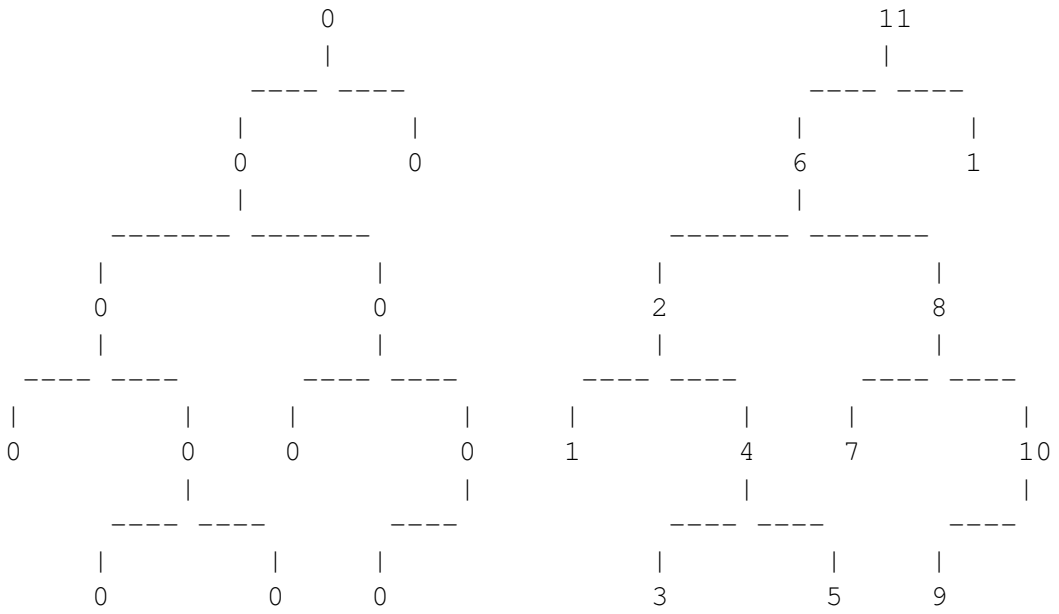
Reemplaçar subarbres en posicions indicades en inordre X80136_ca

Nota: Implementar la versió eficient d'aquest exercici és **difícil**. Us encoratgem fortament a que comenceu implementant una versió ineficient per a superar els jocs de proves públics i obtenir així la meitat de la nota. A continuació hi ha la descripció de l'exercici. I després hi ha una guia per a obtenir una versió ineficient.
Preliminars: Recordeu que el recorregut en inordre d'un arbre és la llista dels nodes de l'arbre ordenada com segueix: en primer lloc, el recorregut en inordre del fill esquerra de l'arbre, després l'arrel de l'arbre, i després el recorregut en inordre del fill dret de l'arbre. En altres paraules:

