1. **选择题**
2. 一棵完全二叉树上有1001个结点，其中叶子结点的个数是（ ）。

A．250 B． 500 C．254 D．501

1. 一个具有1025个结点的二叉树的高h为（ ）。

A．11 B．10

C．11至1025之间 D．10至1024之间

1. 某二叉树T有n个结点，设按某种顺序对T中的每个结点进行编号，编号值为1,2……n。如果有如下性质：T中任意结点v，其编号等于左子树上的最小编号减1，而v的右子树的结点中，其最小编号等于v左子树上结点的最大编号加1，则这是按（ ）编号的。
2. 中序遍历序列 B. 先序遍历序列

C. 后序遍历序列 D. 层次遍历序列

1. 二叉树的叶子结点在先序、中序、后序遍历序列中的相对次序（ ）

A. 不发生改变 B. 发生改变 C. 不确定

1. 若二叉树采用二叉链表存储结构，要交换其所有分支结点左、右子树的位置，利用（ ）遍历方法最合适。
2. 前序 B．中序 C．后序
3. **简答题：**

试找出满足下列条件的二叉树。

1、先序序列与后序序列相同。

2、中序序列与后序序列相同

3、先序序列与中序序列相同

4、中序序列与层次遍历序列相同。

5、先序序列与后序序列正好相反

提交要求：除了空树及只有一个根结点的二叉树外，找找层数大于1的满足条件的二叉树，每个小题画出一个满足条件的二叉树，并写出两个遍历序列验证。

1. **编程实现的二叉链式存储方式的二叉树的数据结构方法**

//2. 销毁二叉链式存储的二叉树，释放每个结点所分配的内存

// 注意是没有头结点的二叉树

bool DestroyBinTree(BiTree bt);

//3.1 从先序遍历的字符串中创建二叉树，将创建的二叉树用返回值返回到调用者

// 先序遍历的字符串中用字符'!'表示空结点

//参考教材 P131 实现，这里的参数nStringStart，表示创建树时

//，从preordString的那个字符开始读入

//使用引用形式的参数传递方式进行参数传递，这样可以在递归调用时，

// 每读入一个字符后，

// nStringStart向后移动一个位置，并将这个值的改变带回到调用者

BiTree CreateBinTree(char \* preordString, int & nStringStart);

//4.1 先序遍历二叉树，将结果输出到控制台（stdc::out）

// 返回值：空树返回false，非空树返回true

bool PreOrderTraverse(BiTree bt);

//4.2 中序遍历二叉树，将结果输出到控制台（stdc::out）

// 返回值：空树返回false，非空树返回true

bool InOrderTraverse(BiTree bt);

//4.3 后序遍历二叉树，将结果输出到控制台（stdc::out）

// 返回值：空树返回false，非空树返回true

bool PostOrderTraverse(BiTree bt);

//4.4 层序遍历二叉树，将结果输出到控制台（stdc::out）

// 返回值：空树返回false，非空树返回true

bool LevelOrderTraverse(BiTree bt);

//5 计算树的结点数目

int BiTreeNodeCount(BiTree bt);

//6 计算树的叶子结点数目

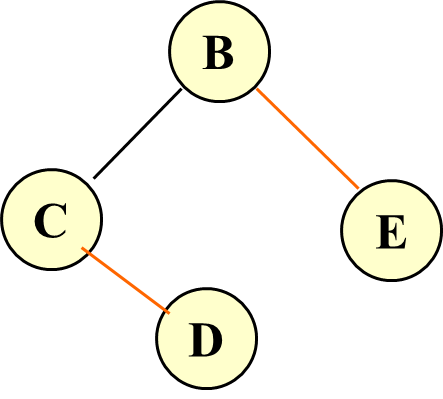
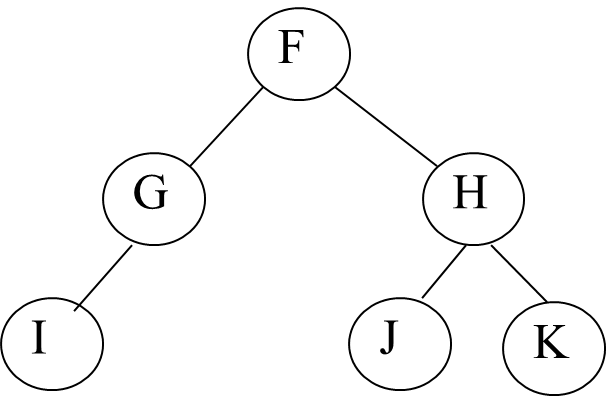
int BiTreeLeavesCount(BiTree bt);

//7 计算树的层数

int BiTreeLevelCount(BiTree bt);

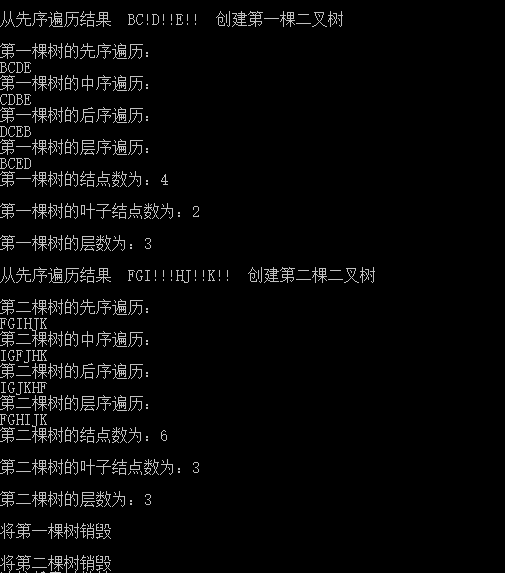
**在06BinTree.cpp中补充所缺的代码来完成作业：**

* + 1. 声明了所有的函数；给出了所需实现函数的空函数框架，补充每个函数的实现代码。
    2. 在main函数里面有两个字符串变量sTree1和sTree2，预先保存了两棵树的先序遍历结果供读入使用，两个字符串对应下图的两棵树

测试时，可用其他树的遍历结果替换字符串换不同的树进行测试。

* + 1. 请仔细阅读main函数，理解main函数调用的各个基本操作。
    2. 将程序调试正确
    3. 程序运行结果参考



1. **编程题（均可以在06BinTree.cpp中实现，自行设计函数参数和返回值并调试通过）**
2. 已知在二叉链表表示的二叉树中，root 为根结点，p和q分别指向二叉树中两个结点，试编写算法求距离它们最近的共同祖先。
3. 试给出算法求出一条从根到叶子结点的序列，其结点数为层数。
4. **编程题（选做，上交代码源文件或者手写代码均可以，自行设计函数参数和返回值并调试通过）**
5. 将后缀表达式读入，构建一棵表达式计算的二叉树，再将二叉树转换为中缀表达式，记得要在合适的位置加入括号维持计算顺序不变。