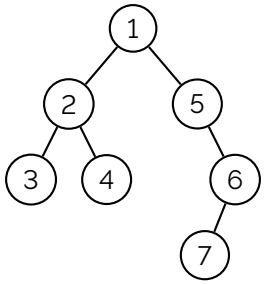
Reconstrucción de un árbol binario

El semestre pasado estuvimos diseñando árboles binarios, con números enteros en los nodos. Los más bonitos eran aquellos que tenían todos los valores distintos, como este:



Cuando acabamos con ellos, guardamos los más bonitos en un archivo, serializados según su recorrido en preorden (ya sabes: raíz, hijo izquierdo, hijo derecho). Por aquello de la tolerancia a fallos, los guardamos en un segundo archivo, pero ahí serializados según su recorrido en inorden (hijo izquierdo, raíz, hijo derecho). *Y imenos mal!* Creemos que con uno solo de los recorridos no habríamos podido ahora reconstruir los árboles y tampoco estamos seguros de poderlo hacer teniendo los dos. ¿Nos ayudas?

Se implementará una función que reciba los recorridos en preorden e inorden de un árbol (por ejemplo en dos vectores) y devuelva un objeto de la clase `BinTree` que represente el árbol que tenga esos dos recorridos.

```
BinTree<int> reconstruir(vector<int> const& preorden, vector<int> const& inorden);
```

¿Cuál es el coste en tiempo de dicha función con respecto a la longitud de los recorridos?

Entrada

La entrada está formada por una serie de casos. Cada caso consta de tres líneas: la primera contiene el número de nodos de un árbol binario; la segunda contiene el recorrido en preorden de ese árbol binario (cuyos nodos tienen valores todos distintos); y la tercera su recorrido en inorden.

Salida

Para cada caso, se escribirá el árbol reconstruido, tal como hace el método `display()`.

Entrada de ejemplo

```
7
1 2 3 4 5 6 7
3 2 4 1 5 7 6
3
2 4 6
2 4 6
8
1 2 4 8 5 3 6 7
8 4 2 5 1 6 3 7
```

Salida de ejemplo

```
(((. 3 .) 2 (. 4 .)) 1 (. 5 ((. 7 .) 6 .)))
(. 2 (. 4 (. 6 .)))
((((. 8 .) 4 .) 2 (. 5 .)) 1 ((. 6 .) 3 (. 7 .)))
```

Autor

Alberto Verdejo