**实现过程：**

Implement Merkle Tree as of RFC 6962 in any programming language that you prefer

对于该过程的实现，首先做以下解释：

一、对于 Merkle Tree 的存储采用了二维列表的形式：

1.将同一层的 hash 值，按数据的前后顺序，按写入一个 列表中.

2.将记录各层 hash 列表，按深度顺序，写入总的列表中。 并将这个总列表作为存储 Merkle Tree 的形式。

二、实现过程中，对于数据的父子节点关系，我们通过列表的下 标来确定。 之后进行对过程进行了实现：

1．明二维列表，并计算树深和叶子节点数量。

2．计算数据 hash 值，并将其写入叶子节点中。

3．对于每两个子节点，计算其加和后的 hash 值，并写到其父 节点那一层的列表中。并对同一层中的节点，持续执行这个 过程，生成下一层（父节点层）Merkle Tree 的节点。

4．对于单数个节点的层中的最后一个节点，直接将其写入下一 层（父节点层）中。

5．持续执行。

三、循环 Depth（第 1 步中计算的树的深度）次，就完成了 Merkle Tree 的生成过程。

四.进行实验测试： 输入测试数据，调用 CreatTree（）函数，并打印。之后会打印出生成的 Merkle Tree，同一层在同一个列表中。 子列表在父列表的前面，第一个列表为叶子节点的列表。

