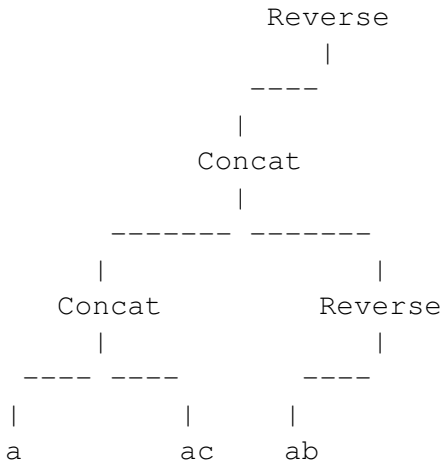
Avaluar expressions amb strings X69228_ca

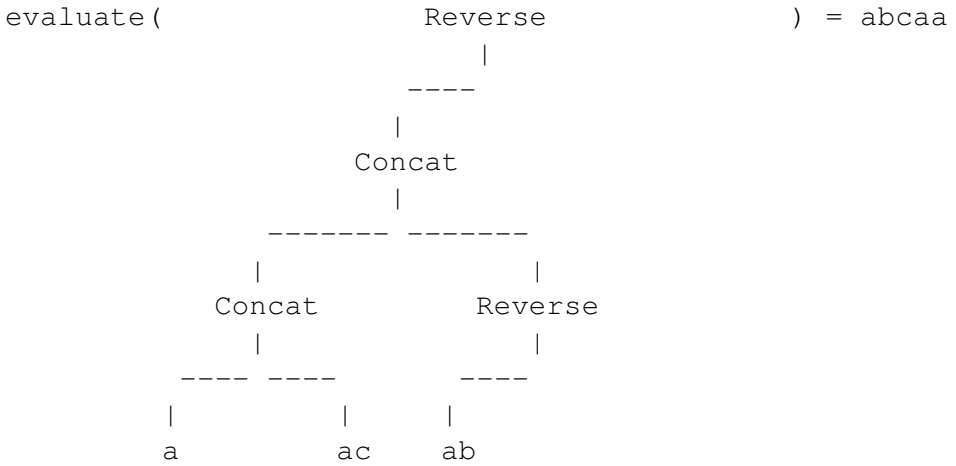
INTRODUCCIÓ:
En aquest exercici avaluarem arbres que representen expressions sobre valors de tipus string (de lletres minúscules) i els operadors de concatenació de dos strings i revessat de un string **Concat, Reverse**. En el cas de **Reverse**, que és un operador amb un sol operand, considerarem que aquest operand és sempre el fill esquerre. Per exemple, el següent arbre s'avalua a **abcaa**.



