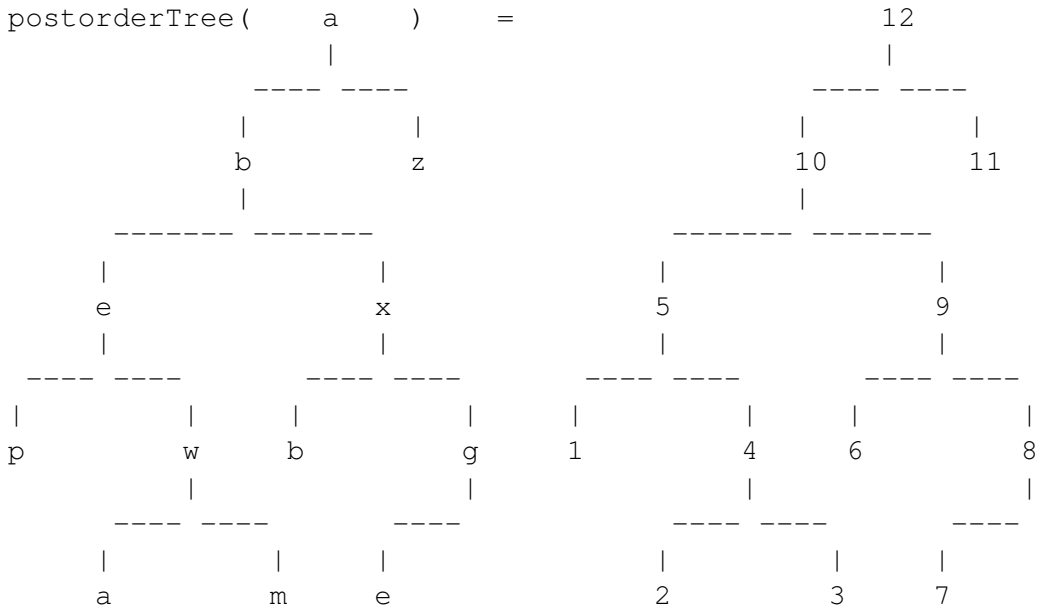
Construeix arbres i accedeix a posicions en postordre X73395_ca

Preliminars:
Preliminars: Recordeu que el recorregut en postordre d'un arbre és la llista dels nodes de l'arbre ordenada com segueix: en primer lloc, el recorregut en postordre del fill esquerra de l'arbre, després el recorregut en postordre del fill dret de l'arbre, i finalment l'arrel de l'arbre. En altres paraules:

- $Postordre(x(t_1, t_2)) = Postordre(t_1) \cdot Postordre(t_2) \cdot x$
- $Postordre(()) = ()$, és a dir, el postordre de l'arbre buit és l'arbre buit.

Donat un arbre de caràcters, el seu corresponent arbre de posicions en postordre és un arbre d'enters amb exactament la mateixa estructura (conjunt de posicions), i a on cada node guardarà la posició d'aquell node en el recorregut en postordre.



En l'exemple anterior, fixeu-vos que el valor guardat en l'arbre original a posició en postordre 1 és 'p', el valor guardat a posició en postordre 8 és 'g', i el valor guardat a posició en postordre 12 és 'a'.
Fi de preliminars
En aquest exercici, heu d'implementar un programa que llegeix comandes que manipulen variables que guarden àrbres binaris de caràcters. La primera comanda `numvars=` n `;` indica el nombre total n de variables. Els noms d'aquestes variables son `t0`, ..., `t(n-1)`, i se suposa que inicialment cadascuna guarda un àrbre buit. Després venen comandes que construeixen nous àrbres a partir de variables i els assignen a variables (com per exemple `t2 =BinTree( a , t0 , t1 );`, i comandes que accedeixen als fills d'un arbre existent i els assignen a variables (com per exemple `t3 = t2 .left();` o `t3 = t2 .right();`). També hi ha comandes per a escriure per la sortida un àrbre en `INLINEFORMAT` (com per exemple `cout<< t2 <<endl;`), i comandes per a escriure el valor guardat en un arbre a una posició en postordre donada (com per exemple `cout<<valueAtPostorderPosition( t2 , 3 )<<endl;`

