2006 Paper 11 Question 12

Introduction to Functional Programming

Let `datatype 'a bintree = empty | node of 'a * 'a bintree * 'a bintree` be the polymorphic data type representing binary trees.

(*a*) Define the curried function `treemap` that given an `'a -> 'b` function and then an `'a bintree` produces a `'b bintree` of exactly the same shape as the given one but with all nodes having been *mapped* by the function. [4 marks]

(*b*) Use `treemap` to define the function `lift` that converts an `'a bintree` into an `'a option bintree`. Further use `treemap` to define the curried functional `prefilter` that given a predicate `'a -> bool` and then an `'a bintree` returns an `'a option bintree` in which the information on the nodes that do satisfy the predicate is omitted. [3 marks]

(*c*) Define the function

```
('a, 'b) DFfoldRtoL : ('a * 'b -> 'b) -> 'b -> 'a bintree -> 'b
```

that *folds* a binary tree (using its second input as base case and its first input in each recursive step) as the tree is traversed in a *depth-first* manner from *right to left*.

Your definition should be such that the function

```
    fun inorder t = DFfoldRtoL op:: [] t
```

lists a binary tree in *infix order*. [5 marks]

(*d*) Use `DFfoldRtoL` to define the function

```
    'a inorderData : 'a option bintree -> 'a list
```

that lists *the data* in its input tree in infix order. [2 marks]

(*e*) Define a function

```
    'a cleanup : 'a option bintree -> 'a bintree
```

satisfying the specification `inorder`(`cleanup` t) $=$ `inorderData`(t) for all types α and values t of type α `option bintree` and such that `cleanup`(`lift` t) $=$ t for all types α and values t of type α `bintree`. [6 marks]

Note that `cleanup` and `prefilter` can be composed to yield a *tree-filter* functional that preserves the infix order.