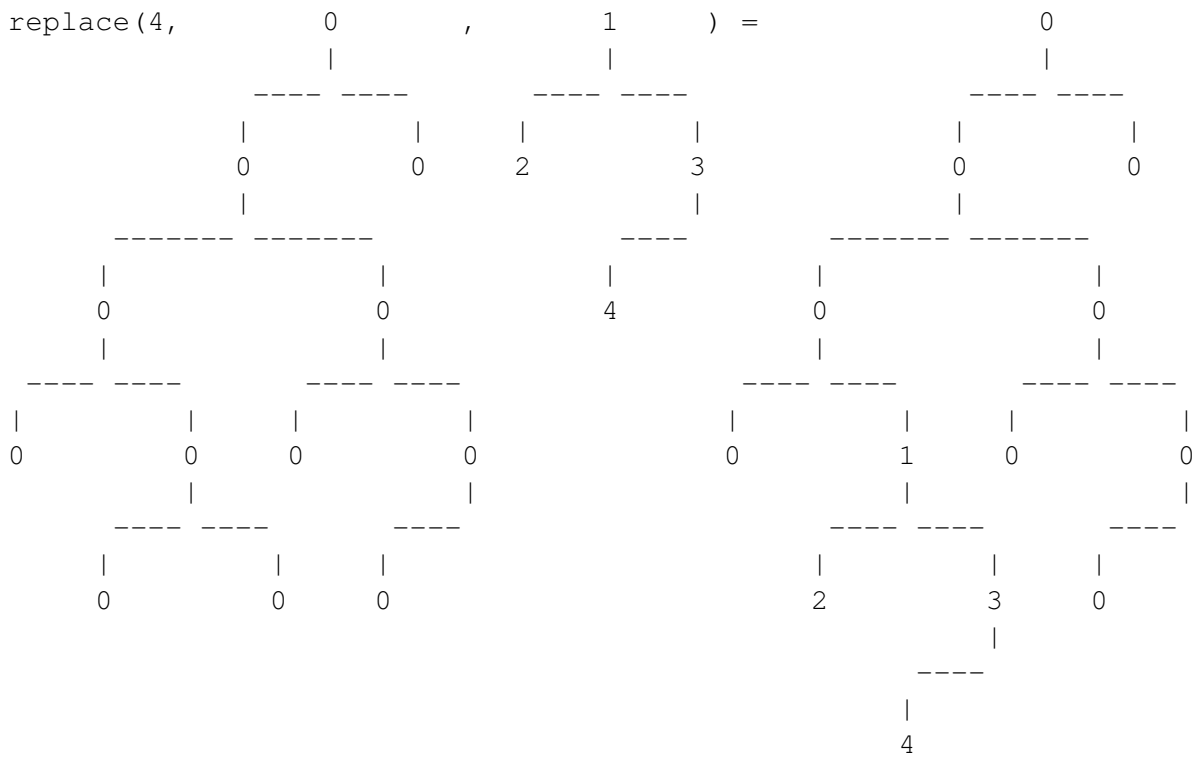
- $Inordre(x(t_1, t_2)) = Inordre(t_1) \cdot x \cdot Inordre(t_2)$
- $Inordre(()) = ()$, és a dir, l'inordre de l'arbre buit és l'arbre buit.

Per exemple, a continuació tenim un arbre a l'esquerra (amb només 0s), i a la dreta hi tenim un altre arbre amb la mateixa estructura (mateix conjunt de posicions), però a cada node hi ha el que seria el seu index en el recorregut en inordre:



Exercici:
Heu d'implementar un programa que llegeix un arbre d'entrada, i una llista d'operacions que, o bé modifiquen aquest arbre, o bé escriuen per la sortida el valor actual d'aquest arbre. Les operacions que modifiquen l'arbre simplement consisteixen en reemplaçar un subarbre per un nou subarbre. Però, la posició del subarbre a reemplaçar s'indica amb el seu índex en inordre.
Per exemple, suposeu que el valor de l'arbre actual és l'arbre amb 0s vist abans, i que ens demanen reemplaçar la seva posició indexada per 4 pel nou subarbre 1(2,3(4,)). Aquest serà el resultat del reemplaçament:

```
replace(4, 0(0(0(0,0(0,0)),0(0,0(0,))),0), 1(2,3(4,))) = 0(0(0(0,1(2,3(4,))),0(
```



Guia per a obtenir una solució ineficient però senzilla
La idea és implementar la funció `replace` anterior, que podria tenir aquesta capcelera:

```
// Pre:  Let n be t.size(). Then, 1<=index<=n.
//       Let's call p to the position of t whose inorder index is the value of
// Post: returns the result of modifying t by replacing the subtree at position
BT replace(int index, const BT t, const BT tsub);
```