Aquest és un exemple d'entrada del programa:

```
numvars= 4 ;
t1 =BinTree( a , t2 , t3 );
t2 =BinTree( b , t1 , t3 );
t3 =BinTree( c , t2 , t1 );
cout<< t0 <<endl;
cout<< t1 <<endl;
cout<< t2 <<endl;
cout<< t3 <<endl;
cout<<valueAtPostorderPosition( t1 , 1 )<<endl;
cout<<valueAtPostorderPosition( t2 , 1 )<<endl;
cout<<valueAtPostorderPosition( t2 , 2 )<<endl;
cout<<valueAtPostorderPosition( t3 , 1 )<<endl;
cout<<valueAtPostorderPosition( t3 , 2 )<<endl;
cout<<valueAtPostorderPosition( t3 , 3 )<<endl;
cout<<valueAtPostorderPosition( t3 , 4 )<<endl;
t1 =BinTree( d , t2 , t3 );
t2 =BinTree( e , t1 , t3 );
t3 =BinTree( f , t2 , t1 );
cout<< t1 <<endl;
cout<< t2 <<endl;
cout<< t3 <<endl;
cout<<valueAtPostorderPosition( t1 , 1 )<<endl;
cout<<valueAtPostorderPosition( t1 , 2 )<<endl;
cout<<valueAtPostorderPosition( t1 , 3 )<<endl;
cout<<valueAtPostorderPosition( t1 , 4 )<<endl;
cout<<valueAtPostorderPosition( t2 , 3 )<<endl;
cout<<valueAtPostorderPosition( t2 , 4 )<<endl;
cout<<valueAtPostorderPosition( t2 , 8 )<<endl;
cout<<valueAtPostorderPosition( t2 , 9 )<<endl;
cout<<valueAtPostorderPosition( t3 , 8 )<<endl;
cout<<valueAtPostorderPosition( t3 , 13 )<<endl;
cout<<valueAtPostorderPosition( t3 , 16 )<<endl;
cout<<valueAtPostorderPosition( t3 , 18 )<<endl;
t1 = t3 .left();
t2 = t1 .right();
t3 = t2 .left();
cout<< t1 <<endl;
cout<< t2 <<endl;
cout<< t3 <<endl;
cout<<valueAtPostorderPosition( t1 , 3 )<<endl;
cout<<valueAtPostorderPosition( t1 , 4 )<<endl;
cout<<valueAtPostorderPosition( t1 , 8 )<<endl;
cout<<valueAtPostorderPosition( t1 , 9 )<<endl;
cout<<valueAtPostorderPosition( t2 , 1 )<<endl;
cout<<valueAtPostorderPosition( t2 , 2 )<<endl;
cout<<valueAtPostorderPosition( t2 , 3 )<<endl;
cout<<valueAtPostorderPosition( t2 , 4 )<<endl;
cout<<valueAtPostorderPosition( t3 , 1 )<<endl;
```

```
cout<<valueAtPostorderPosition( t3 , 2 )<<endl;
```

La sortida del programa amb la seqüència de comandes d'entrada anterior hauria de ser:

```
()
a
b(a,)
c(b(a,),a)
a
a
b
a
b
a
c
d(b(a,),c(b(a,),a))
e(d(b(a,),c(b(a,),a)),c(b(a,),a))
f(e(d(b(a,),c(b(a,),a)),c(b(a,),a)),d(b(a,),c(b(a,),a)))
a
b
a
b
a
c
d
a
b
c
d
c
e
a
b
c
d
c
e
c
d
f
e(d(b(a,),c(b(a,),a)),c(b(a,),a))
c(b(a,),a)
b(a,)
a
b
c
d
c
e
a
```

```
b
a
c
a
b
```

Com podeu observar a l'exemple d'entrada anterior, hi han espais en blanc per a facilitar la lectura.

GUIA PER A OBTENIR UNA SOLUCIÓ INEFICIENT QUE SUPERI ELS JOCS DE PROVES PÚBLICS

A continuació us posem una guia per a obtenir una solució lenta. Aquesta us permetrà superar els jocs de proves públics però no els privats, obtenint així la meitat de la nota. Per tal d'obtenir una solució ràpida, haureu de repensar el programa, quines dades convé mantenir, i com fer-les servir.

Per a obtenir una solució lenta, n'hi ha prou amb guardar un vector de BinTree t[0...numvars-1] sobre el qual es guarden els àrbres que es van calculant. Totes les comandes es transformen en crides directes a mètodes de BinTree excepte `valueAtPostorderPosition`, per a la cual caldrà implementar una funció, amb el mateix nom, que rebi un arbre i un enter que representa una posició en postordre, i retorni el valor de l'arbre en aquella posició.

