

---

## Haskell — Binary tree

P37072\_en

---

In this problem you have to write several functions for generic binary trees. The definition of the trees is given by:

```
data Tree a = Node a (Tree a) (Tree a) | Empty deriving (Show)
```

That is, a tree with elements of type  $a$  is, either an empty tree, either a node with an element (of type  $a$ ) and two other trees of the same type. The **deriving (Show)** statement simply enables an visualization of trees.

1. Write a function  $size :: Tree a \rightarrow \mathbf{Int}$  that, given a tree, returns its size, that is, the number of node it contains.
2. Write a function  $height :: Tree a \rightarrow \mathbf{Int}$  that, given a tree, returns its height, assuming that empty trees have zero height.
3. Write a function  $equal :: \mathbf{Eq} a \Rightarrow Tree a \rightarrow Tree a \rightarrow \mathbf{Bool}$  that, given two trees, tells whether they are the same.
4. Write a function  $isomorphic :: \mathbf{Eq} a \Rightarrow Tree a \rightarrow Tree a \rightarrow \mathbf{Bool}$  that, given two trees, tells whether they are isomorphic, that is, if one can obtain one from the other flipping some of its descendants.
5. Write a function  $preOrder :: Tree a \rightarrow [a]$  that, given a tree, return its pre-order traversal.
6. Write a function  $postOrder :: Tree a \rightarrow [a]$  that, given a tree, return its post-order traversal.
7. Write a function  $inOrder :: Tree a \rightarrow [a]$  that, given a tree, return its in-order traversal.
8. Write a function  $breadthFirst :: Tree a \rightarrow [a]$  that, given a tree, return its traversal by levels.
9. Write a function  $build :: \mathbf{Eq} a \Rightarrow [a] \rightarrow [a] \rightarrow Tree a$  that, given a pre-order traversal of a tree and an in-order traversal of the same tree, returns the original tree. You can assume that the three has no repeated elements.
10. Write a function  $overlap :: (a \rightarrow a \rightarrow a) \rightarrow Tree a \rightarrow Tree a \rightarrow Tree a$  that, given two trees, returns its overlapping using a function. Overlapping two trees with a function consists in placing the two trees one on the other and combine the double nodes using the given function.

### Scoring

Each function scores 10 points.

## Sample input

```
let t7 = Node 7 Empty Empty
let t6 = Node 6 Empty Empty
let t5 = Node 5 Empty Empty
let t4 = Node 4 Empty Empty
let t3 = Node 3 t6 t7
let t2 = Node 2 t4 t5
let t1 = Node 1 t2 t3
let t1' = Node 1 t3 t2
size t1
height t1
equal t2 t3
isomorphic t1 t1'
preOrder t1
postOrder t1
inOrder t1
breadthFirst t1
build [1,2,4,5,3] [4,2,5,1,3]
overlap (+) t2 t3
overlap (+) t1 t3
```

## Sample output

```
7
3
False
True
[1,2,4,5,3,6,7]
[4,5,2,6,7,3,1]
[4,2,5,1,6,3,7]
[1,2,3,4,5,6,7]
Node 1 (Node 2 (Node 4 Empty Empty) (Node 5 Empty Empty)) (Node 3 Empty Empty)
Node 5 (Node 10 Empty Empty) (Node 12 Empty Empty)
Node 4 (Node 8 (Node 4 Empty Empty) (Node 5 Empty Empty)) (Node 10 (Node 6 Empty Empty) (Node 7 Empty Empty))
```

## Problem information

Author : Jordi Petit

Translator : Jordi Petit

Generation : 2023-03-16 16:03:05

© *Jutge.org*, 2006–2023.

<https://jutge.org>