Per a completar aquesta guia, us oferim el programa complert, a on només cal que hi afegiu la implementació de la funció `replace`.

```
#include <iostream>
#include <string>
#include <cstdlib>
// Add more includes if you wish.
// ...

using namespace std;

#include "BinTree.hh"

typedef BinTree<int> BT;

void setFormat(BT &t, string format)
{
  t.setInputOutputFormat(format=="INLINEFORMAT"?
 BT::INLINEFORMAT:
```

```
 BT::VISUALFORMAT);
}

// Add auxiliary functions if you wish.
// ...

// Pre:  Let n be t.size(). Then, 1<=index<=n.
//       Let's call p to the position of t whose inorder index is the value of
// Post: returns the result of modifying t by replacing the subtree at position
BT replace(int index, const BT t, const BT tsub)
{
  // Implement this function.
  // ...
}

int main()
{
  string format;
  getline(cin, format);
  BT t;
  setFormat(t, format);
  cin >> t;
  string command;
  while (cin >> command) {
    if (command == "PRINT") {
      setFormat(t, format);
      cout << t << endl;
    } else if (command == "REPLACE") {
      int index;
      cin >> index;
      string aux;
      getline(cin, aux);
      //cin.ignore();
      BT tsub;
      setFormat(tsub, format);
      cin >> tsub;
      t = replace(index, t, tsub);
    }
  }
}
```

Fixeu-vos que l'enunciat d'aquest exercici us ofereix el fitxer `BinTree.hh`. Us falta crear el fitxer `main.cc`, que haurieu de construïr a partir de la plantilla que us hem oferit abans o una modificació substancial d'ella per a la versió eficient, fent un ús convenient del tipus `BinTree`. Només cal que pugeu `main.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. La segona línia conté un arbre binari d'enters. A partir de

la tercera línia, venen les comandes, que son de dos tipus:

- PRINT
- REPLACE index
  tree

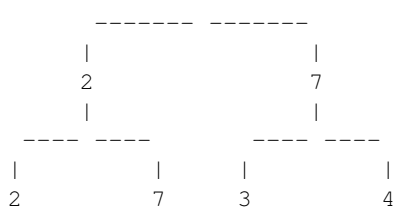
Com podeu observar, la primera comanda és simplement el mot `PRINT`, i el programa ha de respondre escrivint l'arbre actual per la sortida. La segona comanda ve en dues línies, la primera amb el mot REPLACE i l'índex en inordre del node a on s'ha de fer el reemplaçament, i la segona línia amb el nou subarbre per al reemplaçament. És recomanable que, abans de llegir aquest arbre, executeu `getline(cin,s)` o `cin.ignore()` per assegurar que el subarbre es llegeix bé en el cas de `VISUALFORMAT`.
Es garantitza que l'índex de les comandes `REPLACE` sempre serà vàlid per al valor actual de l'arbre.

Sortida

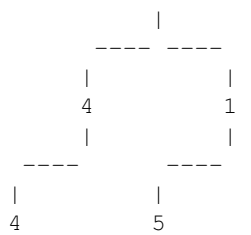
Per a cada cas de `PRINT`, cal escriure l'arbre binari actual.

Exemple d'entrada 1

```
VISUALFORMAT
     8
     |
 ---- ----
|         |
6         4

REPLACE 3
        5
        |
 ------- -------
|               |
6               9
|               |
 ----       ----
     |     |
     1     7

PRINT
REPLACE 1
6
|
 ----
     |
     4

PRINT
REPLACE 4
     7
     |
 ---- ----
|         |
3         6

PRINT
REPLACE 8
     9
     |
 ----
|
6

PRINT
REPLACE 9
3

PRINT
REPLACE 1
1

PRINT
REPLACE 1
6

PRINT
REPLACE 1
7

PRINT
REPLACE 1
     1
     |
 ---- ----
|         |
8         3
          |
      ---- ----
     |         |
     3         4

PRINT
REPLACE 3
             8
             |
```