Podeu utilitzar la plantilla següent, a on només hi manca implementar la funció `getValueAtPostorderPos`

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <vector>
// Add more includes if you wish ...

using namespace std;

#include "BinTree.hh"

typedef BinTree<char> BT;

int getIdVar(string s)
{
    return atoi(s.substr(1).c_str());
}

// Add auxiliary functions if you wish ...

char getValueAtPostorderPosition(BT t, int pos)
{
    // Implement this function ...
}

int main()
{
    string s1, s2, s3, s4, s5, s6, s7;
    int numvars;
    cin >> s1 >> numvars >> s2;
```

```
    vector<BT> t(numvars);
    while (cin >> s1 >> s2) {
        if (s1[0] == 't') {
            int idvar = getIdVar(s1);
            if (s2 == "=BinTree(") {
                char value;
                cin >> value >> s3 >> s4 >> s5 >> s6 >> s7;
                int idvar1 = getIdVar(s4);
                int idvar2 = getIdVar(s6);
                t[idvar] = BT(value, t[idvar1], t[idvar2]);
            } else if (s2 == "=") {
                cin >> s3 >> s4;
                int idvar1 = getIdVar(s3);
                if (s4 == ".left();") {
                    t[idvar] = t[idvar1].left();
                } else {
                    t[idvar] = t[idvar1].right();
                }
            }
        } else if (s1 == "cout<<") {
            int idvar = getIdVar(s2);
            cin >> s3;
            t[idvar].setOutputFormat(BinTree<int>::INLINEFORMAT);
            cout << t[idvar] << endl;
        } else if (s1 == "cout<<valueAtPostorderPosition(") {
            int idvar = getIdVar(s2);
            int pos;
            cin >> s3 >> pos >> s4;
            cout << getValueAtPostorderPosition(t[idvar], pos) << endl;
        } else {
            cout << "Error: unexpected command '" << s1 << "'" << endl;
            exit(1);
        }
    }
}
```

Fixeu-vos que l'enunciat d'aquest exercici us ofereix el fitxer `BinTree.hh`. Us falta crear el fitxer `main.cc`, que podeu construïr, si voleu, a partir de la plantilla que us hem oferit abans. Només cal que pugeu `main.cc` al jutge.

Observació: Com us hem mencionat abans, la guia y el programa que us oferim com a plantilla indica com obtenir una solució lenta. Els arbres dels jocs de proves privats son molt grans, i això fa que, encara que implementeu molt bé `valueAtPostorderPosition`**, s'obtingui temps límit excedit. Tot i així, els arbres dels jocs de proves privats tenen poca profunditat (doncs son bastant equilibrats). Per tant, cal repensar el programa i seguir un enfocament que faci que resoldre les comandes** `valueAtPostorderPosition` **tingui cost proporcional a com a molt la profunditat de l'arbre.**

Entrada

La primera linia de l'entrada és de la forma `numvars= LIMIT ;`, a on LIMIT és un nombre natural positiu. Després venen instruccions d'aquestes menes:

```
tNUM =BinTree( VALUE , tNUM1 , tNUM2 );
tNUM1 = tNUM2 .left();
tNUM1 = tNUM2 .right();
cout<< tNUM <<endl;
cout<<valueAtPostorderPosition( tNUM , POSTORDERINDEX )<<endl;
```

On `VALUE` es una lletra minúscula, `NUM, NUM1, NUM2` son naturals en el rang {0,...,LIMIT-1}, i POSTORDERINDEX és un natural entre 1 i el nombre de nodes de l'arbre guardat en la variable que l'acompanya en la crida a `valueAtPostorderPosition`.
Se suposa que les entrades son correctes. En particular, sempre es demana accedir a `left` o `right` d'arbres no buits.

Sortida

Per a cada instrucció dels següents dos tipus, el vostre programa ha d'escriure el resultat esperat (l'arbre contingut en la variable en `INLINEFORMAT`, o el valor guardat per l'arbre contingut en la variable en la posició indicada per l'índex en postordre, segons el cas).

