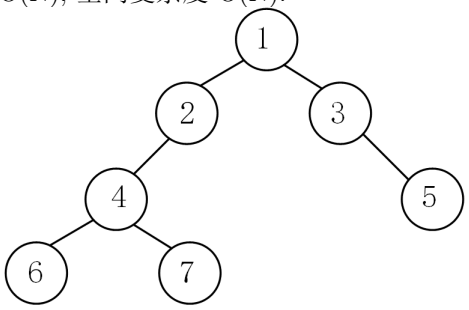
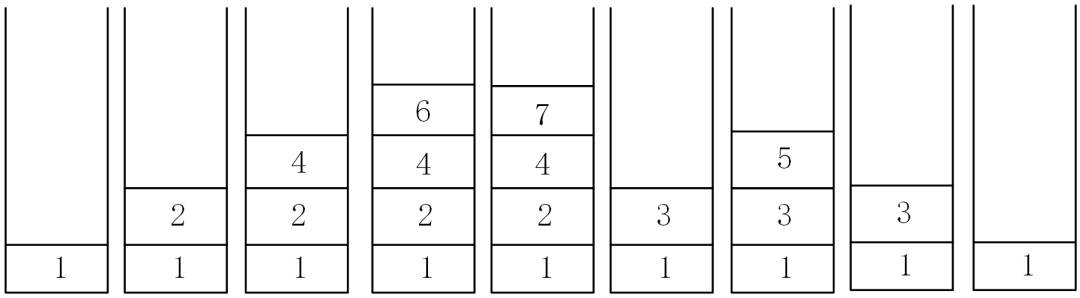
6-37,38

题目要求：用栈的基本操作写出先序遍历，后序遍历的非递归算法。

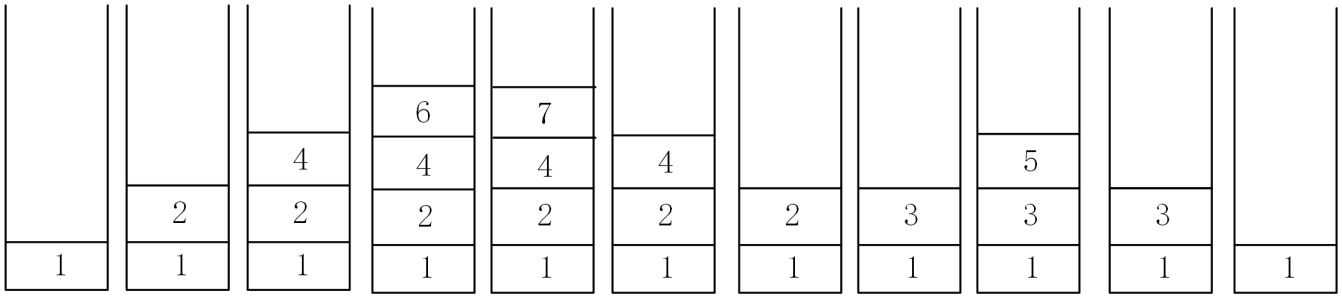
算法分析：先用数组存储树的结构，当栈不为空则一直进行操作，用 vis 进行标记某一节点是否遍历过，如果当前栈顶（根）节点没有左右儿子则弹出栈 pop，否则按照左右顺序依次入栈 push。

后序遍历类似，但先是左右儿子入栈，不标记当前节点遍历，当左右儿子都弹出 pop 后标记当前 top 节点并输出。时间复杂度 O(N), 空间复杂度 O(N).