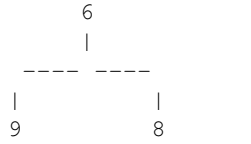


PRINT
REPLACE 6



PRINT
REPLACE 1
5

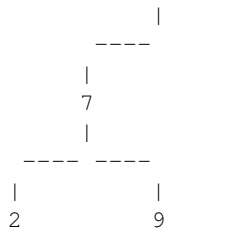
PRINT
REPLACE 16



PRINT
REPLACE 9
8

PRINT
REPLACE 17
8

PRINT
REPLACE 1



PRINT
REPLACE 4
1

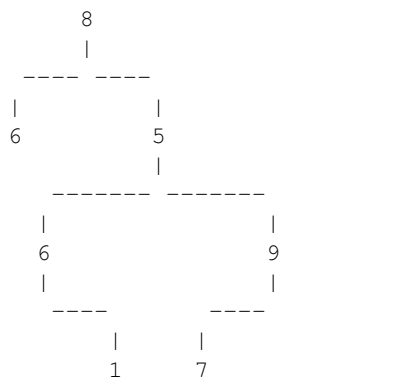
PRINT
REPLACE 12
3

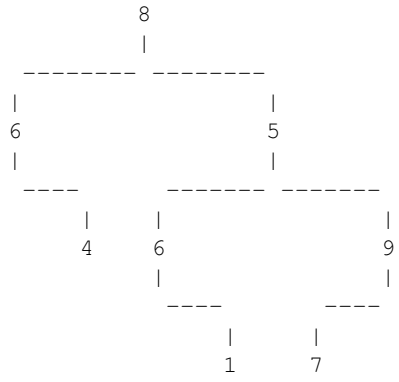
PRINT
REPLACE 11
3

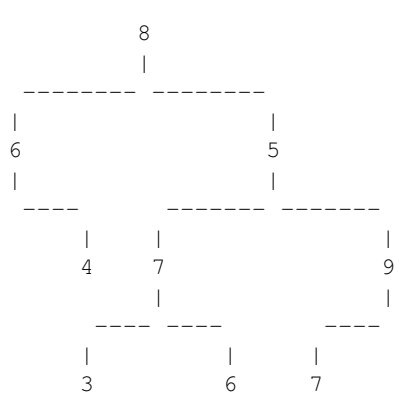
PRINT
REPLACE 1
9

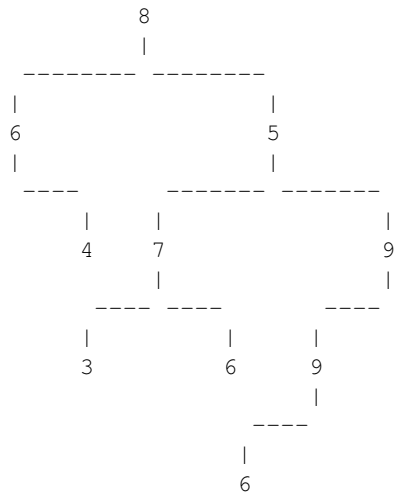
PRINT

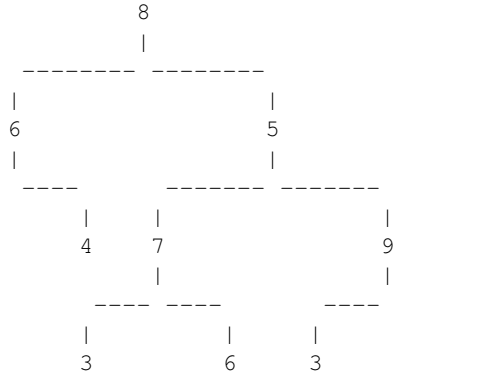
Exemple de sortida 1

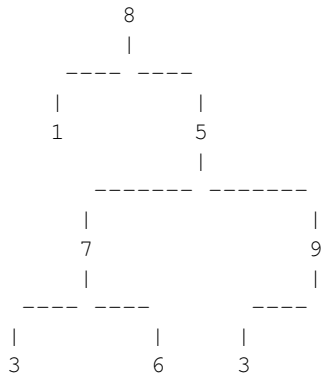


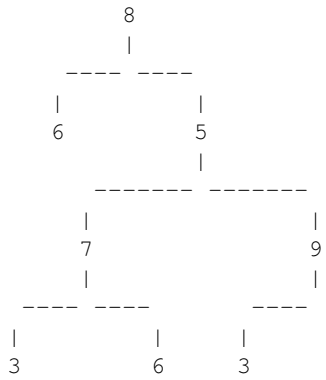


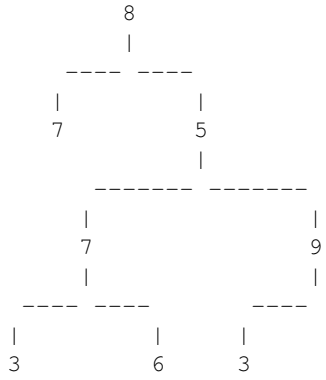


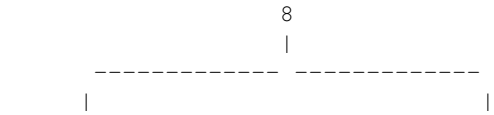


