题意

给定一棵多叉树，每个节点定义val,job ,指定val，初始job全为 -1
两种操作
C x： 查询结点val==x的job值
T x y 将以val == x为根节点的子树的所有节点的job值改为y

分析

这里着重分析如何将多叉树模型转换成线性模型，从而可以进行区间修改
从根节点开始，dfs整棵树，依次盖上时间戳，则对于结点x为根的子树，包含的时间戳范围就是：[刚访问x的时间戳 , 即将从x返回的时间戳]

代码

```
#include<iostream>
#include<cstring>
#include<vector>
#include<algorithm>
#define For(i,a,b) for(int i=(a); i<=(b) ; i++)
#define _For(i,a,b) for(int i=(a); i>=(b) ; i--)
#define Memset(a,b); memset((a),(b),sizeof((a)));
#define Cin(a); scanf("%d",&(a));
#define Cinc(a); scanf(" %c",&(a));
#define Cins(a); scanf("%s",(a));
#define Cout(a,b);  printf("%d",(a));printf(b);
#define Coutc(a,b);  printf("%c",(a));printf(b);
#define Couts(a,b);  printf("%s",(a));printf(b);
using namespace std;
typedef  long long LL;
typedef  unsigned long long ULL;
typedef  long double LDB;
inline int readint() {int x;cin>>x;return x;}
vector<int>v[50005];
int js,ans,n;
int st[50005];
int en[50005];
int lazy[500005];
int previsit = 0;
void dfs(int now)
```

```
{
    st[now] = ++js;
    for(int i=0;i<v[now].size();i++)
    {
        int to = v[now][i];
        if(i == v[now].size()-1) en[to] = n+1;
        else en[to] = v[now][i+1];
        dfs(to);
    }
    en[now] = js;
}
inline void pushdown(int o)
{
    lazy[o<<1] = lazy[o<<1|1] = lazy[o];
    lazy[o] = -1;
}
void update(int o,int l,int r,int L,int R,int d)
{
    if(l>=L && r<=R){
        lazy[o] = d;
        return;
    }
    if(lazy[o]!=-1) pushdown(o);
    int M = (l+r)>>1;
    if(M>=L) update(o<<1,l,M,L,R,d);
    if(M+1<=R) update(o<<1|1,M+1,r,L,R,d);
}
void query(int o,int l,int r,int x)
{
    if(l==r){
        ans = lazy[o];
        return;
    }
    if(lazy[o]!=-1) pushdown(o);
    int M = (l+r)>>1;
    if(M>=x) query(o<<1,l,M,x);
    else query(o<<1|1,M+1,r,x);
}
int main()
{
    int _,x,y;
    char cmd;
    Cin(_);
    For(T,1,_)
    {
        For(i,1,n) v[i].clear();
        Memset(st,0);
        Memset(en,0);
        Memset(lazy,-1);
        Cin(n);
        LL root = n*1LL*(n+1)/2*1LL;
        for(int i=1;i<n;i++)
        {
            Cin(x);Cin(y);
            root -= x;
            v[y].push_back(x);
        }
        js = 0;
        en[root] = n+1;
        dfs(root);
        int m;
        Cin(m);
        char cmd;
        printf("Case #%d:\n",T);
        st[n+1] = n+1;
        for(int i=1;i<=m;i++)
        {
            Cinc(cmd);
            if(cmd == 'C')
            {
                Cin(x);
                query(1,1,n,st[x]);
                Cout(ans,"\n");
            }
            else
            {
                Cin(x);Cin(y);
                int xx = st[x];
                int yy = en[x];
                update(1,1,n,xx,yy,y);
            }
        }

    }
}
```