```
cout<< tNUM <<endl;
cout<<valueAtPostorderPosition( tNUM , POSTORDERINDEX )<<endl;
```

Exemple d'entrada 1

```
numvars= 4 ;
t1 =BinTree( a , t2 , t3 );
t2 =BinTree( b , t1 , t3 );
t3 =BinTree( c , t2 , t1 );
cout<< t0 <<endl;
cout<< t1 <<endl;
cout<< t2 <<endl;
cout<< t3 <<endl;
cout<<valueAtPostorderPosition( t1 , 1 )<<endl;
cout<<valueAtPostorderPosition( t2 , 1 )<<endl;
cout<<valueAtPostorderPosition( t2 , 2 )<<endl;
cout<<valueAtPostorderPosition( t3 , 1 )<<endl;
cout<<valueAtPostorderPosition( t3 , 2 )<<endl;
cout<<valueAtPostorderPosition( t3 , 3 )<<endl;
cout<<valueAtPostorderPosition( t3 , 4 )<<endl;
t1 =BinTree( d , t2 , t3 );
t2 =BinTree( e , t1 , t3 );
t3 =BinTree( f , t2 , t1 );
cout<< t1 <<endl;
cout<< t2 <<endl;
cout<< t3 <<endl;
cout<<valueAtPostorderPosition( t1 , 1 )<<endl;
cout<<valueAtPostorderPosition( t1 , 2 )<<endl;
cout<<valueAtPostorderPosition( t1 , 3 )<<endl;
cout<<valueAtPostorderPosition( t1 , 4 )<<endl;
cout<<valueAtPostorderPosition( t1 , 5 )<<endl;
cout<<valueAtPostorderPosition( t1 , 6 )<<endl;
cout<<valueAtPostorderPosition( t1 , 7 )<<endl;
cout<<valueAtPostorderPosition( t2 , 3 )<<endl;
cout<<valueAtPostorderPosition( t2 , 4 )<<endl;
cout<<valueAtPostorderPosition( t2 , 6 )<<endl;
cout<<valueAtPostorderPosition( t2 , 7 )<<endl;
cout<<valueAtPostorderPosition( t2 , 11 )<<endl;
cout<<valueAtPostorderPosition( t2 , 12 )<<endl;
cout<<valueAtPostorderPosition( t3 , 3 )<<endl;
cout<<valueAtPostorderPosition( t3 , 4 )<<endl;
cout<<valueAtPostorderPosition( t3 , 6 )<<endl;
cout<<valueAtPostorderPosition( t3 , 7 )<<endl;
cout<<valueAtPostorderPosition( t3 , 11 )<<endl;
cout<<valueAtPostorderPosition( t3 , 12 )<<endl;
cout<<valueAtPostorderPosition( t3 , 18 )<<endl;
cout<<valueAtPostorderPosition( t3 , 19 )<<endl;
cout<<valueAtPostorderPosition( t3 , 20 )<<endl;
t1 =t3 .left();
t2 =t1 .right();
t3 =t2 .left();
cout<< t1 <<endl;
cout<< t2 <<endl;
cout<< t3 <<endl;
cout<<valueAtPostorderPosition( t1 , 3 )<<endl;
cout<<valueAtPostorderPosition( t1 , 4 )<<endl;
cout<<valueAtPostorderPosition( t1 , 6 )<<endl;
cout<<valueAtPostorderPosition( t1 , 7 )<<endl;
cout<<valueAtPostorderPosition( t1 , 11 )<<endl;
cout<<valueAtPostorderPosition( t1 , 12 )<<endl;
cout<<valueAtPostorderPosition( t2 , 1 )<<endl;
cout<<valueAtPostorderPosition( t2 , 2 )<<endl;
cout<<valueAtPostorderPosition( t2 , 3 )<<endl;
cout<<valueAtPostorderPosition( t2 , 4 )<<endl;
cout<<valueAtPostorderPosition( t3 , 1 )<<endl;
```

```
cout<<valueAtPostorderPosition( t3 , 2 )<<endl;
```

Exemple de sortida 1

```
()
a
b(a,)
c(b(a,),a)
a
a
b
a
b
a
c
d(b(a,),c(b(a,),a))
e(d(b(a,),c(b(a,),a)),c(b(a,),a))
f(e(d(b(a,),c(b(a,),a)),c(b(a,),a)),d(b(a,),c(b(a,),a))
a
b
a
b
a
c
d
a
b
c
d
c
e
a
b
c
d
c
e
c
d
f
e(d(b(a,),c(b(a,),a)),c(b(a,),a))
c(b(a,),a)
b(a,)
a
b
c
d
c
e
a
b
a
c
a
b
```

Exemple d'entrada 2