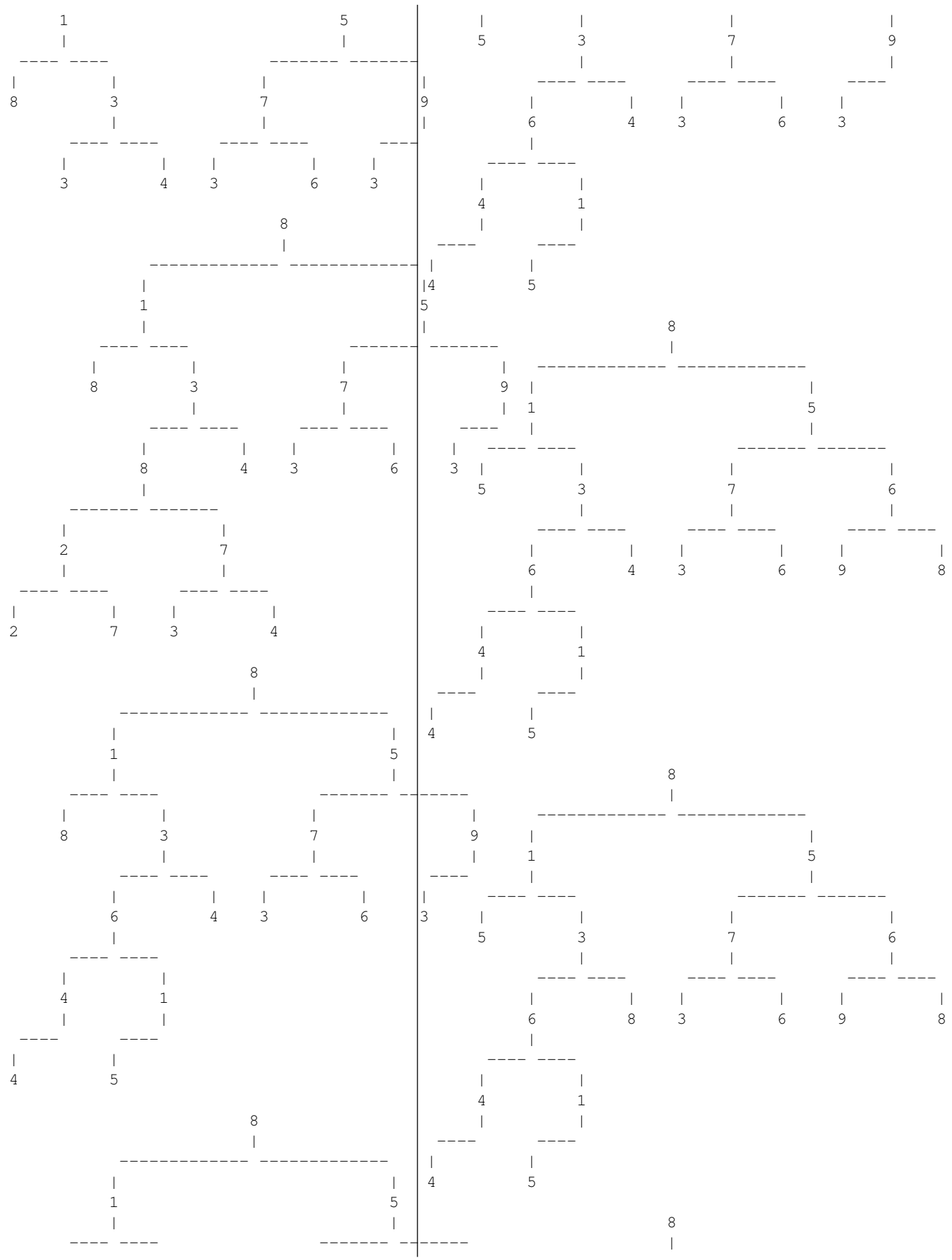


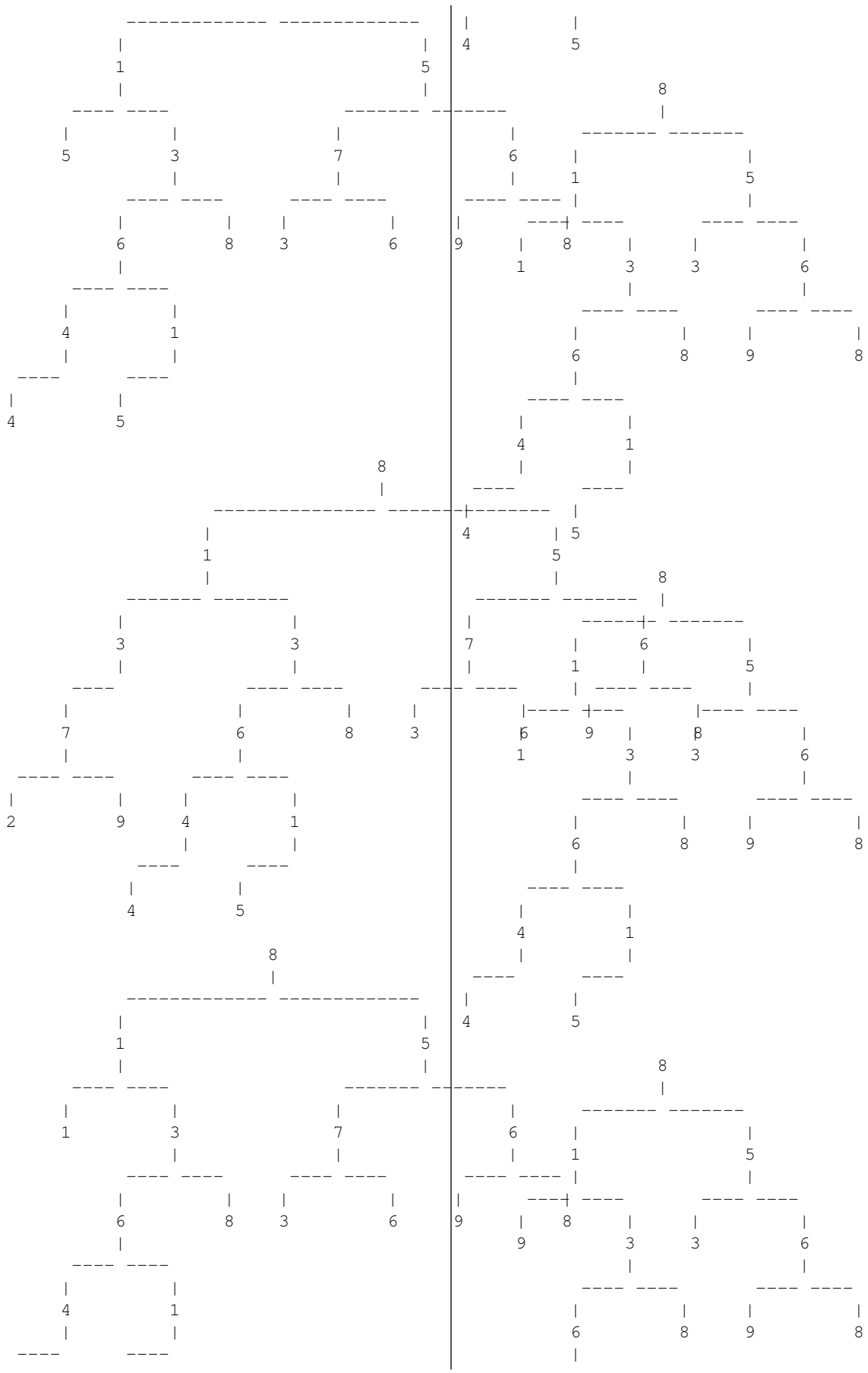


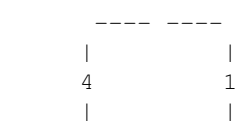












Exemple d'entrada 2

```
INLINEFORMAT
8(6,4)
REPLACE 3
5(6(,1),9(7,))
PRINT
REPLACE 1
6(,4)
PRINT
REPLACE 4
7(3,6)
PRINT
REPLACE 8
9(6,)
PRINT
REPLACE 9
3
PRINT
REPLACE 1
1
PRINT
REPLACE 1
6
PRINT
REPLACE 1
7
PRINT
REPLACE 1
1(8,3(3,4))
PRINT
REPLACE 3
8(2(2,7),7(3,4))
PRINT
REPLACE 6
6(4(4,),1(5,))
PRINT
REPLACE 1
5
PRINT
REPLACE 16
6(9,8)
PRINT
REPLACE 9
8
PRINT
REPLACE 17
8
PRINT
REPLACE 1
3(7(2,9),)
PRINT
REPLACE 4
1
PRINT
REPLACE 12
```

```
3
PRINT
REPLACE 11
3
PRINT
REPLACE 1
9
PRINT
```

Exemple de sortida 2

```
8(6,5(6(,1),9(7,)))
