NAME:PAREENITA A.SHIRSATH PRN:221101062 ROLL.NO:57 B.E.A.I.&.D.S.

BCL EXPERIMENT:1

```python
from typing import List
import hashlib

class Node:
    def __init__(self, left, right, value: str) -> None:
        self.left: Node = left
        self.right: Node = right
        self.value = value

    @staticmethod
    def hash(val: str) -> str:
        return hashlib.sha256(val.encode('utf-8')).hexdigest()

    @staticmethod
    def doubleHash(val: str) -> str:
        return Node.hash(Node.hash(val))

class MerkleTree:
    def __init__(self, values: List[str]) -> None:
        self._buildTree(values)

    def _buildTree(self, values: List[str]) -> None:
        # Create leaf nodes
        leaves: List[Node] = [Node(None, None, Node.doubleHash(e)) for e in values]
        # Duplicate the last element if the number of elements is odd
        if len(leaves) % 2 == 1:
            leaves.append(leaves[-1])
        # Build the tree recursively
        self.root: Node = self.__buildTreeRec(leaves)

    def __buildTreeRec(self, nodes: List[Node]) -> Node:
        half: int = len(nodes) // 2
        if len(nodes) == 2:
            return Node(nodes[0], nodes[1], Node.doubleHash(nodes[0].value + nodes[1].value))
        left: Node = self.__buildTreeRec(nodes[:half])
        right: Node = self.__buildTreeRec(nodes[half:])
        value: str = Node.doubleHash(left.value + right.value)
        return Node(left, right, value)

    def printTree(self) -> None:
        self.__printTreeRec(self.root)

    def __printTreeRec(self, node) -> None:
        if node is not None:
            print(node.value)
            self.__printTreeRec(node.left)
            self.__printTreeRec(node.right)

    def getRootHash(self) -> str:
        return self.root.value

def test() -> None:
    elems = ["Hello", "Good", "Morning", "Yash"]
    mtree = MerkleTree(elems)
    print(mtree.getRootHash())

test()
```

```
c0512930ae034cef4ffe34754fa4653bb419fbf7667a5845d871c81d467003d6
```

What can I help you build?