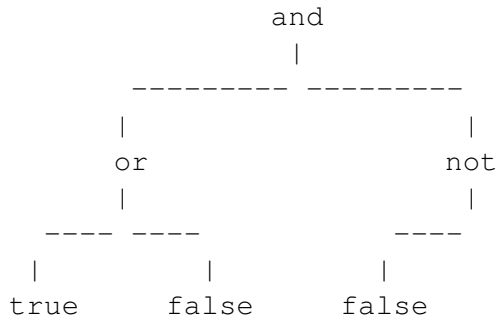
Avaluar expressions booleanes X38940_ca

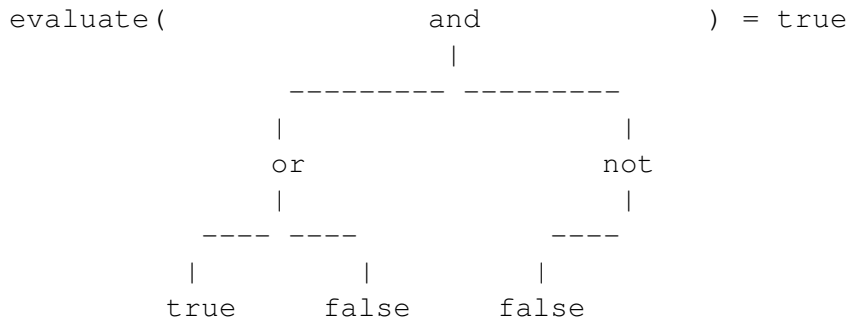
INTRODUCCIÓ:
En aquest exercici considerarem arbres que representen expressions booleanes sobre valors **true, false** i els operadors booleans **and, or, not**. En el cas de **not**, que és un operador amb un sol operand, considerarem que aquest operand és sempre el fill esquerre. Per exemple, el següent arbre representa l'expressió **(true or false) and (not(false))**. Aquesta expressió s'avalua a **true**.



EXERCICI:
Implementeu una funció que, donat un arbre binari d'strings que representa una expressió booleana correcta sobre **true,false** i operadors **and,or,not**, retorna la seva avaluació. Aquesta és la capcelera:

```
// Pre:  t és un arbre no buit que representa una expressió booleana correcta
//       sobre els valors true,false i els operadors and,or,not.
// Post: Retorna l'avaluació de l'expressió representada per t.
bool evaluate(BinTree<string> t);
```

Aquí tenim un exemple de paràmetre d'entrada de la funció i la corresponent sortida:



Fixeu-vos que l'enunciat d'aquest exercici ja ofereix uns fitxers que haureu d'utilitzar per a compilar: `main.cc, BinaryTree.hh, evaluate.hh`. Us falta crear el fitxer `evaluate.cc` amb els corresponents `includes` i implementar-hi la funció anterior. Només cal que pugeu `evaluate.cc` al jutge.
Observació: Per a superar els jocs de proves privats convindrà tenir en compte aquestes optimitzacions típiques d'expressions booleanes:

- Si una expressió e_1 s'avalua a `false`, llavors l'avaluació de (e_1 `and` e_2) és `false` i no requereix avaluar e_2.
- Si una expressió e_1 s'avalua a `true`, llavors l'avaluació de (e_1 `or` e_2) és `true` i no requereix avaluar e_2.

De fet, les operacions `and` i `or` de `C++` ja optimitzen així, de manera que una solució senzilla i natural hauria de poder superar tots els jocs de proves.