8(6(,4),5(6(,1),9(7,)))
8(6(,4),5(7(3,6),9(7,)))
8(6(,4),5(7(3,6),9(9(6,),)))
8(6(,4),5(7(3,6),9(3,)))
8(1,5(7(3,6),9(3,)))
8(6,5(7(3,6),9(3,)))
8(7,5(7(3,6),9(3,)))
8(1(8,3(3,4)),5(7(3,6),9(3,)))
```

```
8(1(8,3(8(2(2,7),7(3,4)),4)),5(7(3,6),9(3,)))
8(1(8,3(6(4(4,),1(5,)),4)),5(7(3,6),9(3,)))
8(1(5,3(6(4(4,),1(5,)),4)),5(7(3,6),9(3,)))
8(1(5,3(6(4(4,),1(5,)),4)),5(7(3,6),6(9,8)))
8(1(5,3(6(4(4,),1(5,)),8)),5(7(3,6),6(9,8)))
8(1(5,3(6(4(4,),1(5,)),8)),5(7(3,6),6(9,8)))
8(1(3(7(2,9),),3(6(4(4,),1(5,)),8)),5(7(3,6),6(9,8)))
8(1(1,3(6(4(4,),1(5,)),8)),5(7(3,6),6(9,8)))
8(1(1,3(6(4(4,),1(5,)),8)),5(3,6(9,8)))
8(1(1,3(6(4(4,),1(5,)),8)),5(3,6(9,8)))
8(1(9,3(6(4(4,),1(5,)),8)),5(3,6(9,8)))
```

Observació

Heu de treballar amb el tipus `BinTree`, però també podeu utilitzar altres classes vistes durant el curs si ho considereu oportú, tot i que realment no és indispensable.
Avaluació sobre 10 punts:

- Solució lenta: 5 punts.
- solució ràpida: 10 punts.

Entenem com a solució ràpida una que és correcta, de cost proporcional a la mida del primer arbre durant la lectura inicial, de cost proporcional a l'arbre a escriure per a les comandes `PRINT`, i de cost proporcional al camí fins al node a reemplaçar més la mida del subarbre d'entrada per al reemplaçament a les comandes `REPLACE`, 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: 2026-01-25T16:33:07.644Z