# Declaratieve Talen

Haskell 2

## 1 Tree Folds

### 1.1 Defining a tree

Given below is a definition for a binary tree. Be sure to include the `deriving (Show, Eq)` construct to generate the right typeclasses.

```
data Tree a = Leaf a | Fork (Tree a) (Tree a)
  deriving (Show, Eq)
```

- Just like lists we can define a fold over trees. Define a function `foldTree :: (a -> b) -> (b -> b -> b) -> (Tree a -> b)` that performs this folding.

### 1.2 Folding trees

Using `foldTree`, define the following functions.

- A function `sumTree :: Tree Int -> Int` that sums the integers stored at the leafs of a tree.
- A function `treeToList :: Tree a -> [a]` that converts a tree to a list.
- A function `nrOfLeaves :: Tree a -> Int` that counts the number of leaves in a tree.
- A function `depthOfTree :: Tree a -> Int` that calculates the maximum depth of the tree.
- A function `mirrorTree :: Tree a -> Tree a` that mirrors all subtrees.
- A function `minTree :: Tree Int -> Int` that returns the smallest integer stored at any of the leaves.
- A function `addOne :: Tree a -> Tree Int` that adds one to each integer stored at the leaves of a tree.
- If given the right functions, `foldTree` can reconstruct the original tree. Define a function `idTree :: Tree a -> Tree a` that performs this reconstruction.

**Examples**

```
Main> sumTree (Leaf 1)
1
Main> sumTree (Fork (Leaf 1) (Fork (Leaf 2) (Leaf 3)))
6

Main> treeToList (Leaf 1)
[1]
Main> treeToList (Fork (Leaf 1) (Fork (Leaf 2) (Leaf 3)))
[1,2,3]

Main> nrOfLeaves (Leaf 1)
1
Main> nrOfLeaves (Fork (Leaf 1) (Fork (Leaf 2) (Leaf 3)))
3

Main> depthOfTree (Leaf 1)
1
Main> depthOfTree (Fork (Leaf 1) (Fork (Leaf 2) (Leaf 3)))
3

Main> mirrorTree (Leaf 1)
Leaf 1
Main> mirrorTree (Fork (Leaf 1) (Fork (Leaf 2) (Leaf 3)))
Fork (Fork (Leaf 3) (Leaf 2)) (Leaf 1)

Main> minTree (Leaf 1)
1
Main> minTree (Fork (Fork (Leaf 20) (Leaf 30)) (Leaf 10) )
10

Main> addOne (Leaf 1)
Leaf 2
Main> addOne (Fork (Leaf 1) (Fork (Leaf 2) (Leaf 3)))
Fork (Leaf 2) (Fork (Leaf 3) (Leaf 4))

Main> idTree (Leaf 1)
Leaf 1
Main> idTree (Fork (Leaf 1) (Fork (Leaf 2) (Leaf 3)))
Fork (Leaf 1) (Fork (Leaf 2) (Leaf 3))
```