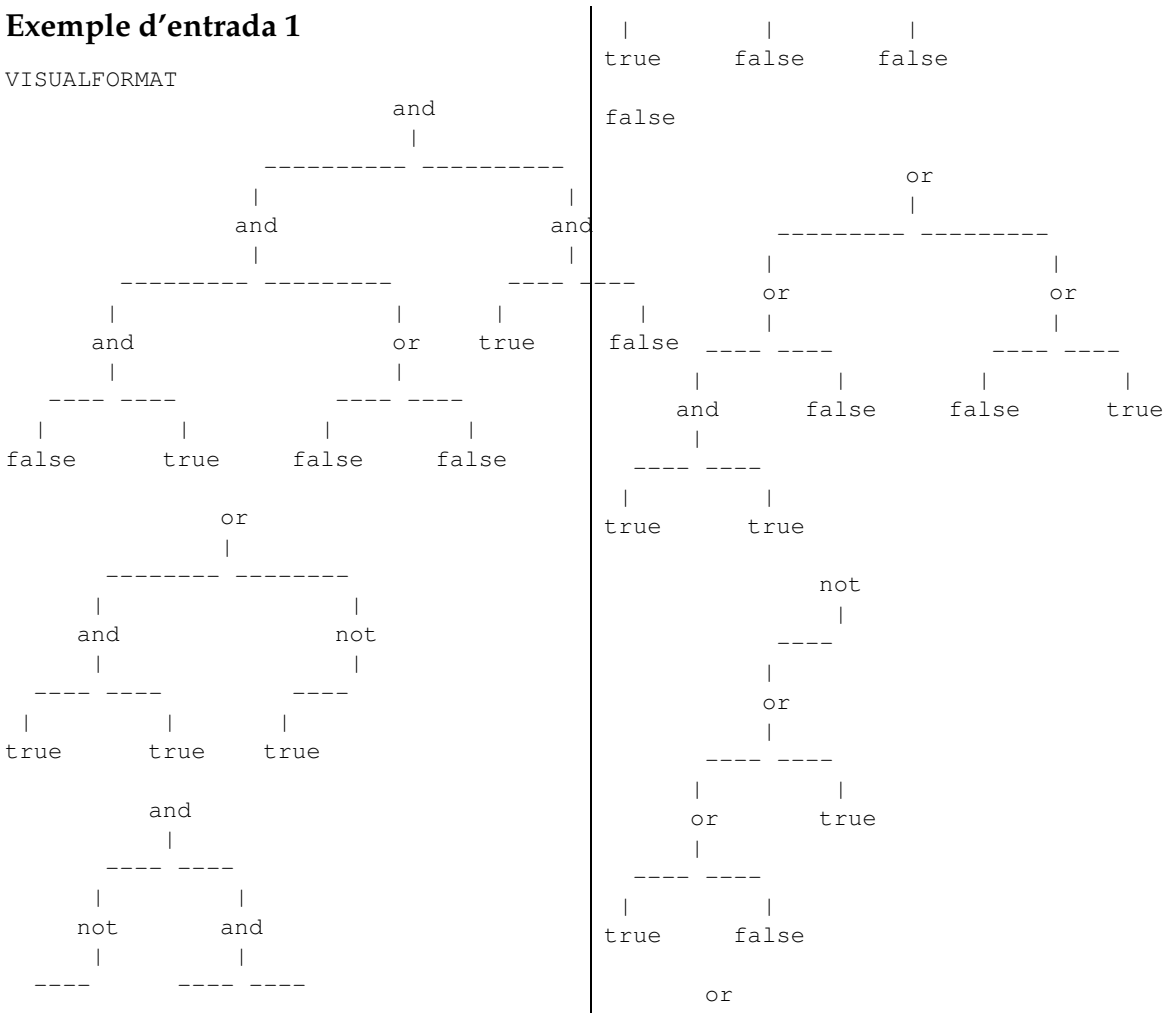
Entrada

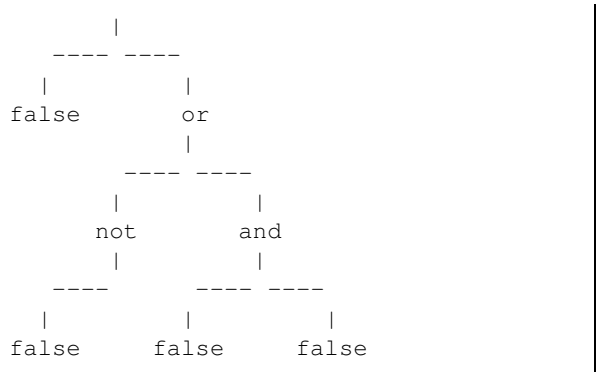
La primera linia de l'entrada descriu el format en el que es descriuen els arbres, o bé INLINEFORMAT o bé VISUALFORMAT. Després venen un nombre arbitrari de casos. Cada cas consisteix en una descripció d'un arbre un arbre binari que representa una expressió booleana correcta. Fixeu-vos en que el programa que us oferim ja s'encarrega de llegir aquestes entrades. Només cal que implementeu la funció abans esmentada.

Sortida

Per a cada cas, la sortida conté la corresponent avaluació de l'arbre. Fixeu-vos en que el programa que us oferim ja s'encarrega d'escriure aquesta avaluació. Només cal que implementeu la funció abans esmentada.