```
numvars= 5 ;
cout<< t1 <<endl;
cout<< t0 <<endl;
t1 =BinTree( b , t2 , t4 );
t2 =BinTree( c , t0 , t4 );
cout<<valueAtPostorderPosition( t1 , 1 )<<endl;
t1 = t2 .left();
t3 =BinTree( h , t2 , t4 );
t2 =BinTree( s , t3 , t2 );
t4 = t2 .right();
t3 = t4 .right();
t4 =BinTree( y , t3 , t1 );
```

```
t2 =BinTree( e , t3 , t4 );
t1 =BinTree( e , t3 , t2 );
cout<< t1 <<endl;
t2 =BinTree( k , t1 , t0 );
t4 =BinTree( f , t4 , t3 );
t3 =BinTree( c , t4 , t3 );
t4 = t4 .right();
cout<< t0 <<endl;
t1 =BinTree( q , t4 , t4 );
cout<<valueAtPostorderPosition( t3 , 3 )<<endl;
cout<< t1 <<endl;
t0 =BinTree( d , t1 , t1 );
cout<< t4 <<endl;
t1 = t2 .right();
cout<< t1 <<endl;
cout<< t1 <<endl;
t4 =BinTree( j , t3 , t1 );
cout<< t4 <<endl;
t4 = t0 .right();
cout<< t3 <<endl;
cout<<valueAtPostorderPosition( t0 , 3 )<<endl;
t0 =BinTree( u , t1 , t1 );
cout<<valueAtPostorderPosition( t0 , 1 )<<endl;
t3 = t4 .right();
t4 = t4 .left();
cout<< t4 <<endl;
t2 =BinTree( x , t0 , t3 );
t3 =BinTree( i , t1 , t3 );
t4 =BinTree( a , t3 , t3 );
cout<< t3 <<endl;
t4 =BinTree( c , t3 , t3 );
t2 =BinTree( n , t2 , t1 );
cout<<valueAtPostorderPosition( t3 , 1 )<<endl;
cout<<valueAtPostorderPosition( t3 , 1 )<<endl;
t2 =BinTree( d , t0 , t4 );
cout<< t0 <<endl;
t1 =BinTree( i , t4 , t2 );
t4 = t2 .right();
t1 =BinTree( d , t0 , t3 );
t1 =BinTree( w , t4 , t1 );
t3 =BinTree( p , t1 , t2 );
t2 =BinTree( s , t2 , t4 );
cout<<valueAtPostorderPosition( t1 , 7 )<<endl;
t1 =BinTree( n , t1 , t0 );
t4 = t0 .left();
cout<< t1 <<endl;
cout<< t4 <<endl;
cout<<valueAtPostorderPosition( t2 , 9 )<<endl;
t2 =BinTree( k , t4 , t4 );
cout<<valueAtPostorderPosition( t3 , 13 )<<endl;
cout<< t1 <<endl;
t1 = t0 .left();
t3 =BinTree( d , t2 , t0 );
t3 = t0 .right();
cout<<valueAtPostorderPosition( t2 , 1 )<<endl;
t2 =BinTree( u , t2 , t2 );
cout<< t0 <<endl;
t4 =BinTree( u , t0 , t2 );
cout<< t3 <<endl;
t0 = t4 .right();
cout<< t0 <<endl;
cout<< t4 <<endl;
t3 =BinTree( x , t1 , t0 );
t2 =BinTree( i , t4 , t4 );
cout<< t0 <<endl;
cout<<valueAtPostorderPosition( t2 , 11 )<<endl;
t1 =BinTree( h , t4 , t1 );
t4 = t2 .left();
cout<<valueAtPostorderPosition( t1 , 6 )<<endl;
cout<< t0 <<endl;
cout<< t1 <<endl;
cout<< t2 <<endl;
cout<< t3 <<endl;
cout<< t4 <<endl;
```

Exemple de sortida 2

```
()
()
b
e(,e(,y))
()
c
q
()
()
()
j(c(f(y,),),)
c(f(y,),)
d
u
()
i
i
i
u
w
n(w(c(i,i),d(u,i)),u)
()
s
p
n(w(c(i,i),d(u,i)),u)
k
u
()
u(k,k)
u(u,u(k,k))
u(k,k)
i
h
u(k,k)
h(u(u,u(k,k)),)
i(u(u,u(k,k)),u(u,u(k,k)))
x(,u(k,k))
u(u,u(k,k))
```

Exemple d'entrada 3

