## Avaluar expressions amb divisió X43096_ca

**INTRODUCCIÓ:**
En aquest exercici considerarem arbres que representen expressions sobre els operadors `+`, `-`, `*`, `/`, i sobre operands naturals. Per exemple, l'arbre `-(+(3,/(4,2)),5)` representa l'expressió `3+4/2-5`.
Alhora d'avaluar una divisió, interpretem la divisió entera que ens ofereix C++. Noteu que, en particular, $(-5)/2 = -2$, contradient la definició que trobem habitualment en llibres de matemàtiques.
Noteu també que la divisió per 0 no està definida, i això ho haurem de tenir en compte en resoldre l'exercici.
**EXERCICI:**
Implementeu una funció que, donat un arbre binari d'strings que representa una expressió correcta sobre naturals i operadors `+`, `-`, `*`, `/`, retorna la seva avaluació i un indicador de si s'ha produït un error de divisió per 0, tot mitjançant paràmetres per referència. Aquesta és la capcelera:

```
// Pre:  t és un arbre no buit que representa una expressió correcta
//       sobre els naturals i els operadors +,-,*,/.
//       Les operacions no produeixen errors d'overflow,
//       però poden produïr error de divisió per 0.
// Post: Si l'avaluació de l'expressió representada per t no produeix errors de
//       llavors 'result' val l'avaluació d'aquesta expressió i 'error' val 'fa
//       En cas contrari, 'error' val 'true', i el valor de 'result' és irrelle
void evaluate(const BinaryTree<string> &t, int &result, bool &error);
```

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

```
evaluate(/(+(1,2),-(5,2)), result, error) produces result=1, error=false
```

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`, `utils.hpp`, `utils.cpp`. Us falta crear el fitxer `evaluate.cpp` amb els corresponents `includes` i implementar-hi la funció anterior. Valdrà la pena que utilitzeu algunes de les funcions oferides a `utils.hpp`. 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. 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 o bé una indicació de que s'ha produït un error de divisió per 0 durant el procès d'avaluar 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

```
-(+(-(5,2),*(4,4)),-(1,8))
+(/(+(6,8),-(6,6)),-(5,4))
5
*(*(/(5,8),2),-(*(7,2),*(4,7)))
*(-(+(1,4),-(4,4)),+(+(5,5),+(2,2)))
-(+(6,8),-(3,6))
+(6,-(*(5,4),*(3,6)))
*(-(+(5,*(4,1)),+(-(6,5),+(3,4))),8)
*(4,-(*(1,*(1,1)),-(7,8)))
4
-(*(*(-(5,5),5),+(5,+(3,6))),*(3,-(5,1)))
5
-(*(8,6),+(7,5))
+(6,5)
*(5,3)
*(-(4,3),5)
/(7,*(-(*(8,4),-(4,5)),-(3,+(7,6))))
2
7
*(3,+(+(7,+(4,1)),/(+(3,8),+(2,6))))
```

## Exemple de sortida 1

```
26
Division by 0
5
0
70
17
8
8
8
4
-12
5
36
11
15
5
0
2
7
39
```

## Exemple d'entrada 2

```
+(12,52)
44
+(65,19)
5
-(-(/(-(7,20),+(71,97)),+(/(75,29),-(87,64))),37)
-(-(89,-(*(77,72),38)),/(92,31))
+(-(77,100),8)
*(/(-(14,89),+(47,7)),-(+(+(93,89),+(65,43)),38))
100
+(/(1,56),+(64,72))
44
-(/(*(+(97,+(97,39)),-(54,76)),/(/(75,/(21,84)),41)),3)
*(-(49,63),/(77,73))
/(+(36,77),-(57,*(23,60)))
*(44,-(83,8))
-(35,*(96,39))
-(+(55,87),-(-(*(60,81),/(53,14)),99))
-(/(25,*(+(27,94),64)),44)
-(59,+(*(57,-(92,33)),-(+(-(27,12),*(83,20)),29)))
*(10,39)
```

## Exemple de sortida 2

```
64
44
84
5
-62
-5419
-15
-388
100
136
44
Division by 0
-14
0
3300
-3709
-4616
-44
-5029
390
```

## Informació del problema

Autor : PRO1
Generació : 2022-09-14 11:20:24