

INTRODUCCIÓ:

En aquest exercici considerarem arbres que representen expressions booleans 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, l'arbre **and(or(true,false),not(false,))** 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 true,false i els operadors and,or,not.
// Post: Retorna l'avaluació de l'expressió representada per t.
bool evaluate(const BinaryTree<string> &t);
```

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

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

Fixeu-vos que l'enunciat d'aquest exercici ja ofereix uns fitxers que haureu d'utilitzar per a compilar: `Makefile`, `program.cpp`, `BinaryTree.hpp`, `evaluate.hpp`. Us falta crear el fitxer `evaluate.cpp` amb els corresponents `includes` i implementar-hi la funció anterior. Quan pugeu la vostra solució al jutge, només cal que pugeu un `tar` construït així:

```
tar cf solution.tar evaluate.cpp
```

Entrada

L'entrada té un nombre arbitrari de casos. Cada cas consisteix en una línia amb un string describint un arbre binari d'strings 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

```
and(and(and(false,true),or(false,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(true,false)
and(or(or(false,true),and(true,true)),and(and(true,
or(true,false)
or(or(and(true,false),or(and(false,true),or(true,false
```

Exemple de sortida

```
false  
true  
false  
false
```

```
| true  
| false  
| true  
| true  
| true  
| true
```

Observació

La vostra funció i subfuncions que creu han de treballar només amb arbres. Heu de trobar una solució **RECURSIVA** del problema. En les crides recursives, incloeu la hipòtesi d'inducció, és a dir una explicació del que es compleix després de la crida, i també la funció de fita/decreixement o una justificació de perquè la funció recursiva acaba.

Avaluació sobre 10 punts:

- Solució lenta: 6 punts.
- Solució lenta + justificació: 7 punts.
- solució ràpida: 9 punts.
- solució ràpida + justificació: 10 punts.

Informació del problema

Autor : PRO1

Generació : 2022-12-07 12:11:46

© *Jutge.org*, 2006–2022.

<https://jutge.org>