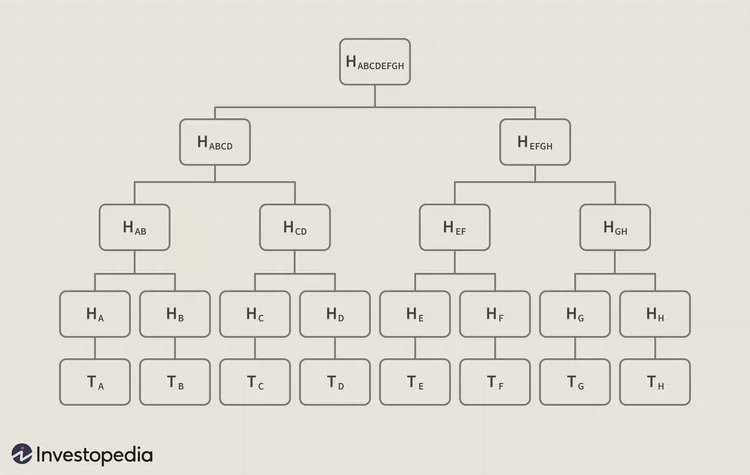
**Blockchain Technology Lab**

**Lab – 3**

**Aim :** Implement Merkle Tree to learn data structure used in blockchain.



**Code:**

*import* hashlib

*class* TreeNode:

*def* \_\_init\_\_(self, value):

*self*.value = value

*self*.left = None

*self*.right = None

*def* hashData(data):

*return* hashlib.sha256(data.encode('utf-8')).hexdigest()

*def* createTree(dataList):

*if* len(dataList) % 2 != 0:

        dataList.append(dataList[-1])

    nodes = [TreeNode(hashData(data)) *for* data *in* dataList]

*while* (len(nodes) > 1):

        level = []

*for* i *in* range(0, len(nodes), 2):

            left = nodes[i]

            right = nodes[i+1]

            contcatedData = hashData(left.value+right.value)

            newNode = TreeNode(contcatedData)

            newNode.left = left

            newNode.right = right

            level.append(newNode)

        nodes = level

*return* nodes[0]

*def* verifyTree(rootHash, dataList):

*return* rootHash == createTree(dataList).value

*if* \_\_name\_\_ == "\_\_main\_\_":

*while* (True):

        choice = int(input("create Tree (0) / verify tree (1) : "))

*if* (choice == 0):

            n = int(input("Enter number of node : "))

            data = []

*for* i *in* range(0, n):

                data.append(input(*f*"Enter data for node {i+1} : "))

            tree = createTree(data)

            print(*f*"Hash of Root : {tree.value}")

*elif* (choice == 1):

            rootHash = input("Enter hash of the root : ")

            n = int(input("Enter number of node : "))

            data = []

*for* i *in* range(0, n):

                data.append(input(*f*"Enter data for node {i+1} : "))

*if* (verifyTree(rootHash, data)):

                print("Verification Successfull")

*else*:

                print("Verification Failed")

*else*:

            print("Invalid choice")

**Output:**

**A computer screen with white text

Description automatically generated**