Prof Ram Meghe College of Engineering & Management, Badnera – Amravati
Department of Computer Science and Engineering

---------- **7**

KS 08 EMERGING TECHNOLOGY LAB IV
Blockchain Fundamentals

List of Experiment

Experiment No.	Title
1	Understanding the Three Pillars of Blockchain: Decentralization, Transparency, and Immutability
2	Creating a Simple Blockchain in Python
3	Introduction to Smart Contracts using Remix IDE
4	Develop a Solidity Smart Contract to Store and Retrieve an Integer *(Your Class Roll No).*
5	Write a Solidity Program to Perform Basic Arithmetic Operations.
6	Write a Solidity Program to Perform Looping Statements (for loop, while loop).
7	Write a Solidity Program to Implement Basic Cryptographic Functions (*keccak256, sha256, ripemd160*).
8	Mini Project using Solidity - Group Implementation of a Decentralized Application (DApp)

KS 08 EMERGING TECHNOLOGY LAB IV
Blockchain Fundamentals

Experiment No. 1

Title: Understanding the Three Pillars of Blockchain: Decentralization, Transparency, and Immutability.

Objective:

To study the fundamental characteristics of blockchain technology through simulation — focusing on the three core principles: **Decentralization**, **Transparency**, and **Immutability**.

Theory

Blockchain Technology

A blockchain is basically a living list of records, called as "blocks". These blocks are connected to each other by the diverse cryptographic mechanisms. In the category of data structures, this can be related to the concept of a Linked List. In Blockchain, the initial block is known as the "Genesis Block". This naming convention is basically a major commendation to Satoshi Nakamoto. The domain of crypto currency was pioneered by a bogus naming convention. It can be related to a random scenario of a person or a group of persons, represented by a peculiar name “Satoshi Nakamoto”. In the year 2008, for the purpose of Bitcoin this name was utilized. The technology that was used behind the Bitcoin spectrum was “Block-Chain”. Initially the structure of a block has basically 3 components namely data, hash of current block and hash of previous block. As an illustration in general, the concept of block chain can be depicted with “m” blocks forming a chain where m can be any random positive integer.

Three pillars of blockchain

Decentralization

The true meaning of decentralization is not having a central unit. Now if we take this concept in Blockchain it means that blockchain is autonomous and does not have a central governing unit.

Transparency

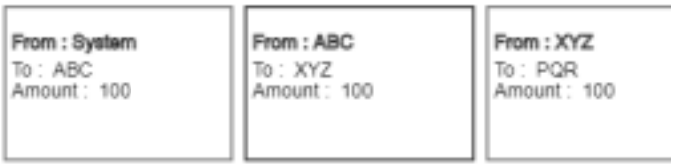
Transparency in real life means something with zero opacity. Now if we take this concept in Blockchain, it means that blockchain has zero privacy to be exact when we talk about transactions, all the transactions are public and can be viewed by anyone on the network.

Immutability

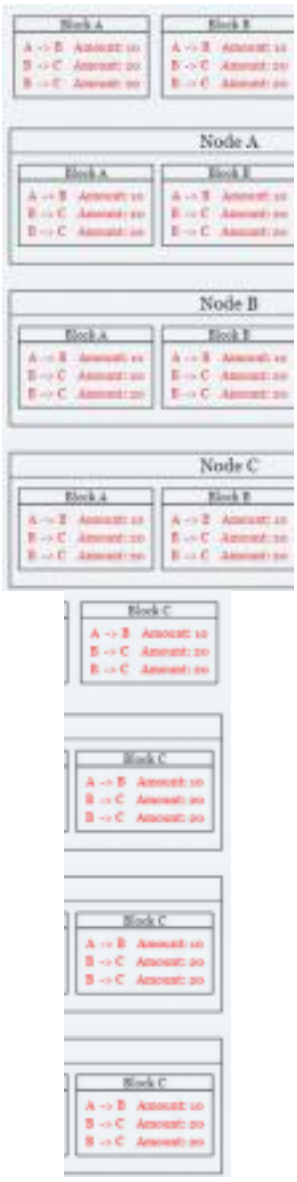
Here immutable means exactly what the word means in any real life i.e. something that cannnot be altered. So when we talk about blockchain it means that once a transaction is pushed into blockchain it cannot be altered.

