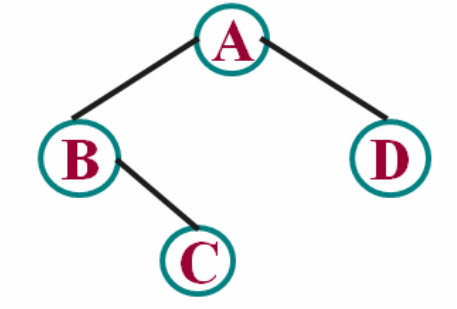
DS二叉树--二叉树构建与遍历

题目描述

给定一颗二叉树的逻辑结构如下图，（先序遍历的结果，空树用字符‘0’表示，例如AB0C00D00），建立该二叉树的二叉链式存储结构，并输出该二叉树的先序遍历、中序遍历和后序遍历结果



输入

第一行输入一个整数t，表示有t个二叉树

第二行起输入每个二叉树的先序遍历结果，空树用字符‘0’表示，连续输入t行

注：可以认为输入的数据均合法

输出

输出每个二叉树的先序遍历、中序遍历和后序遍历结果，不用输出空树'0'

样例输入

```
2
AB0C00D00
AB00C00
```

样例输出

```
ABCD
BCAD
CBDA
ABC
BAC
BCA
```

运行效果

2	输入二叉树数目
AB0C00D00	输入第1个二叉树的前序遍历（含'0'，表示空树）
ABCD	输出第1个二叉树的前序遍历（不含空树）
BCAD	输出第1个二叉树的中序遍历（不含空树）
CBDA	输出第1个二叉树的后序遍历（不含空树）
AB00C00	输入第2个二叉树的前序遍历（含'0'，表示空树）
ABC	输出第2个二叉树的前序遍历（不含空树）
BAC	输出第2个二叉树的中序遍历（不含空树）
BCA	输出第2个二叉树的后序遍历（不含空树）

程序模板：

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define MAXLEN 1000

// 1.定义二叉树
struct BinaryTreeNode{
    char data;  // 节点数据
    struct BinaryTreeNode *lson;    // 左子节点（指针）
    struct BinaryTreeNode *rson;    // 右子节点（指针）
};
typedef struct BinaryTreeNode BTNode;

// 2.二叉树创建函数
// 根据二叉树的先序遍历字符串st，构建一棵二叉树
// 其中，st表示先序遍历字符串，pos表示遍历字符串的下标
BTNode *createTree(char st[], int &pos)
{
    BTNode *node;
    char c;

    c = st[pos++];      // 每次取当前遍历的节点数据
    if (c == '0'){      // 如果数据为'0'，表示当前节点不存在
        ...             // 当前节点为空
    } else {
        ...             // 创建新的node节点，表示当前节点
        ...             // node节点数据为c
        ...             // 递归调用“创建函数”，node的左子节点为“创建函数”返回的节点
        ...             // 递归调用“创建函数”，node的右子节点为“创建函数”返回的节点
    }

    return node;        // 返回当前节点
}

// 3.释放函数（后序遍历可以参考此函数）
void freeTree(BTNode *node){
    if (node != NULL){          // 当前节点非空时才执行
        freeTree(node->lson);   // 递归调用“释放函数”，释放当前节点的左子树
        freeTree(node->rson);   // 递归调用“释放函数”，释放当前节点的右子树
        free(node);             // 释放当前节点内存
    }
}
```

```
// 4.前序遍历
void Pre(BTNode *node){
    if (node != NULL){          // 当前节点非空时才执行
        ...                     // 输出当前节点数据
        ...                     // 递归调用"前序遍历函数", 释放当前节点的左子树
        ...                     // 递归调用"前序遍历函数", 释放当前节点的右子树
    }
}

// 5.中序遍历
void In(BTNode *node){
    if (node != NULL){          // 当前节点非空时才执行
        ...                     // 递归调用"中序遍历函数", 释放当前节点的左子树
        ...                     // 输出当前节点数据
        ...                     // 递归调用"中序遍历函数", 释放当前节点的右子树
    }
}

// 6.后序遍历
void Pos(BTNode *node){
    if (node != NULL){          // 当前节点非空时才执行
        ...                     // 递归调用"后序遍历函数", 释放当前节点的左子树
        ...                     // 递归调用"后序遍历函数", 释放当前节点的右子树
        ...                     // 输出当前节点数据
    }
}

int main(){
    int T, pos;
    BTNode *tree;

    scanf("%d", &T);
    while (T--){
        char st[MAXLEN];
        scanf("%s", st);

        pos = 0;
        tree = createTree(st, pos);

        Pre(tree);
        printf("\n");

        In(tree);
        printf("\n");

        Pos(tree);
        printf("\n");

        freeTree(tree);
    }
    return 0;
}
```