```
numvars= 10 ;
t7 =BinTree( w , t5 , t3 );
cout<< t6 <<endl;
t1 =BinTree( b , t2 , t7 );
t3 =BinTree( r , t6 , t0 );
cout<< t2 <<endl;
cout<< t1 <<endl;
t2 =BinTree( s , t3 , t7 );
cout<< t9 <<endl;
t8 =BinTree( o , t9 , t7 );
t1 =BinTree( s , t2 , t9 );
t9 =BinTree( d , t4 , t7 );
t6 =BinTree( b , t1 , t0 );
cout<< t3 <<endl;
t6 =BinTree( g , t1 , t5 );
cout<< t4 <<endl;
cout<<valueAtPostorderPosition( t6 , 5 )<<endl;
cout<< t6 <<endl;
t3 = t7 .right();
t2 =BinTree( p , t5 , t4 );
t0 =BinTree( h , t7 , t8 );
cout<< t8 <<endl;
t9 =BinTree( w , t2 , t0 );
cout<< t8 <<endl;
t2 = t6 .right();
cout<< t4 <<endl;
t5 = t0 .right();
t7 =BinTree( c , t1 , t7 );
t2 =BinTree( v , t2 , t6 );
t6 =BinTree( k , t1 , t5 );
t4 = t9 .right();
t1 =BinTree( v , t7 , t7 );
t5 =BinTree( l , t9 , t7 );
cout<<valueAtPostorderPosition( t6 , 7 )<<endl;
cout<< t5 <<endl;
cout<< t3 <<endl;
t4 = t8 .right();
t9 =BinTree( a , t3 , t9 );
t5 = t6 .left();
t1 =BinTree( q , t0 , t3 );
t4 =BinTree( c , t4 , t4 );
cout<<valueAtPostorderPosition( t6 , 7 )<<endl;
cout<< t2 <<endl;
t8 =BinTree( n , t5 , t7 );
cout<<valueAtPostorderPosition( t5 , 4 )<<endl;
t8 =BinTree( k , t3 , t1 );
t9 = t6 .left();
cout<< t0 <<endl;
cout<<valueAtPostorderPosition( t7 , 6 )<<endl;
cout<< t4 <<endl;
t3 =BinTree( g , t0 , t9 );
t4 =BinTree( f , t0 , t5 );
t4 = t6 .right();
t2 = t2 .right();
t7 =BinTree( t , t5 , t4 );
t1 = t2 .left();
t9 = t3 .left();
cout<< t8 <<endl;
t1 =BinTree( o , t0 , t5 );
t4 =BinTree( c , t6 , t2 );
cout<< t8 <<endl;
cout<< t2 <<endl;
t4 = t7 .left();
t0 =BinTree( a , t6 , t2 );
t9 = t0 .right();
t1 = t3 .left();
t0 =BinTree( l , t3 , t4 );
t9 =BinTree( p , t1 , t9 );
cout<< t9 <<endl;
cout<< t6 <<endl;
t0 =BinTree( u , t4 , t6 );
t7 =BinTree( c , t5 , t6 );
t8 = t7 .right();
cout<< t0 <<endl;
t6 =BinTree( h , t1 , t3 );
cout<<valueAtPostorderPosition( t5 , 4 )<<endl;
t7 = t0 .left();
```