Blockchain Demo: https://andersbrownworth.com/blockchain/

Open ledger



Distributed Ledger **Distributed Ledger**	**Immutability** **Immutability**



Conclusion:

This experiment demonstrated the three essential features of blockchain technology:

This experiment demonstrated the three essential features of blockchain technology:

This experiment demonstrated the three essential features of blockchain technology: ·

Decentralization: No single controlling entity; all nodes

Decentralization: No single controlling entity; all nodes participate equally.

participate equally.

· Transparency: All peers have access to a copy of the ledger.

Transparency: All peers have access to a copy of the ledger.

· Immutability: Once recorded, data cannot be changed without altering all subsequent blocks.

Immutability: Once recorded, data cannot be changed without altering all subsequent blocks.
Immutability: Once recorded, data cannot be changed without altering all subsequent blocks.

Viva Questions:

1. What is a blockchain?
2. What are the three main pillars of blockcha
   What are the three main pillars of blockchain?
3. Why is decentralization important in blockchain?
   Why is decentralization important in blockchain?
4. What do you understand by “immutability” in blockchain?
   What do you understand by “immutability” in blockchain?
5. How is transparency achieved in blockchain networks?
   How is transparency achieved in blockchain networks?

KS 08 EMERGING TECHNOLOGY LAB IV
Blockchain Fundamentals

Experiment No. 2

Title: Creating a Simple Blockchain in Python.

Objective: To write a Python program that creates a simple blockchain with multiple blocks linked using hash values and mined using Proof of Work.

Problem Statement:

Write a Python program to create a simple blockchain consisting of five blocks (including a Genesis Block). Each block should store the following:

- Data
- Timestamp
- Current hash (SHA-256)
- Previous hash

Blocks should be connected using the *previous_hash* field to maintain the integrity of the chain.

Code:

Code:

```python
import hashlib, time

class Block:
    def __init__(self, data, prev_hash='0'):
        self.timestamp = time.ctime()
        self.data = data
        self.prev_hash = prev_hash
        self.hash = self.compute_hash()

    def compute_hash(self):
        content = f"{self.timestamp}{self.data}{self.prev_hash}"
        return hashlib.sha256(content.encode()).hexdigest()

# Create blockchain
chain = [Block("Genesis Block")]
for i in range(1, 5):
    chain.append(Block(f"Block {i}", chain[-1].hash))

# Display the chain
for i, block in enumerate(chain):
    print(f"\nBlock {i}")
    print(f"Timestamp  : {block.timestamp}")
    print(f"Data       : {block.data}")
    print(f"Prev Hash  : {block.prev_hash}")
    print(f"Hash       : {block.hash}")
```

Output:

Conclusion:

A simple blockchain was created using Python where each block was linked using cryptographic hashes. This demonstrated the basic structure and integrity of a blockchain.

Viva Questions:

1. What is a Genesis Block?
2. What fields are typically included in a block?
3. What is a hash? Why is it important in a blockchain?
4. What is SHA-256 and why is it used here?
5. What would happen if someone changes the data of a block?

---------- 7

KS 08 EMERGING TECHNOLOGY LAB IV
Blockchain Fundamentals

Experiment No. 3

Title: Introduction to Smart Contracts using Remix IDE.

Objective: To study the working of smart contracts and execute basic Solidity programs using Remix IDE in a simulated Ethereum environment.

Problem Statement:

Use the Remix IDE to create, deploy, and test simple smart contracts written in Solidity. Understand how Ethereum Virtual Machine (EVM) executes contract functions and handles transactions using gas. The goal is to get hands-on experience with writing, compiling, deploying, and interacting with smart contracts.