先序：进栈遍历



后序：出栈遍历

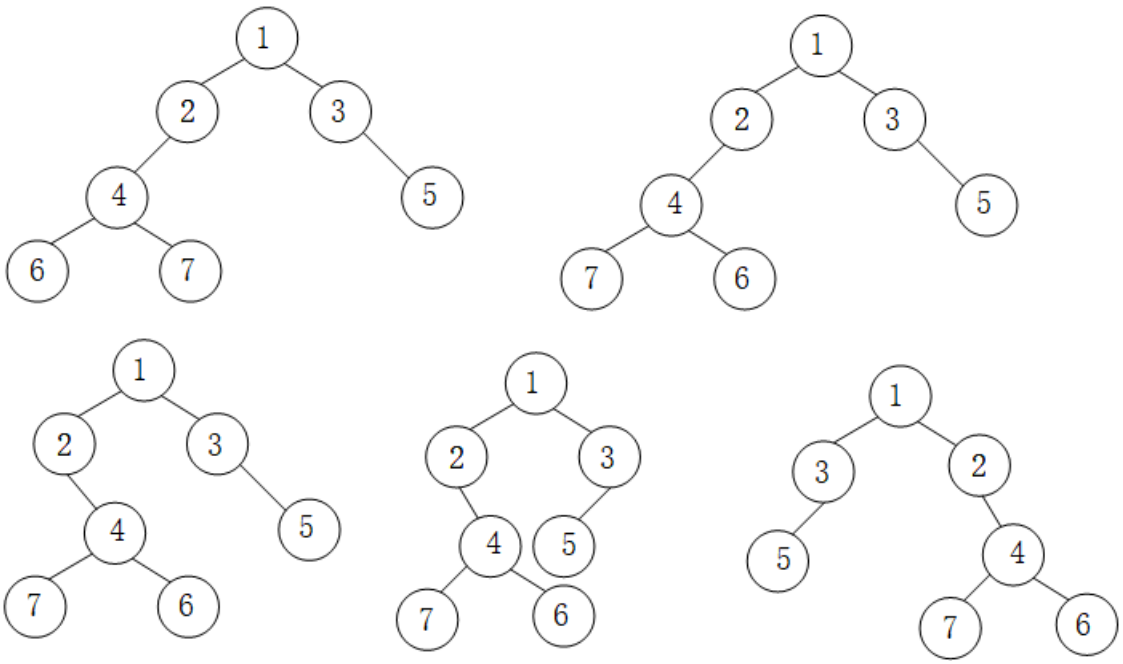
code

```
#include<bits/stdc++.h>
#define maxn 1000
using namespace std;
int n;
struct node
{
    int num;
    int lson;
    int rson;
    bool vis;
    bool pri;
}tree[maxn];
stack<node> s;
void pre_order()//先序遍历
{
    while(!s.empty())//栈不为空则运行
    {
```

```
18          if (!tree[s.top().num].vis)
19          {
20             cout<<s.top().num;
21             tree[s.top().num].vis=true;
22          }
23          if  (s.top().lson==-1&&s.top().rson==-1)  s.pop();
24          if (s.top().lson!=-1&&!tree[s.top().lson].vis) {//tree[s.top().lson].vis=true;
                                                                    s.push(tree[s.top().lson]);
              continue;}
25          if (s.top().rson!=-1&&!tree[s.top().rson].vis) {//tree[s.top().rson].vis=true;
26                                                  s.push(tree[s.top().rson]);continue;}
27          s.pop();
28       }
29    }
30    void mid_order()//中序遍历
31    {
32
33       while(!s.empty())
34       {
35          if (s.top().lson==-1&&s.top().rson==-1)  {tree[s.top().num].vis=true;cout<<s.top().
              num;s.pop();}
36          if (s.top().lson!=-1&&!tree[s.top().lson].vis) {s.push(tree[s.top().lson]);continue;}
37          if (!tree[s.top().num].vis)
38             {
39              cout<<s.top().num;
40              tree[s.top().num].vis=true;
41             }
42          if (s.top().rson!=-1&&!tree[s.top().rson].vis) {s.push(tree[s.top().rson]);continue;}
43          s.pop();
44       }
45    }
46    void last_order()//后序遍历
47    {
48
49       while(!s.empty())
50       {
51          if (s.top().lson==-1&&s.top().rson==-1)  {tree[s.top().num].vis=true;cout<<s.top().
              num;s.pop();}
52          if (s.top().lson!=-1&&!tree[s.top().lson].vis) {tree[s.top().lson].vis=true;s.push(
              tree[s.top().lson]);continue;}
53          if (s.top().rson!=-1&&!tree[s.top().rson].vis) {tree[s.top().rson].vis=true;s.push(
              tree[s.top().rson]);continue;}
54          cout<<s.top().num;
55          s.pop();
56       }
57    }
58
59    int main()
60    {
61       int F,S,root;
62       cin>>n;
63       for (int i=1;i<maxn;i++)
```

```
64      {
65          tree[i].lson=-1;
66          tree[i].rson=-1;
67          tree[i].vis=false;
68      }
69      for (int i=1;i<=n;i++)
70      {
71          int s_size;
72
73          scanf("%d%d%d",&s_size,&F,&S);
74          if (s_size==1)
75              tree[F].lson=S;
76          else tree[F].rson=S;
77          tree[F].num=F;
78          tree[S].num=S;
79      }
80      cin>>root;
81      while(!s.empty()) s.pop();
82      s.push(tree[root]);
83
84      cout<<"pre-order:";
85      pre_order();
86      cout<<endl;
87
88      for (int i=1;i<maxn;i++)
89          tree[i].vis=false;
90      tree[root].vis=true;
91      s.push(tree[root]);
92
93      cout<<"last-order:";
94      last_order();
95      cout<<endl;
96
97      for (int i=1;i<maxn;i++)
98      tree[i].vis=false;
99      s.push(tree[root]);
100     cout<<"mid-order:";
101     mid_order();
102 }
```

6-43 题目要求：编写递归算法，将二叉树的所有结点的左右子树互相交换。
算法分析：用数组结构体的形式表示二叉树，先根据输入信息建立数据关系，再递归从根节点到叶子节点进行翻转。
时间复杂度 O(N), 空间复杂度 O(N).