EXERCICI:
Implementeu una funció que, donat un arbre binari d'strings que representa una expressió correcta sobre strings de lletres minúscules i operadors **Concat, Reverse**, retorna la seva avaluació. Aquesta és la capcelera:

```
// Pre:  t és un arbre no buit que representa una expressió correcta
//       sobre strings de lletres minúscules i els operadors Concat, Reverse.
// Post: Retorna l'avaluació de l'expressió representada per t.
string 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, BinTree.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.

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 binari que representa una expressió 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.

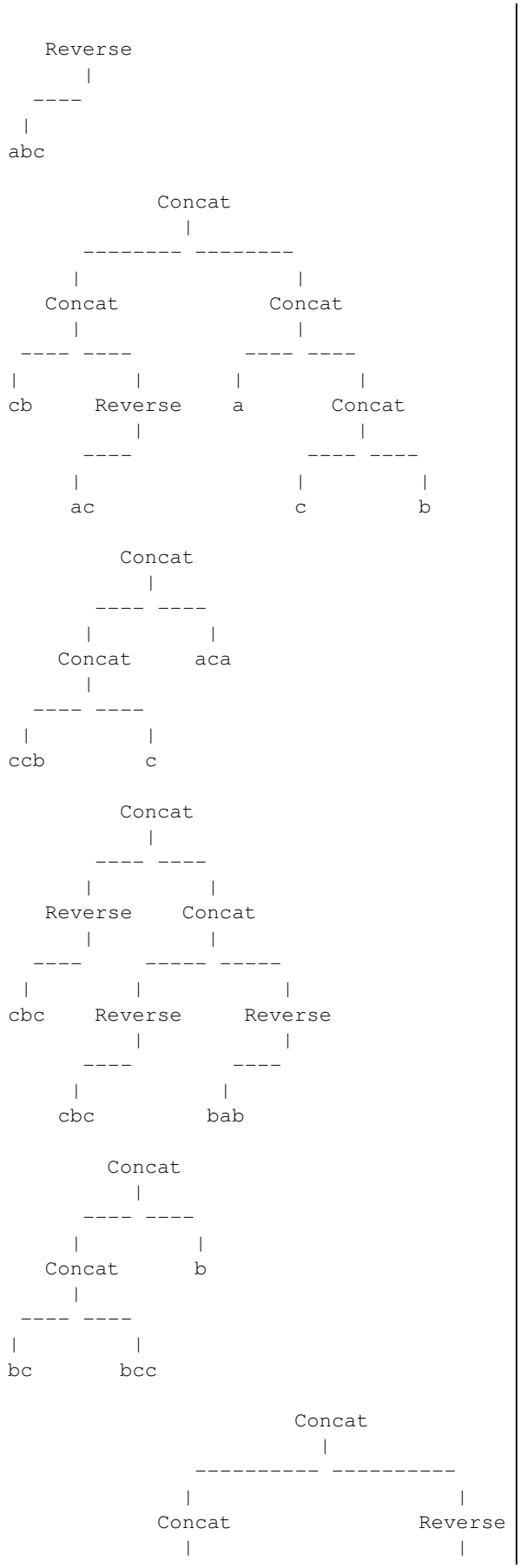
Sortida

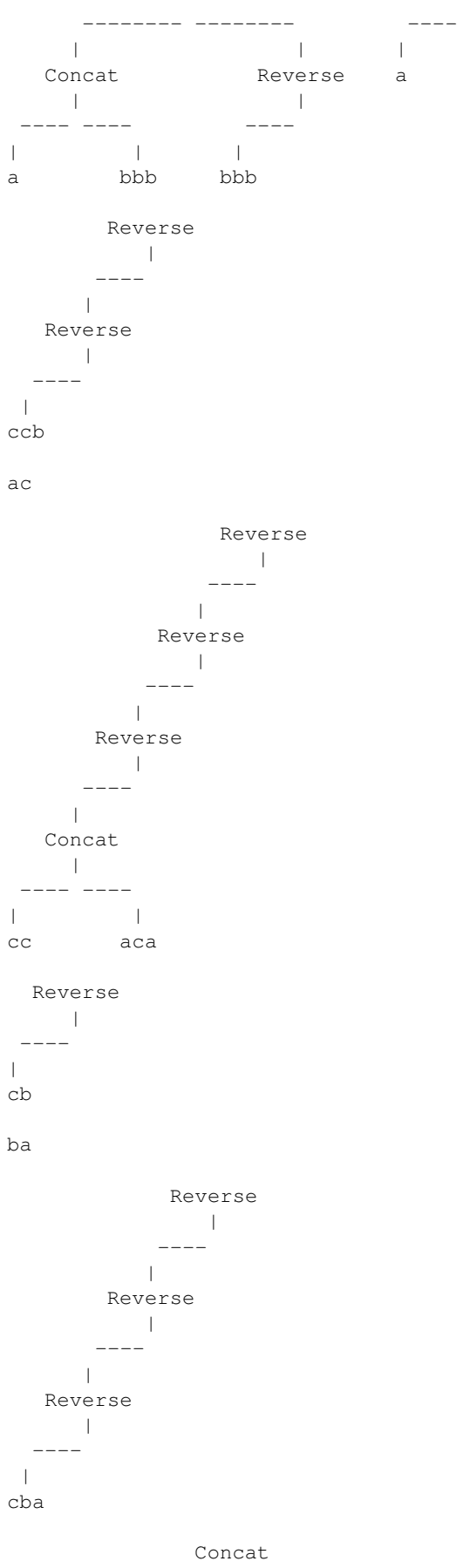
Per a cada cas, la sortida conté la corresponent avaluació de l'arbre. Fixeu-vos en que el programa que us oferim ja s'encarrega d'escriure aquesta avaluació. Només cal que implementeu la funció abans esmentada.

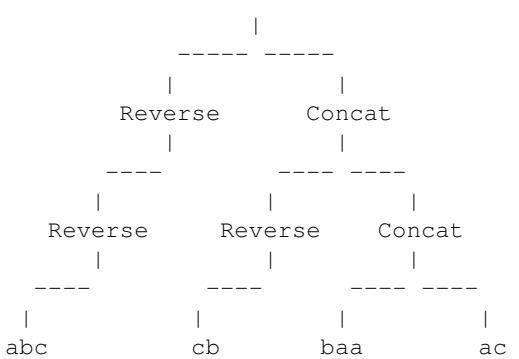
Exemple d'entrada 1

