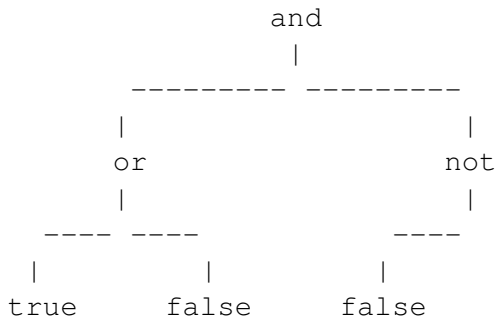
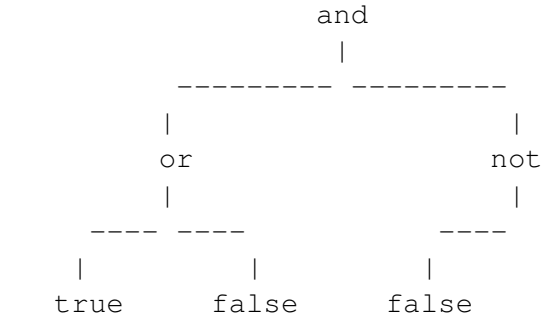
Avaluar expressions booleanes X45696_ca

INTRODUCCIÓ:
En aquest exercici considerarem arbres d'strings 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:
Afegiu un mètode a la classe Arbre que, quan el paràmetre implícit sigui un arbre binari d'strings que representa una expressió booleana correcta sobre **true,false** i operadors **and,or,not**, retorni la seva avaluació.
Mostrem un exemple de paràmetre d'entrada de la funció i la corresponent sortida. Si l'arbre *a* és



llavors tenim que la crida `a.evaluate()` retornarà `true`.
D'entre els fitxers que s'adjunten en aquest exercici, trobareu `Arbre.hh`, a on hi ha una implementació de la classe genèrica Arbre binari. Haureu de buscar dins `Arbre.hh` les següents línies:

```
// Pre:  el, p.i. és un arbre no buit que representa una expressió booleana cor
//       sobre els valors true,false i els operadors and,or,not.
// Post: Retorna l'avaluació de l'expressió representada pel p.i.
// Descomenteu les següents dues línies i implementeu el mètode:
//  bool evaluate(){
// }
```

Descomenteu les dues línies que s'indiquen i implementeu el mètode, fent servir l'operació privada que trobareu just després i que també haureu d'implementar. No toqueu la resta de la implementació de la classe.
La implementació d'aquest mètode hauria de consistir en accedir a nodes mitçançant punters. De fet, possiblement qualsevol altra implementació produirà error d'execució.
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.
D'entre els fitxers que s'adjunten a l'exercici també hi ha `main.cc` (programa principal), i el podeu compilar directament, doncs inclou `Arbre.hh`. Només cal que pugeu `Arbre.hh` al jutge.

Entrada

Un nombre arbitrari d'arbres. Cada cas consisteix en una descripció d'un arbre binari que representa una expressió booleana correcta. La descripció consisteix en un recorregut en preordre del nodes de l'arbre, amb marques on hi anirien els arbres buits. 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

```
and and and false # # true # # or false #
or and true # # true # # not true # # #
and not true # # # and false # # false #
false # #
or or and true # # true # # false # # or
not or or true # # false # # true # # #
or false # # or not false # # # and false
and or or and false # # true # # and true
or true # # false # #
or or and true # # false # # or and false
#
```

Exemple de sortida 1

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

Exemple d'entrada 2

```
and and and false # # true # # or false #
or and true # # true # # not true # # #
and not true # # # and false # # false #
false # #
or or and true # # true # # false # # or
not or or true # # false # # true # # #
or false # # or not false # # # and false
and or or and false # # true # # and true
or true # # false # #
or or and true # # false # # or and false
#
```

Exemple de sortida 2

```
false
true
false
true
false
true
true
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, punters i nodes d'arbre. 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

Autoria: PRO2

Generació: 2026-01-27T18:52:28.559Z