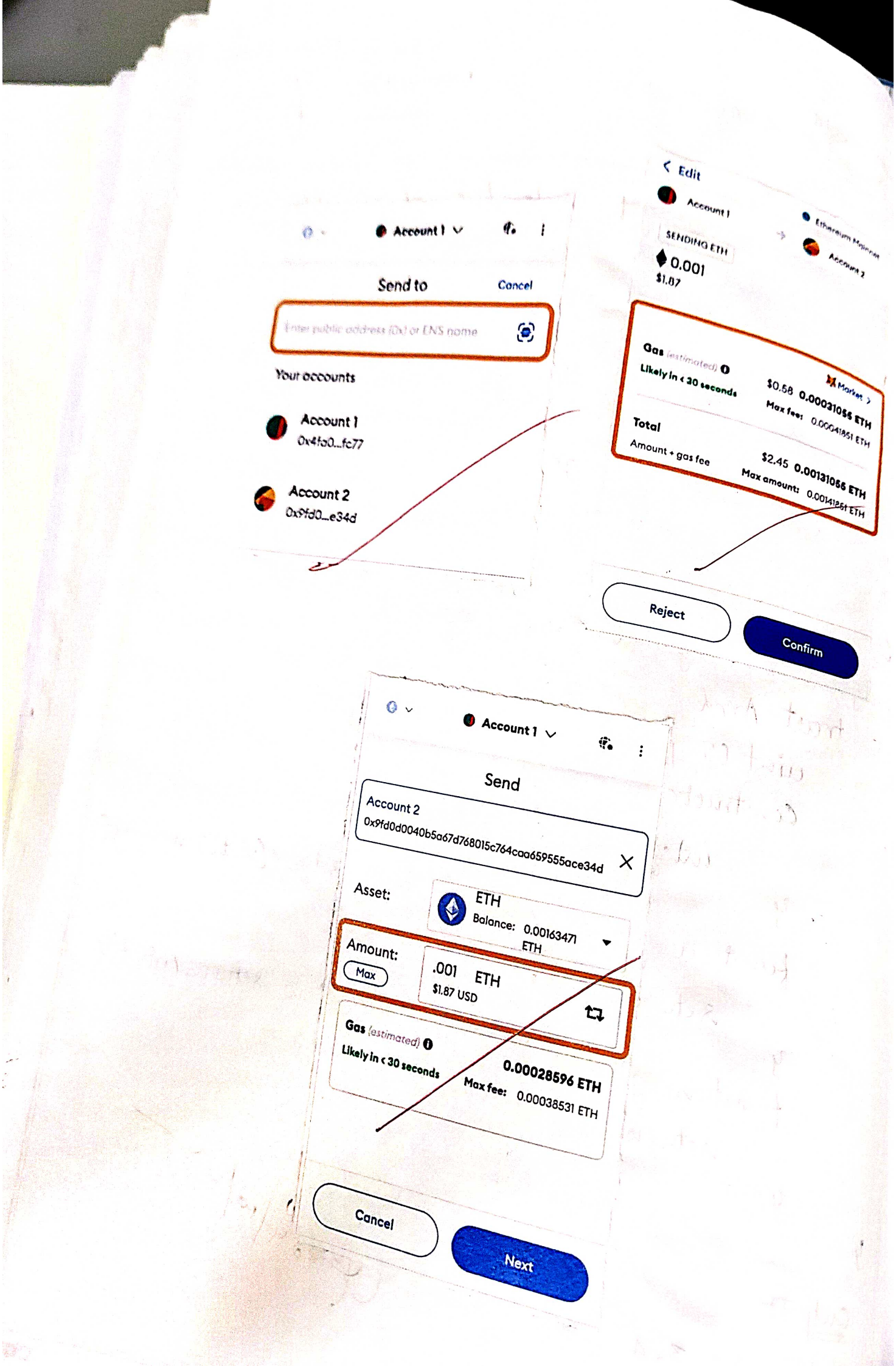
```
pragma solidity ^0.8.0;
contract Arr{
    uint[] public integers;
    constructor(){
        integers = [1,2,3,4];
    }
    function getter() public view returns(uint[] memory){
        return integers;
    }
    function getArrLength() public view returns(uint){
        return integers.length;
    }
}
```