```
VISUALFORMAT
                          Concat
                            |
                 ----------- -----------
                |                       |
             Concat                  Reverse
               |                       |
       -------- --------              ----
      |                 |           |
    Concat            Reverse     bbb
      |                 |
  ---- ----           ----
 |         |        |
bcb       baa      caa

                Reverse
                   |
                ----
               |
             Concat
               |
           ---- ----
          |         |
       Reverse      c
          |
       ----
      |
  Reverse
     |
 ----
|
a

                  Reverse
                     |
                  ----
```

```
                |
              Concat
                |
          ----- -----
         |           |
       Reverse      Concat
         |            |
      ----         ---- ----
     |            |         |
  Reverse      Reverse     bab
     |            |
   ----         ----
  |            |
acc          c

   Concat
     |
 ---- ----
|         |
c      Reverse
          |
       ----
      |
    acb

            Reverse
               |
            ----
           |
        Reverse
           |
        ----
       |
    Concat
      |
 ---- ----
|         |
aa       abb
```


Reverse
abc
Concat
Concat
Concat
cb
Reverse
a
Concat
ac
c
b
Concat
Concat
aca
ccb
c
Concat
Reverse
Concat
cbc
Reverse
Reverse
cbc
bab
Concat
Concat
b
bc
bcc
Concat
Concat
Reverse


Concat
Reverse
a
a
bbb
bbb
Reverse
Reverse
ccb
ac
Reverse
Reverse
Reverse
Reverse
Concat
cc
aca
Reverse
cb
ba
Reverse
Reverse
Reverse
Reverse
cba
Concat



cca

aaa

Exemple de sortida 1

```
bcbbaaaacbbb
ca
babccca
cbca
aaabb
cba
cbcaacb
ccbcaca
cbccbcbab
bcbccb
abbbbbba
ccb
ac
acacc
bc
ba
abc
abcbcbaaac
cca
aaa
```

Exemple d'entrada 2

```
INLINEFORMAT
Concat(Concat(Concat(bcb,baa),Reverse(caa,)),)
Reverse(Concat(Reverse(Reverse(a,),),c),)
Reverse(Concat(Reverse(Reverse(acc,),),Concat(a,
Concat(c,Reverse(acb,))
Reverse(Reverse(Concat(aa,abb),),)
Reverse(abc,)
Concat(Concat(cb,Reverse(ac,)),Concat(a,Concat(ca,
Concat(Concat(ccb,c),aca)
Concat(Reverse(cbc,),Concat(Reverse(cbc,),Reverse(
Concat(Concat(bc,bcc),b)
Concat(Concat(Concat(a,bbb),Reverse(bbb,)),c)
Reverse(Reverse(ccb,),)
ac
Reverse(Reverse(Reverse(Concat(cc,aca),),),)
Reverse(cb,)
ba
Reverse(Reverse(Reverse(cba,),),)
Concat(Reverse(Reverse(abc,),),Concat(Reverse(
cca
aaa
```

Exemple de sortida 2

```
bcbbaaaacbbb
ca),Reverse(bbb,))
babccca
cbca(Reverse(c,),bab)),)
aaabb
cba
cbcaacb
ccbca(ca,b)))
cbccbcbab
bcbccb,Reverse(bab,)))
abbbbbba
cc,Reverse(a,))
ac
acacc
bc)
ba
abc
abcbcbaaac
cca(cb,),Concat(baa,ac)))
aaa
```

Observació

La vostra funció i subfuncions que creeu han de treballar només amb arbres. Heu de trobar una solució **RECURSIVA** del problema.
En aquest exercici necessitareu un càlcul auxiliar que revessi un string. Podeu revessar string iterativament, però pot estar bé que, per a practicar, el calculeu recursivament. Convindrà, en tal cas, usar pas per referència, a fi d'aconseguir una solució eficient.

Informació del problema

Autor : PRO2
Generació : 2023-10-21 13:50:04