Experiment No: 08

●**Aim**: **To implement Smart Contract using solidity.**

●**Theory**:

SMART CONTRACT:

A smart contract is a self-executing computer program that automatically enforces the terms of an agreement between two or more parties. It is a piece of code that is stored on a blockchain and is executed by the network of computers that participate in the blockchain.

Smart contracts are designed to be secure, transparent, and tamper-proof. They use cryptographic algorithms to ensure that they cannot be altered once they have been deployed to the blockchain. They also eliminate the need for trust between parties, as the code automatically enforces the terms of the contract.

LOCAL BLOCKCHAIN:

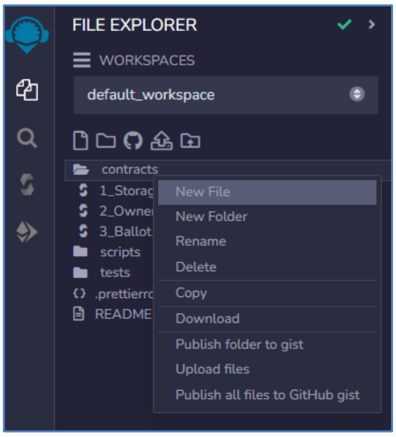
A local blockchain is a blockchain network that is hosted on a local machine or a private network. It is typically used for development, testing, and experimentation purposes.
Local blockchains can be set up using various blockchain platforms such as Ethereum, Hyperledger Fabric, or Corda. Developers can install these platforms on their local machines and configure them to create a private blockchain network.

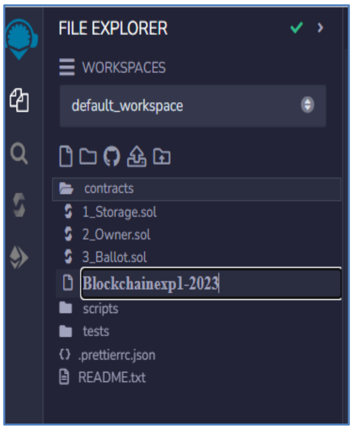
Local blockchains can be useful for testing and debugging smart contracts in a controlled environment before deploying them to a public blockchain network. They also provide a way for developers to experiment with new blockchain technologies and build decentralized applications without incurring the costs associated with deploying on a public blockchain.

To develop and deploy a smart contract on a local blockchain , we can follow these steps:

1) **Open the Remix IDE in our web browser**
- https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.18+commit.87f61d96.js

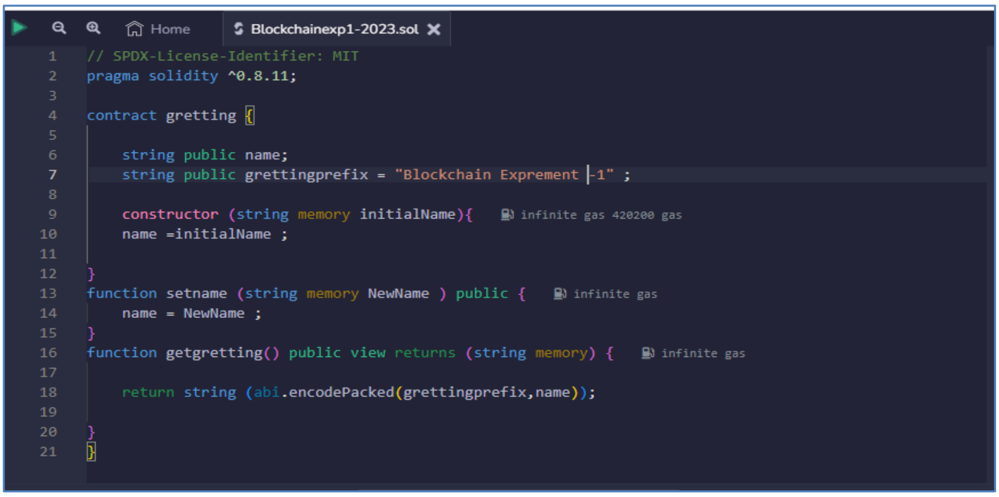
2 Create a new file called “Blockchainexp1-2023. sol”





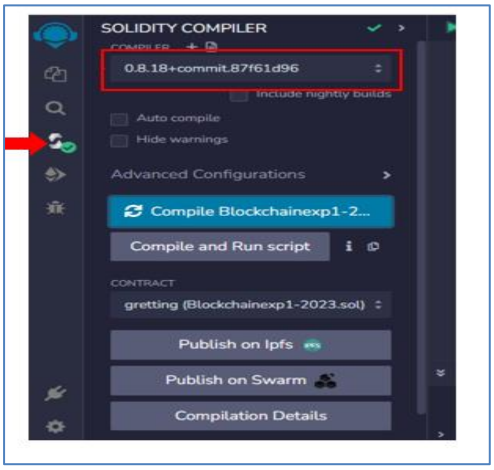
2)

3) **write the code written below:**



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.11;

contract gretting {

    string public name;
    string public grettingprefix = "Blockchain Exprement -1" ;

    constructor (string memory initialName){    infinite gas 420200 gas
    name =initialName ;

}
function setname (string memory NewName ) public {    infinite gas
    name = NewName ;
}
function getgretting() public view returns (string memory) {    infinite gas

    return string (abi.encodePacked(grettingprefix,name));

