

2007 Paper 11 Question 11

Introduction to Functional Programming

(a) Sorting.

(i) Write a parametric sorting SML function

`sort: (α * α \rightarrow bool) \rightarrow α list \rightarrow α list` [3 marks]

(ii) Write an SML function

`lex: (α * α \rightarrow bool) \rightarrow (α list * α list) \rightarrow bool`

that given as input a comparison function c returns as output a comparison function (`lex c`) that would be useful for sorting lists into lexicographical (or alphabetical) order. [2 marks]

(iii) Write a parametric lexicographic list-sorting SML function

`lexsort: (α * α \rightarrow bool) \rightarrow α list list \rightarrow α list list` [1 mark]

(b) Let

`datatype α tree = empty | node of α * α tree list`
`type α forest = α tree list`

respectively be the types of finitely-branching trees and forests.

As usual, the *maximal paths* of a tree are given by the lists of consecutive nodes from the root to a leaf (empty tree). The *trace* of a tree is the list of all its maximal paths, and the trace of a forest that of all its trees.

(i) Write an SML function `trace: α forest \rightarrow α list list` that outputs the trace of a forest according to a depth-first traversal. [3 marks]

Write an SML function `mkf: α list \rightarrow α forest` that outputs a forest whose trace consists only of the input list. [2 marks]

(ii) The *paths* of a forest are given by lists of consecutive nodes from a root to any other node. A *trie* (or *prefix forest*) is a forest without repeated paths.

Write an SML function `add: α list \rightarrow α forest \rightarrow α forest` that, given as input a list ℓ and then a trie t , returns as output the trie resulting from adding ℓ to t ; in the sense that the trace of (`add ℓ t`) is the trace of t together with ℓ . [4 marks]

Write a parametric trie-sorting SML function

`triesort: (α * α \rightarrow bool) \rightarrow α forest \rightarrow α forest`

such that `trace(triesort c t) = lexsort c (trace t)` for all comparison functions c and tries t . [5 marks]