Functional Programming - Lab Class Exercises 4

Frank C Langbein
frank@langbein.org

Version 1.4.0

1. Trees

Consider the follow two tree data structures. `BTree` defines a binary tree with all values at the leaf nodes. `CTree` defines a more general tree where the values are stored at the nodes and the children in a list.

```
data BTree a = BLeaf a
             | BNode (BTree a) (BTree a)    deriving (Show)
data CTree a = CNode a [CTree a]    deriving(Show)
```

For each data structure write a function that converts the tree into a comma-separated list of in-order values. For `CTree` assume the in-order traversal splits the tree after the first child. Note `show` converts a value into a `String`. The signatures of these functions are

```
btree_string :: (Show a) => BTree a -> String
ctree_string :: (Show a) => CTree a -> String
```

2. Functors, Applicatives and Monads for Trees

For the two tree data structures defined in 1. create a `Functor`, `Applicative` and `Monad` instance. Carefully consider what these operations could be and note that for some there may not be a useful/any answer.

Multiple answers here may be possible. This is an advanced question to further explore these concepts.