}
}
```

4) **Compile the contract by clicking on the "Solidity Compiler" tab on the left-hand side of the screen. Make sure the compiler version matches the one specified in the contract (^0.8.18):**



0.8.18+commit.87f61d96

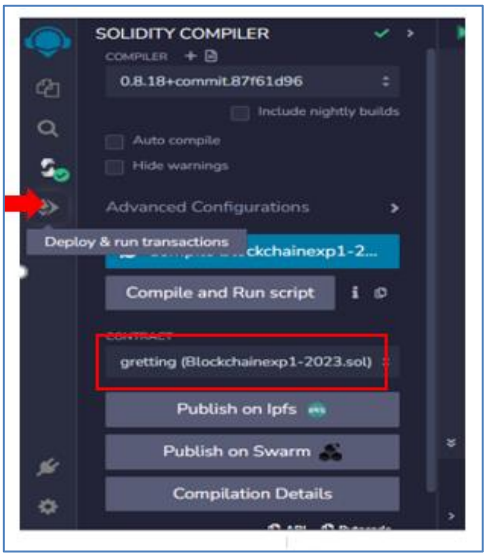
□ Include nightly builds

□ Auto compile

□ Hide warnings

gretting (Blockchainexp1-2023.sol)

5) **After the contract has been compiled successfully, go to the "Deploy & Run Transactions" tab on the left-hand side of the screen and Click the "Deploy" button to deploy the contract:**



0.8.18+commit.87f61d96

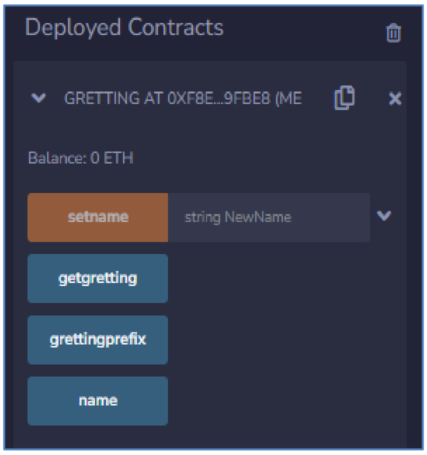
□ Include nightly builds

□ Auto compile

□ Hide warnings

gretting (Blockchainexp1-2023.sol)

6) Once the contract is deployed, you can interact with it using the "Deployed Contracts" section.

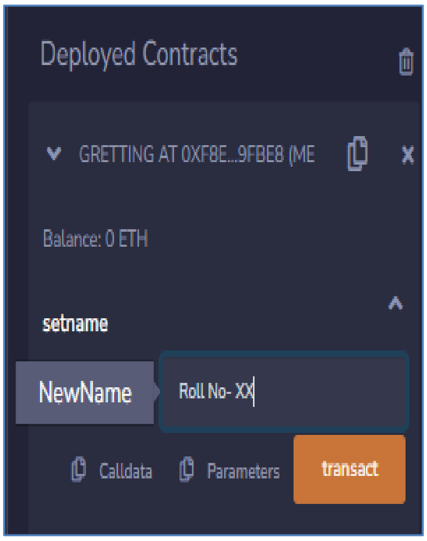


Deployed Contracts

GRETTING AT 0XF8E...9FBE8 (ME

Balance: 0 ETH

7) To set the name, click on the "setName" button and enter the name you want to set in the "Value" field. Then click "transact" to send the transaction to the blockchain.



Deployed Contracts

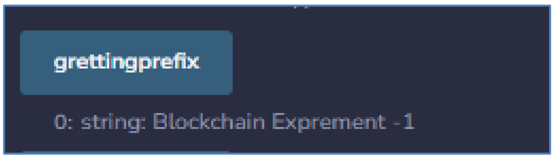
GRETTING AT 0XF8E...9FBE8 (ME

Balance: 0 ETH

setname

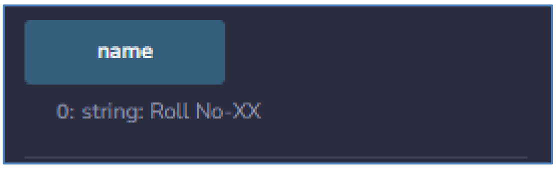
NewName: Roll No- XX

8) To print greeting which we have set by default in our code ,please click on the greetingprefix



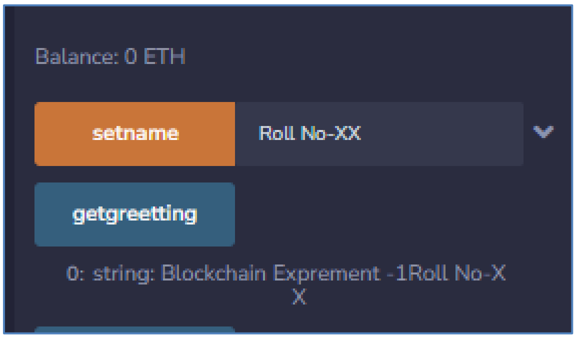
0: string: Blockchain Exprement -1

9) To print the name, click on the "Name" button. The name should appear in the "decoded output" section below the button.



```
0: string: Roll No-XX
```

10) To print the Name with greeting, click on the " getgreeting" button. The name should appear in the "decoded output" section below the button.



setname: Roll No-XX

```
0: string: Blockchain Exprement -1Roll No-X
X
```

● **Conclusion**:

In this program, we learned how to Develop and Deploy Smart Contracts in Solidity - Remix - Ethereum IDE.