HW 16 **Prof. Caldwell**
Due: 2 November 2010 **COSC 3015**

This is an exercise in some rather complex recursion on Rose trees and Binary trees. You will implement the mapping (similar to the one described in class) of Rose trees to Binary Trees and the inverse mapping binary trees to rose trees.

```
data Rose a = Node a [Rose a] deriving (Eq,Show)
data BTree a = Empty | Fork a (BTree a) (BTree a) deriving (Eq,Show)
```

The mapping is called the *natural correspondence* between trees and binary trees [1, pp.332]. Given a rose tree (with arbitrary but finite branching) you construct a binary tree where the left fork for each node in the binary tree is the translation of the leftmost child of the corresponding node in the rose tree and the right fork is the binary tree resulting form translating the right sibling of the node in the translated rose tree. Since root nodes in rose trees have no siblings, the binary trees resulting from the translation always have the form `(Fork x t Empty)`.

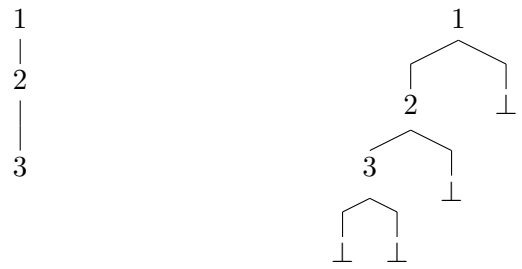
Here are some examples:

Example 0.1.

`(Rose 1 [Rose 2 []])` $\xrightarrow{nat}$ `Fork 1 (Fork 2 Empty Empty) Empty`

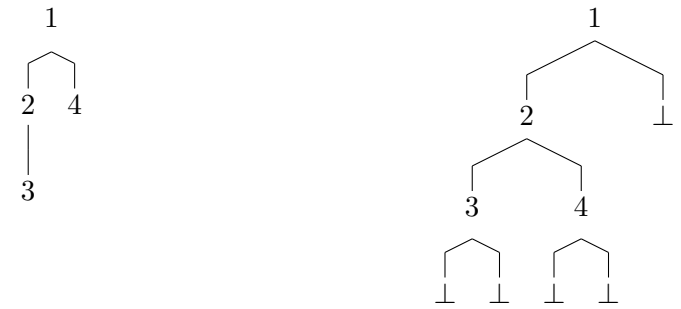
Example 0.2.

`(Node 1 [Node 2 [Node 3 []]])` $\xrightarrow{nat}$ `Fork 1 (Fork 2 (Fork 3 Empty Empty) Empty) Empty`



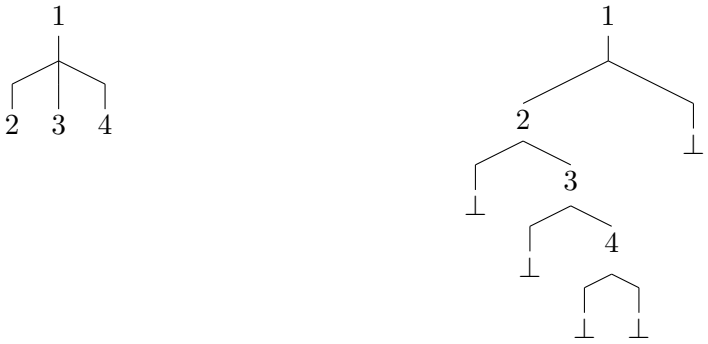
Example 0.3.

```
(Node 1 [Node 2 [Node 3 []],Node 4 []]) ⟶nat
    Fork 1 (Fork 2 (Fork 3 Empty Empty) (Fork 4 Empty Empty)) Empty
```



Example 0.4.

```
Rose 1 [Rose 2 [],Rose 3 [],Rose 4 []] ⟶nat
    Fork 1 (Fork 2 Empty (Fork 3 Empty (Fork 4 Empty Empty))) Empty
```



Exercise 0.1. Write a function `nat::Rose a → Btree a` that implements the natural transformation by recursion on the structure of rose trees.

Exercise 0.2. Write a function `inv::Btree a → Rose a` that implements the inverse mapping.

References

[1] Donald Knuth. *The Art of Computer Programming: Fundamental Algorithms*, volume 1. Addison Wesley, Reading, MA, 1973.