2000 Paper 13 Question 11

Introduction to Functional Programming

The following is a recursive definition of a datatype `ltree`, which is intended to represent binary trees in which data is stored only at the leaves, not at internal nodes.

```
datatype 'a ltree = Empty
                  | Leaf of 'a
                  | Branch of ('a ltree) * ('a ltree);
```

(a) Write a simple recursive function

`elems: ('a ltree) -> ('a list)`

which gives a list of the data elements stored in a tree. [4 marks]

(b) Write an iterative version of this function

`elemsi: ('a ltree * 'a list) -> ('a list)`

which does not require appending of lists, and which satisfies the equality:

$$\texttt{elemsi}(t, l) = \texttt{elems}(t)@l$$

You do not have to prove the equality. [6 marks]

(c) Given the datatype of sequences:

```
datatype 'a seq = Nil
                | Cons of 'a * (unit -> 'a seq)
```

write a function `appendq:(('a seq)*('a seq))-> ('a seq)` for appending two sequences. [4 marks]

Use this to define a function `elemsq:('a ltree)-> ('a seq)` which, given a tree, produces a lazy list of the data elements stored in it. [6 marks]