URL: https://remix.ethereum.org/

Smart Contract Code:

Write a contract to print a string ("Your Name").

```
pragma solidity ^0.8.0;

contract Name {
 string public message = "Saurabh Shah";
}
```

Output:

Conclusion:

A basic Solidity smart contract was successfully written, compiled, deployed, and tested using Remix IDE.

Viva Questions:

1. What is a smart contract?
2. What is Remix IDE used for?
3. What is Solidity?
4. What is the role of the Ethereum Virtual Machine (EVM)?
5. What is the purpose of the constructor in a smart contract?

KS 08 EMERGING TECHNOLOGY LAB IV
Blockchain Fundamentals

Experiment No. 4

Title: Develop a Solidity Smart Contract to Store and Retrieve an Integer ***(Your Class Roll No).***

Objective: To implement and deploy a smart contract using Remix IDE that stores the student's own class roll number and retrieves it using getter and setter functions.

Problem Statement:

Write a Solidity smart contract named *MyRollNumber* that stores **your own class roll number** using an integer variable.

The contract should include:

- A function to store the roll number
- A function to retrieve the stored roll number

Deploy and test the contract using Remix IDE.

URL: https://remix.ethereum.org/

Smart Contract Code:

```
pragma solidity ^0.8.0;

contract MyRollNumber {
 uint private rollNo;

 function setRollNo(uint _rollNo) public {
 rollNo = _rollNo;
 }

 function getRollNo() public view returns (uint) {
 return rollNo;
 }
}
```

Output:

Conclusion:

The smart contract was successfully deployed to store and retrieve the student's own roll number.

Viva Questions:

1. What is a smart contract in Ethereum?
2. What is the purpose of using Remix IDE?
3. What does the uint data type represent in Solidity?
4. What does the private keyword do for a variable?
5. What is the difference between view and pure functions?

KS 08 EMERGING TECHNOLOGY LAB IV
Blockchain Fundamentals

Experiment No. 5

Title: Write a Solidity Program to Perform Basic Arithmetic Operations.

Objective: To implement a Solidity smart contract that performs basic arithmetic operations such as addition, subtraction, multiplication, and division.

Problem Statement:

Create a smart contract named Calculator that takes two unsigned integers as input and provides separate functions to:

- Add the two numbers
- Subtract one number from the other
- Multiply the numbers
- Divide one number by the other

Deploy and interact with the contract using Remix IDE.

URL: https://remix.ethereum.org/

Smart Contract Code:

```
pragma solidity ^0.8.0;

contract Calculator {
 function add(uint a, uint b) public pure returns (uint) {
 return a + b;
 }

 function subtract(uint a, uint b) public pure returns (uint) {
 return a - b;
 }

 function multiply(uint a, uint b) public pure returns (uint) {
 return a * b;
 }

 function divide(uint a, uint b) public pure returns (uint) {
 require(b != 0, "Division by zero not allowed");
 return a / b;
 }
}
```

Output:

Conclusion:

A Solidity contract was successfully created to perform basic arithmetic operations using functions and deployed on Remix IDE.

Viva Questions:

1. What is the purpose of the pure keyword in Solidity functions?
2. Why do we use require(b != 0) in the divide function?
3. Can this contract store any values on the blockchain? Why or why not?
4. What is the difference between public and private in function visibility?
5. What data types are used in this contract?

KS 08 EMERGING TECHNOLOGY LAB IV
Blockchain Fundamentals

Experiment No. 6

Title: Write a Solidity Program to Perform Looping Statements (for loop, while loop).

Objective: To understand and implement looping constructs (`for`, `while`) in Solidity using smart contracts deployed via Remix IDE.

Problem Statement:

Create a smart contract in Solidity that:

· Demonstrates a for loop to return the sum of first N natural numbers

· Demonstrates a while loop to count how many times a number can be divided by 2 before reaching zero

Test both functionalities using Remix IDE.