```
t0 =BinTree( w , t4 , t2 );
t6 = t1 .right();
t2 =BinTree( u , t2 , t2 );
t8 =BinTree( m , t3 , t8 );
t6 = t2 .right();
cout<< t4 <<endl;
t9 =BinTree( j , t3 , t6 );
t5 =BinTree( z , t1 , t7 );
t1 =BinTree( h , t4 , t6 );
t4 = t2 .right();
t8 = t9 .left();
t8 =BinTree( e , t8 , t6 );
t3 = t8 .left();
t3 =BinTree( p , t3 , t7 );
cout<< t5 <<endl;
t7 = t9 .left();
cout<< t0 <<endl;
cout<<valueAtPostorderPosition( t0 , 10 )<<endl;
t6 = t2 .left();
cout<<valueAtPostorderPosition( t8 , 15 )<<endl;
t1 = t5 .right();
t2 =BinTree( d , t5 , t7 );
t9 =BinTree( n , t4 , t5 );
t3 = t5 .right();
cout<<valueAtPostorderPosition( t0 , 10 )<<endl;
t1 = t9 .left();
t7 = t8 .right();
cout<< t2 <<endl;
t0 =BinTree( q , t7 , t3 );
cout<< t7 <<endl;
cout<<valueAtPostorderPosition( t4 , 5 )<<endl;
t4 =BinTree( y , t7 , t4 );
t5 = t8 .left();
t0 =BinTree( s , t6 , t6 );
cout<< t0 <<endl;
cout<<valueAtPostorderPosition( t0 , 11 )<<endl;
t1 =BinTree( y , t3 , t7 );
cout<<valueAtPostorderPosition( t2 , 19 )<<endl;
cout<< t4 <<endl;
cout<< t2 <<endl;
t4 = t5 .right();
t6 = t0 .left();
t7 = t5 .right();
t7 = t8 .right();
cout<< t3 <<endl;
t8 =BinTree( t , t7 , t4 );
t5 = t8 .left();
t0 = t9 .left();
cout<<valueAtPostorderPosition( t9 , 15 )<<endl;
t3 = t2 .right();
cout<< t1 <<endl;
cout<<valueAtPostorderPosition( t4 , 4 )<<endl;
cout<<valueAtPostorderPosition( t0 , 5 )<<endl;
cout<< t6 <<endl;
t9 =BinTree( j , t6 , t4 );
t0 = t8 .left();
cout<<valueAtPostorderPosition( t5 , 5 )<<endl;
cout<< t4 <<endl;
cout<< t3 <<endl;
cout<<valueAtPostorderPosition( t9 , 10 )<<endl;
cout<<valueAtPostorderPosition( t4 , 4 )<<endl;
t1 = t7 .right();
t4 =BinTree( z , t1 , t5 );
t8 = t3 .right();
t5 = t2 .left();
t7 =BinTree( u , t0 , t3 );
t3 =BinTree( o , t6 , t3 );
cout<<valueAtPostorderPosition( t2 , 19 )<<endl;
cout<< t8 <<endl;
t0 = t2 .right();
cout<< t1 <<endl;
cout<< t0 <<endl;
t0 = t4 .left();
t1 =BinTree( k , t5 , t0 );
t3 = t5 .left();
cout<< t5 <<endl;
cout<<valueAtPostorderPosition( t6 , 5 )<<endl;
t5 =BinTree( o , t0 , t8 );
t7 =BinTree( r , t0 , t0 );
t8 =BinTree( h , t6 , t0 );
t9 = t2 .left();
cout<< t9 <<endl;
t1 = t3 .left();
cout<< t7 <<endl;
t9 =BinTree( p , t6 , t0 );
cout<< t4 <<endl;
t9 = t3 .left();
t8 =BinTree( k , t9 , t6 );
cout<<valueAtPostorderPosition( t7 , 1 )<<endl;
t0 =BinTree( d , t9 , t0 );
t1 =BinTree( g , t3 , t7 );
t1 =BinTree( x , t3 , t0 );
t1 =BinTree( i , t7 , t5 );
cout<< t5 <<endl;
t1 =BinTree( y , t6 , t5 );
cout<<valueAtPostorderPosition( t8 , 7 )<<endl;
cout<< t9 <<endl;
cout<<valueAtPostorderPosition( t9 , 1 )<<endl;
cout<<valueAtPostorderPosition( t2 , 19 )<<endl;
t6 =BinTree( t , t3 , t0 );
t5 =BinTree( x , t6 , t5 );
cout<< t0 <<endl;
cout<<valueAtPostorderPosition( t8 , 7 )<<endl;
cout<< t5 <<endl;
t7 =BinTree( e , t7 , t7 );
cout<< t9 <<endl;
t8 = t8 .right();
t7 =BinTree( p , t0 , t6 );
t2 = t1 .right();
cout<<valueAtPostorderPosition( t7 , 10 )<<endl;
t7 = t6 .right();
t4 = t5 .left();
cout<< t5 <<endl;
cout<< t8 <<endl;
t8 =BinTree( s , t3 , t8 );
cout<<valueAtPostorderPosition( t5 , 13 )<<endl;
cout<<valueAtPostorderPosition( t3 , 4 )<<endl;
cout<< t7 <<endl;
cout<< t2 <<endl;
t2 =BinTree( u , t9 , t7 );
cout<<valueAtPostorderPosition( t7 , 2 )<<endl;
t8 = t1 .right();
```