Code

```
#include<bits/stdc++.h>
using namespace std;
#define maxn 1000
#define inf 0x3f3f3f3f
int n,root,Mnode=-inf;
struct innode//树结点
{
    char name;
    innode* l;//结点左儿子
    innode* r;//结点右儿子
}A[maxn],*p=A;

void rev(innode x)//翻转左右儿子
{
    *innode pn;
    if (x.l!=NULL)  rev(*(x.l));
    if (x.r!=NULL)  rev(*(x.r));
    pn=x.l;
    x.l=x.r;
    x.r=pn;
}
int main()
{
    cin>>n;
    cin>>root;
    int a,ls,rs;
    char c;
    for (int i=0;i<maxn;i++)
        {
            A[i].l=NULL;
            A[i].r=NULL;
        }
```

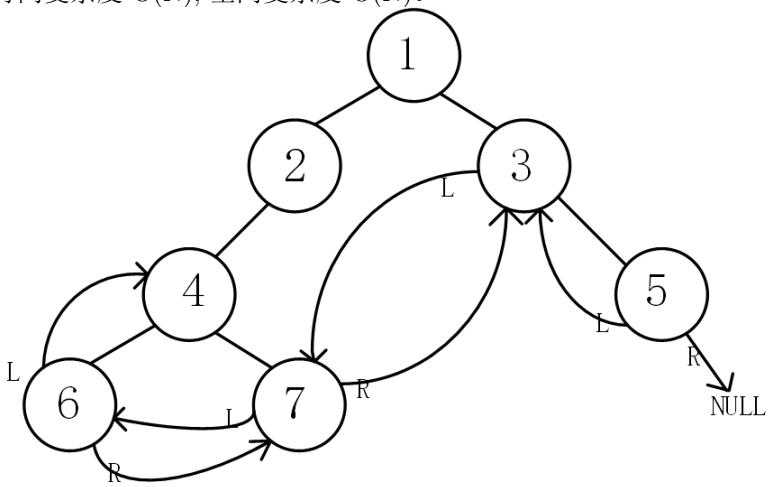
```
    for (int i=1;i<=n;i++)
    {
        scanf("%d %d %d %c",&a,&ls,&rs,&c);//输入每个结点的父亲信息和该点的标号
        A[a].name=c;
        A[a].l=(p+ls);
        A[a].r=(p+rs);
    }
    rev(root);
 return 0;
}
```

6-56

题目要求: 试写一个算法，再先序后继线索二叉树中，查找给定结点 *p 在先序序列中的后继。

算法分析：建立线索二叉树，若该节点没有右子树，则后继为 p->rson. 存在右子树，不存在左子树，则为 p-rson。若存在左子树不存在右子树，则为 p->lson。

时间复杂度 O(N), 空间复杂度 O(N)。



Code

```
#include<bits/stdc++.h>
using namespace std;
#define maxn 1000
#define inf 0x3f3f3f3f
int n,root;
struct node//树结点
{
    char name;
    node* ls;
    node* rs;
    int llog;
    int rlog;
    int num;
};
char A[maxn]={0};
int n;
node *heap=(node *)malloc(sizeof(node));//根节点
void build(node * x)//建立一棵树
{

```

```
21          node *p1,*p2;//左右子树指针
22          if (A[2*x->num])
23          {
24              x->llog=1;
25
26              p1=(node*)malloc(sizeof(node));
27              x->ls=p1;
28              p1->ls=p1->rs=NULL;
29              p1->name=A[2*x->num];
30              p1->llog=p1->rlog=0;
31              p1->num=2*x->num;
32              build(p1);//左子树
33          }
34
35          if (A[(2*x->num)+1])
36          {
37              x->rlog=1;
38              p2=(node*)malloc(sizeof(node));
39              x->rs=p2;
40              p2->ls=p2->rs=NULL;
41              p2->name=A[(2*x->num)+1];
42              p2->llog=p2->rlog=0;
43              p2->num=(2*x->num)+1;
44              build(p2);//右子树
45          }
46      }
47      node *P1,*P2;
48      void oline(node * pf)//先序遍历将其线索化
49      {
50          P1=P2;
51          P2=pf;
52          if (P1->rs==NULL) P1->rs=pf;
53          if (pf->ls==NULL) pf->ls=P1;
54          if (pf->ls!=NULL) oline(pf->ls);
55          if (pf->rs!=NULL) oline(pf->rs);
56      }
57      int main()
58      {
59          cin>>n;
60          for (int i=0;i<maxn;i++)
61              p[i]=NULL;
62          for (int i=1;i<n;i++)
63              cin>>A[i];//输入树节点信息
64          heap->name=A[1];
65          heap->llog=0;
66          heap->rlog=0;//根初始化条件
67          heap->num=1;
68          build(heap);
69          oline(heap);
70          return 0;
71      }
```