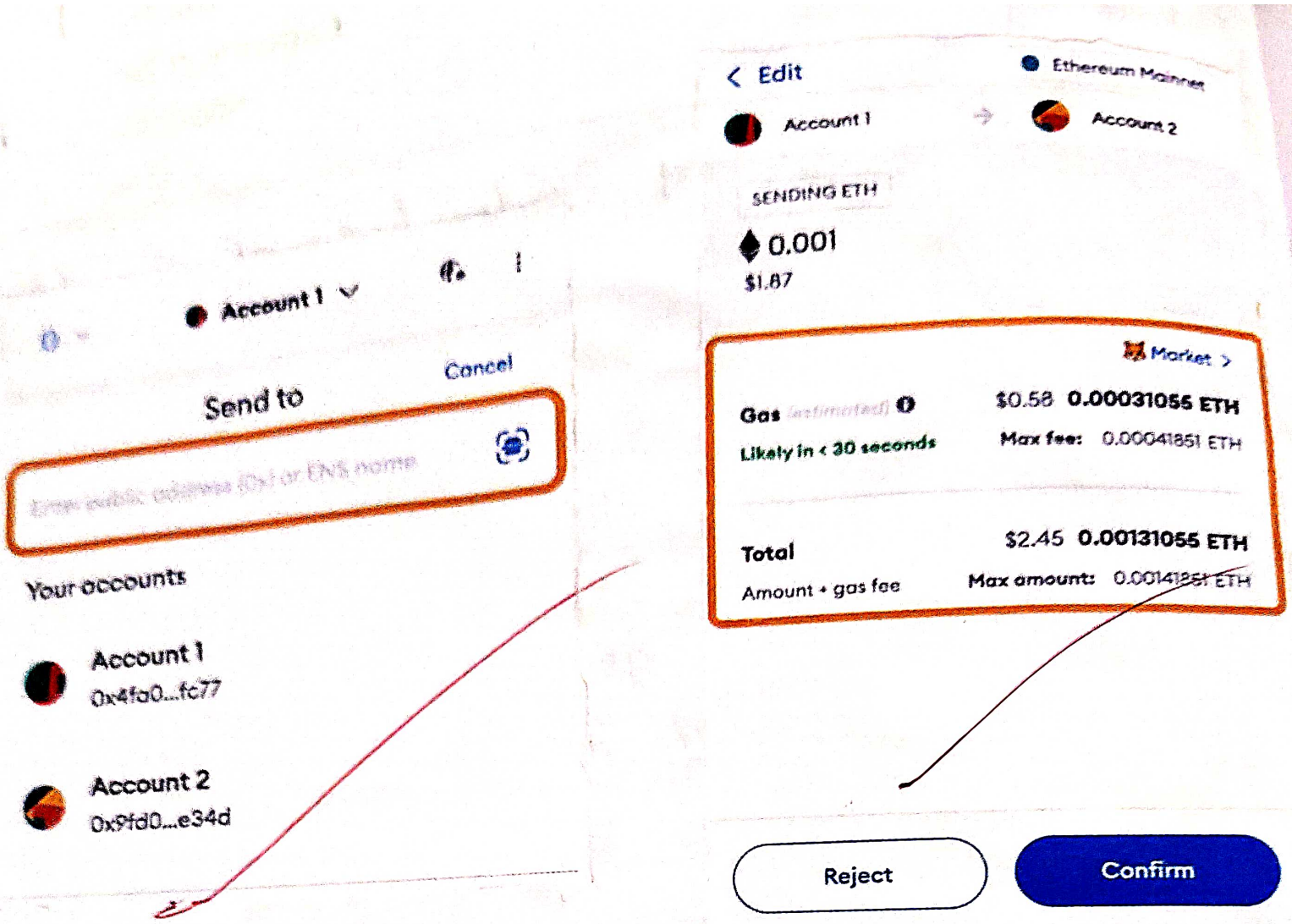
Output:-

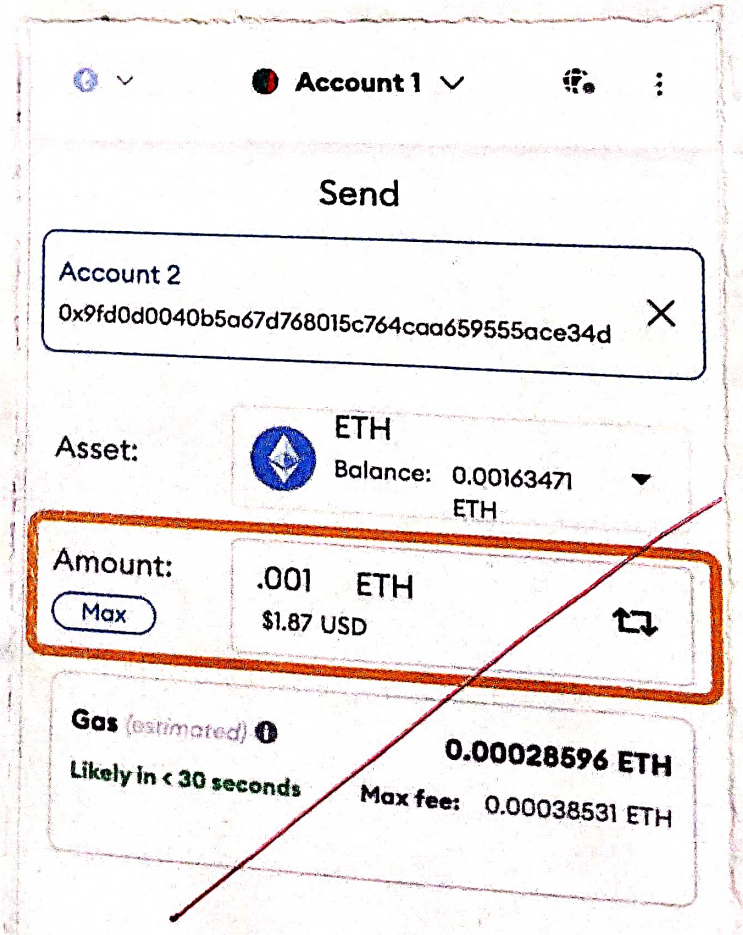
getter : 1,2,3,4

getArrLength() : 4

04/10/


Account 1
Send to
Cancel
Enter public address (0x) or ENS name
Your accounts
Account 1
0x4fb0...fc77
Account 2
0x9fd0...e34d
Edit
Account 1
SENDING ETH
0.001
$1.87
Ethereum Mainnet
Account 2
Gas (estimated)
Likely in < 30 seconds
$0.58 0.00031058 ETH
Max fee: 0.00041851 ETH
Market
Total
Amount + gas fee
$2.45 0.00131058 ETH
Max amount: 0.00141851 ETH
Reject
Confirm
Account 1
Send
Account 2
0x9fd0d0040b5a67d768015c764caa659555ace34d
Asset:
ETH
Balance: 0.0016347 ETH
Amount:
Max
.001 ETH
$1.87 USD
Gas (estimated)
Likely in < 30 seconds
0.00028596 ETH
Max fee: 0.00038531 ETH
Cancel
Next


Account 1
Cancel
Send to
Your accounts
Account 1
0x4fa0...fc77
Account 2
0x9fd0...e34d
Edit
Ethereum Mainnet
Account 1
Account 2
SENDING ETH
0.001
$1.87
Market
Gas
$0.58 0.00031055 ETH
Likely in < 30 seconds
Max fee: 0.00041851 ETH
Total
$2.45 0.00131055 ETH
Amount + gas fee
Max amount:
Reject
Confirm


Account 1
Send
Account 2
0x9fd0d0040b5a67d768015c764caa659555ace34d
Asset:
ETH
Balance: 0.00163471 ETH
Amount:
Max
.001 ETH
$1.87 USD
Gas (estimated)
Likely in < 30 seconds
0.00028596 ETH
Max fee: 0.00038531 ETH

Laboratory Record
of FACT

Roll No.: 160120737157
Experiment No.: 09
Sheet No.: 20
Date: 04/10/2023

Aim:- Setup a Simple Etherium wallet and use it to send and receive Ethers.

Description:

→ MetaMask is a free software cryptocurrency wallet that allows users to interact with the Etherium blockchain. It is available as a browser extension or mobile app.
→ It allows users to store tokens, interact with Dapps, Trade Etherium, Buy, sell and swap coins.

Procedures

→ From the landing page of your wallet, make sure you're in the account from which you want to transact, and hit the "Send" button in the middle of the screen.
→ Now you need to input the public address of the recipient. If you already have addresses saved in your address book, they will appear now.
→ Enter the amount of tokens you want to send and click next.
→ Now you're presented with the estimated gas fees of your transaction, which you can also adjust. Double-checking the recipient address before clicking "Confirm" to proceeding with the transaction is generally a good idea
→ You will then be redirected to the homepage, where you can see a list of your recent transactions on the "Activity" tab.