URL: https://remix.ethereum.org/

Smart Contract Code:

```
pragma solidity ^0.8.0;

contract LoopExamples {

function forLoop(uint n) public pure returns (uint) {
uint sum = 0;
for (uint i = 1; i <= n; i++) {
sum += i;
}
return sum;
}

function whileLoop(uint n) public pure returns (uint) {
uint sum = 0;
uint i = 1;
while (i <= n) {
sum += i;
i++;
}
return sum;
}
```

```
function doWhileLoop(uint n) public pure returns (uint) {
uint sum = 0;
uint i = 1;
if (n == 0) return 0; // do-while runs at least once
do {
sum += i;
i++;
} while (i <= n);
return sum;
}
}
```

Output:

Conclusion:

All three types of loops (`for`, `while`, and `do...while`) were successfully implemented and tested in Solidity using Remix IDE.

Viva Questions:

1. What is a loop? Why is it used in programming?
2. What are the different types of loops supported in Solidity?
3. How is a for loop different from a while loop?
4. What is the key characteristic of a do...while loop?
5. Why should we handle edge cases (like n = 0) in loops?

KS 08 EMERGING TECHNOLOGY LAB IV
Blockchain Fundamentals

Experiment No. 7

Title: Write a Solidity Program to Implement Basic Cryptographic Functions (*keccak256, sha256, ripemd160*).

Objective: To learn the basics of cryptography and implement fundamental cryptographic functions using Solidity's inbuilt methods.

Problem Statement:

Solidity provides built-in functions for cryptographic hashing. In this experiment, write a smart contract named CryptoFunctions that:

- Accepts a string input
- Hashes the input using:
  - keccak256 (SHA-3)
  - sha256
  - ripemd160
- Returns the result as bytes/hash

URL: https://remix.ethereum.org/

Smart Contract Code:

```
pragma solidity ^0.8.0;

contract CryptoFunctions {

 function getKeccak256(string memory input) public pure returns (bytes32) {
 return keccak256(abi.encodePacked(input));
 }

 function getSha256(string memory input) public pure returns (bytes32) {
 return sha256(abi.encodePacked(input));
 }

 function getRipemd160(string memory input) public pure returns (bytes20) {
 return ripemd160(abi.encodePacked(input));
 }
}
```

Output:

Conclusion:

Solidity provides multiple inbuilt cryptographic functions like keccak256, sha256, and ripemd160. These functions were implemented and tested successfully on Remix IDE.

Viva Questions:

1. What is a cryptographic hash function?
2. What are the main properties of a cryptographic hash function?
3. Can a hash function be reversed? Why or why not?
4. What is the difference between SHA-256 and Keccak-256?
5. What is the output size of Keccak256, SHA-256, and RIPEMD-160?

KS 08 EMERGING TECHNOLOGY LAB IV
Blockchain Fundamentals

Experiment No. 8

Title: Mini Project using Solidity - Group Implementation of a Decentralized Application (DApp)

Group Size: Maximum 4 students per group

Objective: To design and implement a basic decentralized application (DApp) using **Solidity** and **Remix IDE**, applying concepts like smart contracts, cryptographic functions, mappings, loops, and struct management.

General Problem Statement:

Each group must choose and develop ***one of the following (or similar)*** projects using Solidity: 1. Decentralized To-Do List

2. Simple Voting System
3. Blockchain-based Certificate Verifier
4. Student Result Manager
5. Leave Management System
6. Expense Tracker
7. Book Catalog
8. Personal Digital Diary
9. Polling or Survey App
10. Blockchain-based Reminder System

Each contract should:

· Store and manage structured data
· Have at least 3 write functions and 2 read (view) functions
· Be tested with sample data on Remix IDE
· Contain validations using require and if where needed

URL: https://remix.ethereum.org/

Expected Deliverables:

A one-page report with:

· Project title & group member names
· Objective
· Source code
· Description of smart contract
· Screenshots
· Conclusion

https://github.com/Aniket-Engg