```
struct link_cut_tree{
    struct node{
        node *p, *ch[2];
        bool rev;
        node(){
            p=ch[0]=ch[1]=NULL;
            rev=false;
        }
        bool d(){ return this==p->ch[1]; }
        bool isroot() { return !p || (this!=p->ch[0] && this!=p->ch[1]); }
        void relax(){
            if(rev){
                if(ch[0]) ch[0]->rev^=1;
                if(ch[1]) ch[1]->rev^=1;
                swap(ch[0],ch[1]);
                rev=false;
            }
        }
    };

    node *tree[8007];

    link_cut_tree(int n){
        f(i,1,n+1) tree[i]=new node();
    }

    link_cut_tree(){}

    void rotate(node *u){
        node *v=u->p;
        v->relax(), u->relax();
        bool d=u->d();
        if(!v->isroot()) v->p->ch[v->d()]=u;
        u->p=v->p;
        if(u->ch[!d]) u->ch[!d]->p=v;
        v->ch[d]=u->ch[!d];
        u->ch[!d]=v;
        v->p=u;
    }

    void splay(node *u){
        u->relax();
        while(!u->isroot()){
            if(u->p->isroot()) rotate(u);
            else (u->d()==u->p->d())? (rotate(u->p), rotate(u)):(rotate(u), rotate
(u));
        }
    }

    node *access(node *u){
        node *v;
        for(v=NULL; u; v=u, u=u->p){
            splay(u);
            u->ch[1]=v;
        }
        return v;
    }

    void make_root(node *u){
        access(u)->rev^=1;
        splay(u);
    }

    void link(node *u, node *v){
        make_root(v);
        v->p=u;
```

```
        access(v);
    }

    void cut(node *u, node *v){
        make_root(u);
        access(v);
        splay(v);
        u->p=v->ch[0]=NULL;
    }

    bool is_con(node *u, node *v){
        make_root(u);
        make_root(v);
        return !u->isroot();
    }

    void link(int u, int v){
        link(tree[u],tree[v]);
    }

    void cut(int u, int v){
        cut(tree[u],tree[v]);
    }

    bool is_con(int u, int v){
        return is_con(tree[u],tree[v]);
    }
};
```