```
cout<<valueAtPostorderPosition( t1 , 11 )<<endl;
t2 =BinTree( d , t0 , t6 );
cout<< t0 <<endl;
cout<< t1 <<endl;
cout<< t2 <<endl;
cout<< t3 <<endl;
cout<< t4 <<endl;
cout<< t5 <<endl;
cout<< t6 <<endl;
cout<< t7 <<endl;
cout<< t8 <<endl;
cout<< t9 <<endl;
```

Exemple de sortida 3

```
()
()
b(,w)
()
r
()
g
g(s(s(r,w),),)
o(,w)
o(,w)
()
k
l(w(p,h(w,o(,w))),c(s(s(r,w),),w))
()
k
v(,g(s(s(r,w),),))
s
h(w,o(,w))
c
c(w,w)
k(,q(h(w,o(,w)),))
k(,q(h(w,o(,w)),))
g(s(s(r,w),),)
p(h(w,o(,w)),g(s(s(r,w),),))
k(s(s(r,w),),o(,w))
u(s(s(r,w),),k(s(s(r,w),),o(,w)))
s
s(s(r,w),)
z(h(w,o(,w)),s(s(r,w),))
w(s(s(r,w),),g(s(s(r,w),),))
w
e
w
d(z(h(w,o(,w)),s(s(r,w),)),g(h(w,o(,w)),s(s(r,w),)))
g(s(s(r,w),),)
g
s(g(s(s(r,w),),),g(s(s(r,w),),))
s
d
y(g(s(s(r,w),),),g(s(s(r,w),),))
d(z(h(w,o(,w)),s(s(r,w),)),g(h(w,o(,w)),s(s(r,w),)))
s(s(r,w),)
n
y(s(s(r,w),),g(s(s(r,w),),))
s
g
g(s(s(r,w),),)
g
s(s(r,w),)
g(h(w,o(,w)),s(s(r,w),))
j
s
d
s(s(r,w),)
()
g(h(w,o(,w)),s(s(r,w),))
z(h(w,o(,w)),s(s(r,w),))
g
z(h(w,o(,w)),s(s(r,w),))
```

```
r
z(,g(s(s(r,w),),))
r
o(,s(s(r,w),))
k
w
w
d
d(w,)
k
x(t(h(w,o(,w)),d(w,)),o(,s(s(r,w),)))
w
p
x(t(h(w,o(,w)),d(w,)),o(,s(s(r,w),)))
g(s(s(r,w),),)
x
```

```
h
d(w,)
o(,s(s(r,w),))
d
y
d(w,)
y(g(s(s(r,w),),),o(,s(s(r,w),)))
d(d(w,),t(h(w,o(,w)),d(w,)))
h(w,o(,w))
t(h(w,o(,w)),d(w,))
x(t(h(w,o(,w)),d(w,)),o(,s(s(r,w),)))
t(h(w,o(,w)),d(w,))
d(w,)
o(,s(s(r,w),))
w
```

Observació

Podeu seguir l'enfocament que considereu oportú, i podeu utilitzar qualsevol de les estructures de dades presentades al curs (**string, vector, stack, queue, list, map**) com a element de suport, si ho considereu oportú. De totes maneres, `BinTree` ha de jugar un paper rellevant en la vostra solució. Qualsevol solució que ignori això i faci servir enfocaments o estructures de dades alternatives que no formen part de l'assignatura serà invalidada.
Avaluació sobre 10 punts:

- Solució lenta: 5 punts.
- solució ràpida: 10 punts.

Entenem com a solució ràpida una que és correcta, on cada operació té cost **CONSTANT** (excepte per a la d'escriptura d'arbres, a on s'espera cost proporcional a la mida de l'arbre involucrat, i a la d'escriptura del valor a posició en postordre, a on s'espera cost proporcional a la profunditat d'aquella posició en l'arbre involucrat), i capaç de superar els jocs de proves públics i privats. Entenem com a solució lenta una que no és ràpida, però és correcta i capaç de superar els jocs de proves públics.

Informació del problema

Autoria: PRO2

Generació: 2026-01-25T21:24:27.371Z