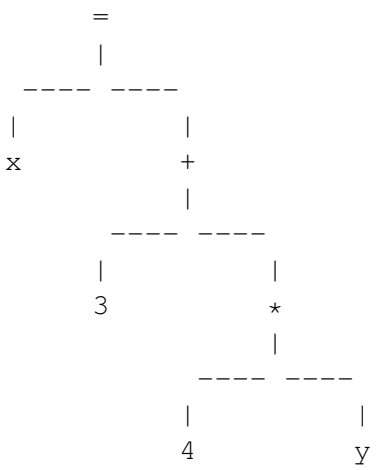
Executar instruccions d'assignació — X33949_ca

PRELIMINARS:
En aquest exercici assumim que ja heu resolt un exercici anterior a on havieu d'avaluar expressions amb variables. De fet, assumim que heu fet això creant un fitxer `evaluate.cc` amb la implementació de la següent funció:

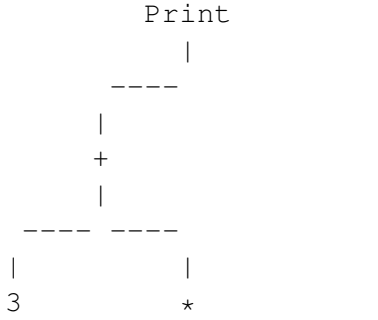
```
// Pre:  t és un arbre no buit que representa una expressió correcta
//       sobre naturals i variables enteres, i els operadors +,-,*.
//       Totes les variables que apareixen a t estan definides a variable2value
//       Les operacions no produeixen errors d'overflow.
// Post: Retorna l'avaluació de l'expressió representada per t.
int evaluate(map<string,int> &variable2value, BinTree<string> t);
```

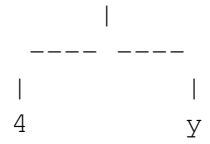
INTRODUCCIÓ:
En aquest exercici considerarem arbres que representen instruccions. Hi ha dos tipus d'instruccions, que es representen amb arbres tal i com s'explica a continuació:

- Instrucció d'assignació `x = e`, on `x` és una variable i `e` és una expressió. Per exemple, el següent arbre representa la instrucció `x=3+4*y`.



- Instrucció d'escriure per la sortida estandard `Print(e)`, on `e` és una expressió. Per exemple, el següent arbre representa la instrucció `Print(3+4*y)`.





Per a guardar els valors assignats sobre les variables usarem un map d'strings a enters anomenat `variable2value`.

EXERCICI:

Implementeu una funció que, donat un `map<string,int>` anomenat `variable2value`, i donat un arbre binari d'strings `t` que representa, o bé una instrucció d'assignació, o bé una instrucció d'escriptura per la sortida estandard, executa la instrucció, modificant `variable2value` o escrivint per la sortida, segons el cas. Aquesta és la capcelera:

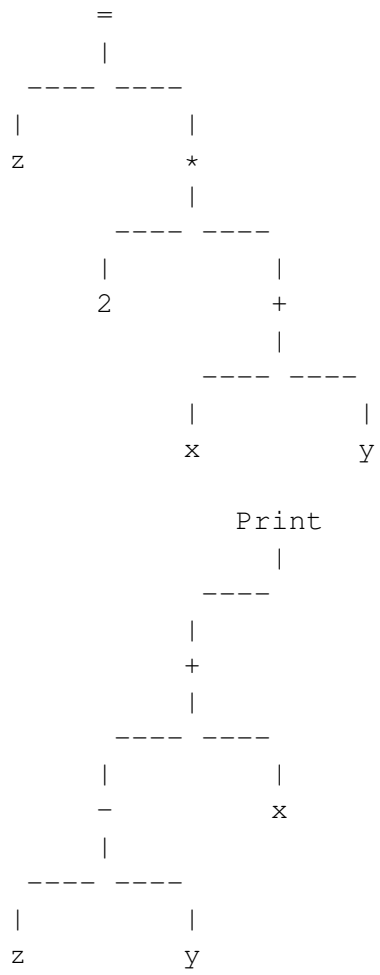
```
// Pre:  t és un arbre no buit que representa o bé una instrucció d'assignació
//       o bé una instrucció d'escriure per la sortida estandard.
//       Totes les variables que apareixen a la expressió de t estan definides
//       En el cas d'assignació, la variable esquerra podria no estar definida
//       Les operacions no produeixen errors d'overflow.
// Post: modifica variable2value o escriu per la sortida estandard un valor,
//       simulant exactament la instrucció que representa t.
void execute(map<string,int> &variable2value, BinTree<string> t);
```

La entrada del programa que crida a la funció `execute` consisteix en una seqüència d'instruccions representades amb arbres tal i com hem explicat. Al principi, el programa assumeix que `variable2value` és un map buit. Considereu les següents instruccions d'exemple:

```
x=3
y=3+x
z=2*(x+y)
Print(z-y+x)
```

Aquestes instruccions, com a entrada del programa, queden representades així:

```
VISUALFORMAT
     =
     |
 ---- ----
|         |
x         3

     =
     |
 ---- ----
|         |
y         +
          |
       ---- ----
     |         |
     3         x
```