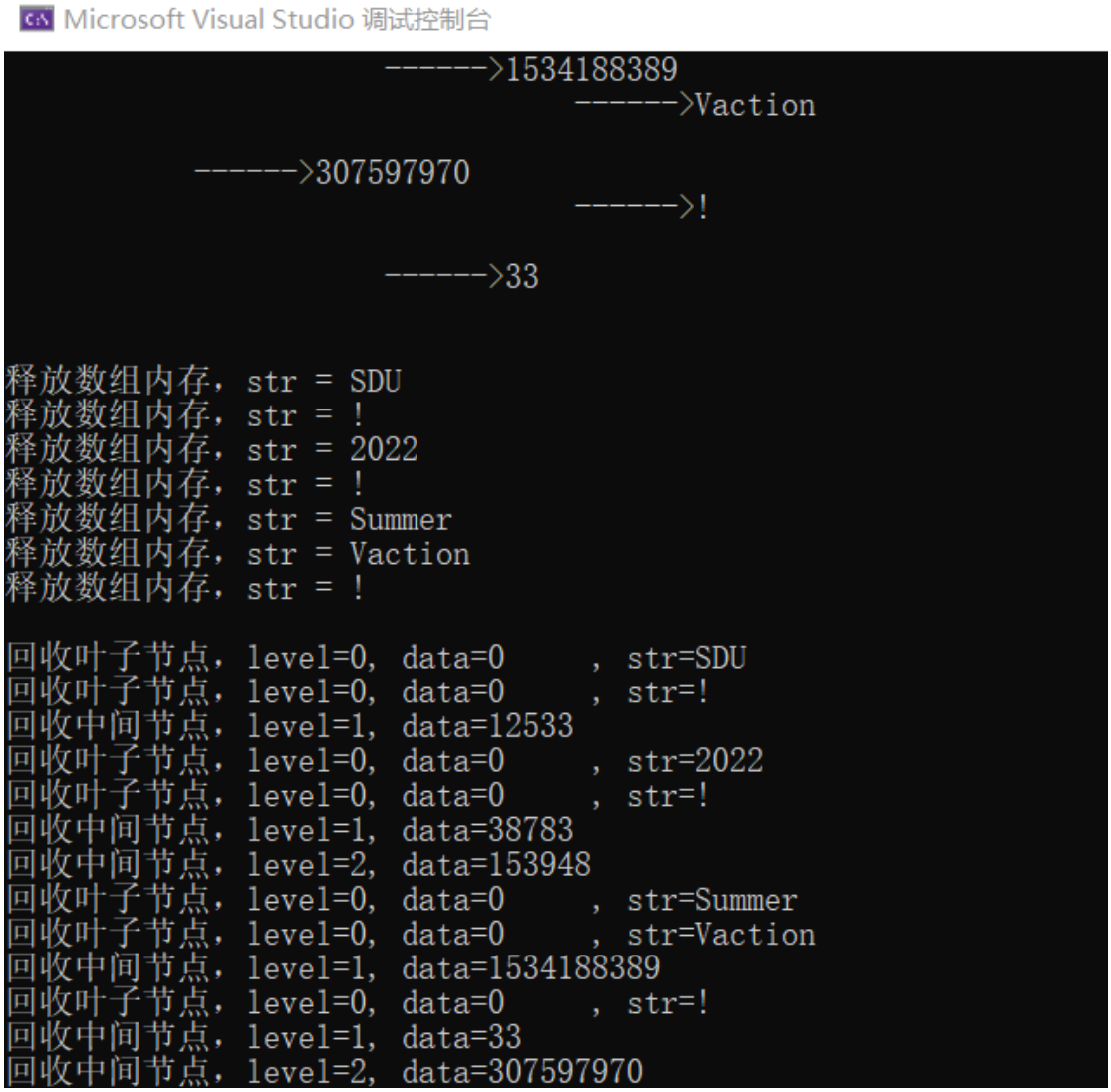
**运行指导：**

直接运行即可

**运行结果：**

Microsoft Visual Studio 调试控制台

```
原词组 SDU!2022!Summer Vaction!
解析到 7个词组

 [SDU]  [!]  [2022]  [!]  [Summer]  [Vaction]  [!]

新建叶节点 [133] tree_depth=0, level=0, data=0     , nums=7, str=SDU  , arr=SDU   (3)
新建头节点 [142] tree_depth=1, level=1, data=0
新建叶节点 [133] tree_depth=1, level=0, data=0     , nums=6, str=!    , arr=!     (1)
新建叶节点 [133] tree_depth=1, level=0, data=0     , nums=5, str=2022 , arr=2022  (4)
新建头节点 [209] tree_depth=2, level=2, data=0
新中间节点 [215] tree_depth=2, level=1, data=0
新建叶节点 [133] tree_depth=2, level=0, data=0     , nums=4, str=!    , arr=!     (1)
新建叶节点 [133] tree_depth=2, level=0, data=0     , nums=3, str=Summer, arr=Summer (6)
新建头节点 [209] tree_depth=3, level=3, data=0
新中间节点 [215] tree_depth=3, level=2, data=0
新中间节点 [226] tree_depth=3, level=1, data=0
新建叶节点 [133] tree_depth=3, level=0, data=0     , nums=2, str=Vaction, arr=Vaction (7)
新建叶节点 [133] tree_depth=3, level=0, data=0     , nums=1, str=!    , arr=!     (1)
新中间节点 [177] tree_depth=3, level=1, data=0
Merkle Tree 创建完毕！！！

开始打印当前 Merkle 树:
                                ------>SDU

                        ------>12533
                                ------>!

                ------>153948
                                ------>2022
```


Microsoft Visual Studio 调试控制台
------>1534188389
------>Vaction
------>307597970
------>!
------>33
释放数组内存，str = SDU
释放数组内存，str = !
释放数组内存，str = 2022
释放数组内存，str = !
释放数组内存，str = Summer
释放数组内存，str = Vaction
释放数组内存，str = !
回收叶子节点，level=0，data=0 ，str=SDU
回收叶子节点，level=0，data=0 ，str=!
回收中间节点，level=1，data=12533
回收叶子节点，level=0，data=0 ，str=2022
回收叶子节点，level=0，data=0 ，str=!
回收中间节点，level=1，data=38783
回收中间节点，level=2，data=153948
回收叶子节点，level=0，data=0 ，str=Summer
回收叶子节点，level=0，data=0 ，str=Vaction
回收中间节点，level=1，data=1534188389
回收叶子节点，level=0，data=0 ，str=!
回收中间节点，level=1，data=33
回收中间节点，level=2，data=307597970