Seller Organization

ORG1

```
application:

seller = ORG1;
buyer = ORG2;
transfer(CAR1, seller, buyer);
```

```
car contract:

query(car):
 get(car);
 return car;

transfer(car, buyer, seller):
 get(car);
 car.owner = buyer;
 put(car);
 return car;

update(car, properties):
 get(car);
 car.colour = properties.colour;
 put(car);
 return car;
```

Buyer Organization

ORG2

```
application:

seller = ORG2;
buyer = ORG1;
transfer(CAR2, seller, buyer);
```

```
user@user-VirtualBox:~/fabric-samples/test-network$ sudo docker exec peer0.org1.example.com pee
r channel list
2020-07-20 04:24:03.803 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connec
tions initialized
Channels peers has joined:
user@user-VirtualBox:~/fabric-samples/test-network$
```

```
user@user-VirtualBox:~/fabric-samples/test-network$ sudo docker ps
CONTAINER ID        IMAGE                               COMMAND                  CREATED             STATUS              PORTS
                    NAMES
42b0cc7bd413        hyperledger/fabric-peer:latest      "peer node start"        53 seconds ago      Up 47 seconds       0.0.0.0:
7051->7051/tcp      peer0.org1.example.com
0bbd3434dd98        hyperledger/fabric-peer:latest      "peer node start"        53 seconds ago      Up 47 seconds       7051/tcp
, 0.0.0.0:9051->9051/tcp   peer0.org2.example.com
bb282a207caa        hyperledger/fabric-orderer:latest   "orderer"                53 seconds ago      Up 48 seconds       0.0.0.0:
7050->7050/tcp      orderer.example.com
```

Laboratory Record
of FACT

Roll No.: 160120737157
Experiment No.: 10
Sheet No.: 21
Date: 11/10/2023

Aim: Implementation of Hyperledger Fabric Demo.

Description:-

→ Hyperledger Fabric platform is a private blockchain framework that's used to develop blockchain-based applications, networks, and more. It is an open source platform that's designed for use in enterprise contexts.

→ Fabric networks are permissioned, meaning all participating member's identities are known and authenticated

Prerequisites for Hyperledger Fabric setup:- & Commands

* Curl Installation
  $ sudo apt-get install curl

* NodeJS Installation
  $ curl -SL https://deb.nodesource.com/setup_10.x | sudo -E bash -
  $ sudo apt-get install nodejs

* Git Installation
  $ sudo apt-get install git

* Python Installation
  $ sudo apt-get install python

* Lib Tools Installation
  $ sudo apt-get install libltdl-dev

* Docker CE (Community Edition) & Docker Compose Install
  $ wget https://download.docker.com/linux/.../docker-ce.deb
  $ sudo dpkg -i docker-ce.deb

Laboratory Record
of FBCT

Roll No.: 160120733157
Experiment No.: 10
Sheet No.: 22
Date: 11/10/2023

$ Dockes sudo apt install docker
$ sudo pip install docker-compose (through python)
$ sudo apt install doccker-compose (without python)

* Hyperledger Installation

Run the below command to download and setup Fabric.

$ curl -sSL http://bit.ly/2ysbOFE | bash -s

Note: If you get "permission denied" error then fix this using the below command

$ sudo chmod 666 /var/run/docker.sock

Steps to Procedure:-

Step 1: Go to the fabric-samples folder by using command

$ cd fabric-samples

Step 2: Go to the test-network

$ cd test-network

Step 3: start your test-network

$sudo ./network.sh up

(Note: To check docker containers: sudo docker ps

This shows you three docker containers

(i) One for Org1 peer node
(ii) One for Org2 peer node
(iii) One for Orderer)

* When you start the network, you will also not get any channel by default. You can check the channel by using below command.

$ sudo docker exec peer0.org1.example.com peer channel list

* The command will show that, we don't have any channel created.

Step 4: Create new channel by using below command

$ sudo ./network.sh createChannel -c testchannel.

* This will create a new channel with the name "testchannel"
* To verify this channel, run below command on both the peers.

$ sudo docker exec peer0.org1.example.com peer channel list

$ sudo docker exec peer0.org2.example.com peer channel list

Step 5: To stop the Network, you need to run the below command.

$ sudo ./network.sh down