Si anem cridant a `execute` passant aquests arbres, l'un després de l'altre, i el mapa `variable2value`, al final, hauriem de veure per la sortida el valor `15`.
Fixeu-vos que l'enunciat d'aquest exercici ja ofereix uns fitxers que haureu d'utilitzar per a compilar: `Makefile, program.cc, BinaryTree.hh, evaluate.hh, execute.hh, utils.hh, utils.cc`. Us falta afegir el fitxer `evaluate.cc` que teniu fet d'un exercici anterior, i crear el fitxer `execute.cc` amb els corresponents `includes` i implementar-hi la funció `execute` que hem explicat. Valdrà la pena que utilitzeu algunes de les funcions oferides a `utils.hh`. Quan pugeu la vostra solució al jutge, només cal que pugeu un `tar` construït així:

```
tar cf solution.tar execute.cc evaluate.cc
```

Entrada

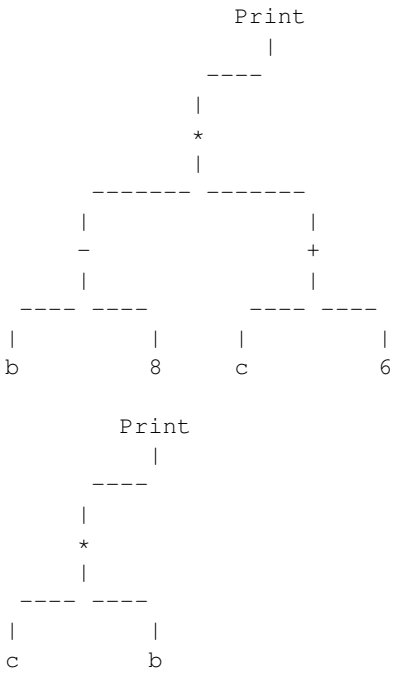
La primera linia de l'entrada descriu el format en el que es descriuen els arbres, o bé INLINEFORMAT o bé VISUALFORMAT. Després ve una seqüència d'arbres binaris d'strings que alhora representen instruccions. Fixeu-vos en que el programa que us oferim ja s'encarrega de llegir aquesta entrada. Només cal que implementeu la funció abans esmentada.

Sortida

El programa dona com a sortida el que seria la sortida resultant d'executar la seqüència d'instruccions donada d'entrada. Fixeu-vos en que el programa que us oferim ja s'encarrega d'escriure aquesta sortida. Només cal que implementeu la funció abans esmentada.

Exemple d'entrada 1

```
VISUALFORMAT
      =
      |
  ---- ----
|          |
a          +
           |
       ---- ----
      |          |
      1          2

   Print
      |
  ----
|
a

      =
      |
  ---- ----
|          |
b          *
           |
       ---- ----
      |          |
      3          4

      =
      |
  ---- ----
|          |
b          *
           |
       ---- ----
      |          |
      +          2
      |
  ---- ----
|          |
a          2

   Print
      |
  ----
|
b

      =
      |
  ---- ----
|          |
a          -
           |
       ---- ----
      |          |
      9          b

                Print
                   |
                ----
               |
               -
               |
       -------- --------
      |                 |
      -                 *
      |                 |
  ---- ----         ---- ----
|          |      |          |
6          1      2          a

      =
      |
  ---- ----
|          |
c          *
           |
       ---- ----
      |          |
      2          a

         =
         |
     ---- ----
    |          |
    a          +
               |
       -------- --------
      |                 |
      *                 *
      |                 |
  ---- ----         ---- ----
|          |      |          |
c          1      3          c

   Print
      |
  ----
|
a

      =
      |
  ---- ----
|          |
b          1
```



Exemple de sortida 1

```
3
10
7
-8
-28
-2
```

Exemple d'entrada 2

```
INLINEFORMAT
Print(-(+(+(4,7),-(6,8)),-(8,2)),)
=(c,-(-(*(6,1),4),-(+(-(1,1),6),6)))
Print(+(c,5),)
=(c,7)
=(cc,+(*(-(+(6,c),c),4),*(7,c)))
Print(-(-(+(+(c,3),-(7,2)),-(+(4,1),+(4,cc))),*(*(-(1,c),-(cc,2)),*(+(6,6),-(7,6)))),)
=(b,*(9,+(+(+(9,4),-(cc,cc)),+(3,8))))
Print(-(c,*(2,2)),)
=(d,-(-(+(*(3,9),4),*(9,-(5,cc))),-(b,*(b,7))))
Print(-(*(*(8,8),+(5,cc)),+(-(7,-(b,d)),+(-(1,cc),*(4,cc)))),)
Print(*(3,3),)
=(b,b)
Print(*(*(1,-(1,-(3,6))),9),)
=(da,*(2,9))
Print(-(+(5,2),-(da,8)),)
Print(*(5,-(cc,d)),)
=(b,-(+(-(c,d),2),*(+(1,b),-(da,4))))
=(b,8)
Print(-(+(*(-(da,6),+(b,da)),*(-(c,6),*(4,3))),-(da,*(1,*(cc,da)))),)
Print(-(-(+(d,5),cc),*(*(c,2),*(+(3,7),6))),)
```

Exemple de sortida 2

```
3
7
5199
3
3041
9
-36
-3
-9330
1620
-1021
```

Exemple d'entrada 3

```
INLINEFORMAT
=(a,+(1,2))
Print(a)
=(b,*(3,4))
=(b,*(+(a,2),2))
Print(b)
=(a,-(9,b))
Print(-(-(6,1),*(2,a)),)
=(c,*(2,a))
=(a,+(*(c,1),*(3,c)))
Print(a)
=(b,1)
Print(*(-(b,8),+(c,6)),)
Print(*(c,b),)
```

Exemple de sortida 3

```
3
10
```

```
7
-8
-28
-2
```

Informació del problema

Autor : PRO2
Generació : 2023-10-21 13:48:03