```cpp
//日期：2018/ 时间：
/*
重建思路：        先序遍历的长度小于等于0，则返回。（preL>preR）
                  在中序遍历中，寻找那个k。将中序遍历序列分为左 右
                  在先序遍历中，寻找那个numLeft(左子树元素个数)。将先序序列（除去第一个元素）划分为左 右
*/
#include <stdio.h>
#include <stdlib.h>

struct node{
    int data;
    node* lchild;
    node* rchild;
    int layer;      //层次遍历时用到
};


//给定一颗二叉树的 先序遍历序列 和 中序遍历序列
//当前先序序列区间是[preL,preR],中序遍历区间是[inL,inR]
node* Create(int preL,int preR,int inL,int inR){
    if(preL > preR){
        return NULL;        //先序序列的长度小于等于0时，直接返回
    }
    node* root = new node;  //新建一个节点，用来存放当前二叉树的根节点
    root->data = pre[preL]; //新节点的数据域为根节点的值
    int k;
    for(k=inL;k<=inR;k++){
        if(in[k]==pre[preL]){   //在中序遍历序列中寻找in[k] == pre[preL]的节点
            break;
        }
    }
    int numLeft = k-inL;        //左子树的节点个数

    //则左子树：先序遍历区间是[preL+1,preL+numLeft] 中序遍历区间是[inL,k-1]
    //返回左子树的根节点指针，赋值给root的左指针
    root->lchild = create(preL+1,preL+numLeft,inL,k-1);

    //右子树：先序遍历区间是[preL+numLeft+1,preR] 中序遍历区间是[k+1,inR]
    //返回右子树的根节点指针，赋值给root的右指针
    root->rchild = create(preL+numLeft+1,preR,k+1,inR);

    return root;        //返回根节点
}

int main(){


    return 0;
}


```