Ex : 10

Title: Design and develop a Program to implement MerkleTree typically used in Blockchain.

Problem Description: Generate a Merkle tree to store a given block of transactions in a tamper-proof manner.

Method: In a Merkle Tree, each leaf node is labelled with the hash value of a data block and each non-leaf node is labelled with the hash value of its child nodes labels. Ensure that the program should uses SHA-256 hash function to hash transaction data continuously till the Merkle root is obtained.

Theory Reference: Module 5

Explanation:

- The program takes a list of transaction strings.
- Each string is hashed into a leaf node.
- The leaf nodes are then recursively combined into internal nodes by hashing pairs of nodes, until only one node remains — the Merkle root.

Algorithm:

Step 1: Create Leaf Nodes:

- For each transaction string, hash it using SHA-256 and create a leaf node containing this hash.
- Example: `Transaction = "Tx1: Alice -> Bob 100 BTC"`

  `Leaf Node Hash = SHA-256("Tx1: Alice -> Bob 100 BTC")`

Step 2: Create Internal Nodes:

- Pair up the leaf nodes and hash their combined data to create internal nodes.

- If the number of nodes is odd, duplicate the last node to make the count even.
- For each pair of nodes, concatenate their hashes and hash the concatenated value to create a parent node.

Step 3: Recursively Build the Tree:

- Once a level of nodes is hashed, treat the resulting internal nodes as the new input nodes, and repeat the process until only one node (the root) remains.

Step 4: Output the Merkle Root:

- The last remaining node is the Merkle root. This hash represents the cryptographic summary of all the transactions in the tree.

Output:

```
== RESTART: C:\Users\sonim\AppData\Local\Programs\Python\Python37\merklegp1.py =
Merkle Root Hash:
4e05c7945c20122ec850271219ad10992bce602cbb904039674968c277432b95
>>>
```