#include<iostream>

#include<string.h>

using namespace std;

const int Size=100;

class node

{

public:

node\*rc,\*lc;int data;

node(int k)

{

rc=NULL;

lc=NULL;

data=k;

}

};

class stk

{

public:

node\*arr[Size];int top;

stk()

{

top=-1;

}

void push(node\*ele)

{

if(top!=Size-1)

{

arr[++top]=ele;

}

else

{

cout<<"stk is full"<<endl;

return;

}

}

node\*pop()

{

if(top==-1){return NULL;}

return arr[top--];

}

};

class Tree

{

public:

node\*root;

Tree(){root=NULL;}

node\*createiterative(char exp[],int k)

{

node\*temp=new node(k);

int n=strlen(exp);

if(root==NULL)

{

root=temp;

return root;

}

node\*t=root;int i;node\*prev=NULL;

for(i=0;i<n;i++)

{

if(t==NULL)

{

break;

}

prev=t;

if(exp[i]=='L'){t=t->lc;}

else {t=t->rc;}

}

if(t!=NULL||i!=n)

{ cout<<"not possible"<<endl;

delete(temp);

return root;

}

if(exp[i-1]=='L'){prev->lc=temp;}

else {prev->rc=temp;}

return root;

}

void display(node\*root,int level)

{

int i;

if(root==NULL){return;}

display(root->rc,level+1);

for(int i=0;i<level;i++)

{

cout<<" ";

}

cout<<root->data<<endl;

display(root->lc,level+1);

return;

}

node\*createrecursive(int k)

{

if(k==-1)

{

return NULL;

}

node\*t=new node(k);

cout<<"enter left child of"<<k<<endl;

int ln;cin>>ln;

t->lc=createrecursive(ln);

cout<<"enter right child of"<<k<<endl;

int rn;cin>>rn;

t->rc=createrecursive(rn);

return t;

}

void inorder(node\*root)

{

if(root==NULL)

{

return;

}

inorder(root->lc);

cout<<root->data<<" ";

inorder(root->rc);

}

void postorder(node\*root)

{

if(root==NULL)

{

return;

}

postorder(root->lc);

postorder(root->rc);

cout<<root->data<<" ";

}

void preorder(node\*root)

{

if(root==NULL)

{

return;

}

cout<<root->data<<" ";

preorder(root->lc);

preorder(root->rc);

}

int depth(node\*root)

{

}

};

int main()

{

Tree obj;

int n;

cin>>n;

char exp[100];

while(n--)

{

cout<<"enter the exp"<<endl;

cin>>exp;

cout<<"enter the data"<<endl;

int k;

cin>>k;

obj.createiterative(exp,k);

}

obj.root=obj.createrecursive(5);

obj.inorder(obj.root);

cout<<"2:inorder traversal"<<endl;

cout<<"3:preorder"<<endl;

cout<<"4:postorder"<<endl;

cout<<"5:print the parent"<<endl;

cout<<"6:print the depth"<<endl;

cout<<"7:print the ancestors"<<endl;

}