递归树的深度 叶子结点个数和结点数总和

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

#define N 20


//二叉树结点的结构体表示形式
typedef struct BiTNode {
    char data;
    struct BiTNode *lchild;
    struct BiTNode *rchild;
} BiTNode, *BiTree;


void CreateBiTree(BiTree *T) {
    char ch;
    scanf("%c", &ch);
    if (ch == '#')
        *T = NULL;
    else {
        *T = (BiTree) malloc(sizeof(BiTNode));
        if (!*T)
            exit(-1);
        (*T)->data = ch;
        CreateBiTree(&(*T)->lchild);
        CreateBiTree(&(*T)->rchild);
    }
}


void PreOrder(BiTree T) {
    BiTNode **s;
    BiTNode *p;
    int top = -1;
    //创建栈;
    s = (BiTNode **) malloc((N + 1) * sizeof(BiTNode *));
    //初始化栈;
    s[++top] = T;
    //非递归前序遍历;
    while (top != -1) {
        p = s[top--];
        printf("%c ", p->data);     //栈的特点，先进后出;
        if (p->rchild)
            s[++top] = p->rchild;
        if (p->lchild)
            s[++top] = p->lchild;
    }
    free(s);
```

```
}

/*
 * 树的叶子节点和总节点数 求取
 */
void fun(BiTree T,int *levNum,int *sumNum){
    if (T == NULL){
        return;
    }

    (*sumNum)++;
    if (T->lchild == NULL && T->rchild == NULL){
        (*levNum)++;
    }

    fun(T->lchild,levNum,sumNum);
    fun(T->rchild,levNum,sumNum);

}

// 树深度
int TreeHeight(BiTree T){
    if (T == NULL){
        return 0;
    }
    int LD = TreeHeight(T->lchild);
    int RD = TreeHeight(T->rchild);

    return LD > RD ? LD + 1 : RD +1;

}

int main() {

    // ABDG##H###CE#I##F##
    printf("请以顺序输入二叉树(#表示该结点的子结点为空):\n");
    BiTree T;
    CreateBiTree(&T);
//    printf("前序遍历非递归实现: \n");
//    PreOrder(T);

    int levNum = 0,sumNode = 0;
    fun(T,&levNum,&sumNode);
    printf("%d  %d",levNum,sumNode);

    int h = TreeHeight(T);
    printf("\n%d",h);
    return 0;
}
```