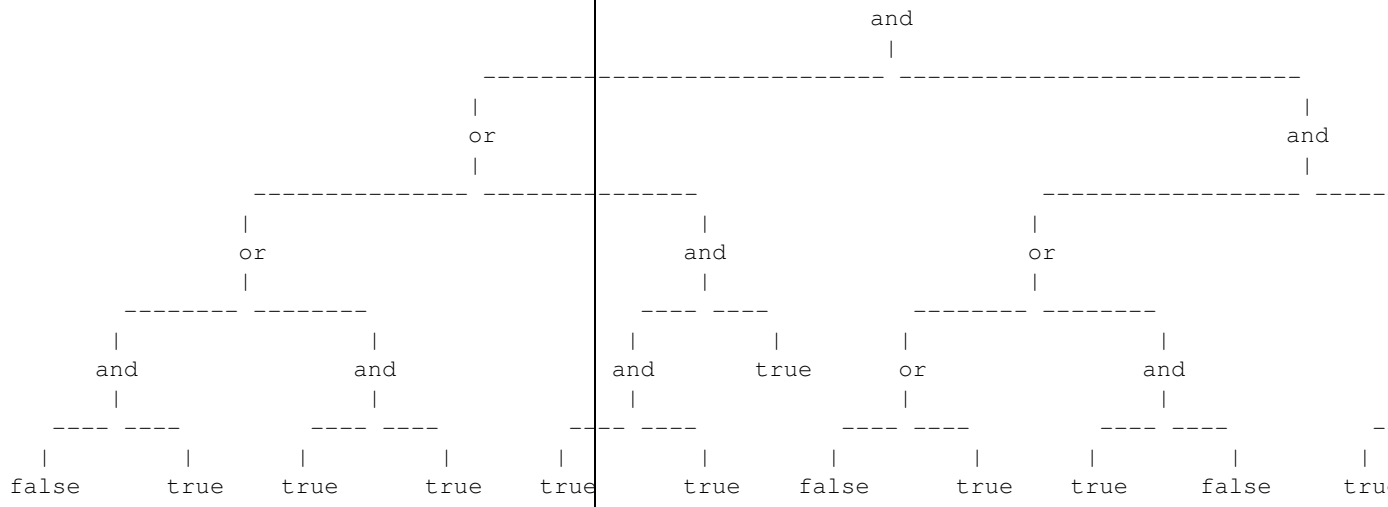
Exemple d'entrada 1

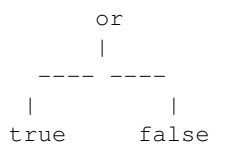


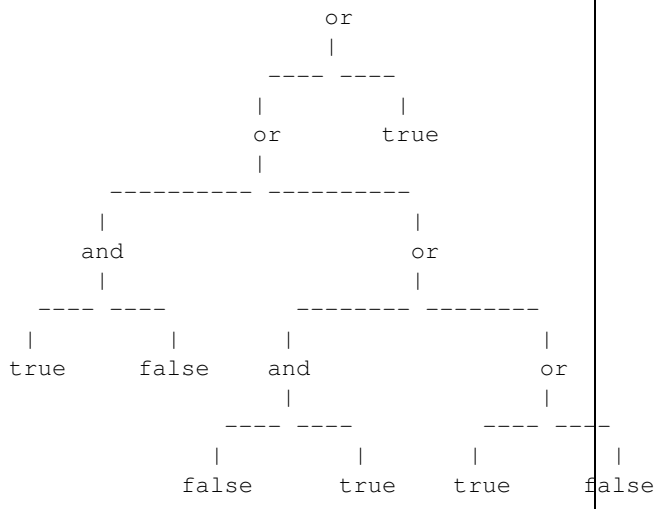


Exemple de sortida 1

```
false
true
false
false
true
false
true
true
true
true
```







Exemple d'entrada 2

```
INLINEFORMAT
and(and(and(false,true),or(false,false)),and(true,false))
or(and(true,true),not(true,))
```

```
and(not(true,),and(false,false))
false
or(or(and(true,true),false),or(false,true))
not(or(or(true,false),true),)
```

```
or(false,or(not(false,),and(false,false)))
and(or(or(and(false,true),and(true,true)),and(true,true)),and(and(true,true),true)),and(or(or(false,true),and(tr
or(true,false)
or(or(and(true,false),or(and(false,true),or(true,false))),true)
```

Exemple de sortida 2

```
false
true
false
false
true
false
true
true
true
true
```

Observació

La vostra funció i subfuncions que creeu han de treballar només amb arbres. Heu de trobar una solució **RECURSIVA** del problema.
Avaluació sobre 10 punts:

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

Entenem com a solució ràpida una que és correcta, de cost lineal i que optimitza operacions booleanes, 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

Autor : PRO2
Generació : 2023-10-24 17:29:54