1 Foundations of Computer Science (LCP)

This question has been translated from Standard ML to OCaml

Recall that a dictionary of \((key, value)\) pairs can be represented by a binary search tree. Define the \textit{union} of two binary search trees to be any binary search tree consisting of every node of the given trees.

\(a\) Write an OCaml function \texttt{union} to return the union of two given binary search trees. \([\text{Note: You may assume that they have no keys in common.}]\) \([6 \text{ marks}]\)

Define a \textit{slice} of a binary search tree to be a binary search tree containing every \((key, value)\) node from the original tree such that \(x \leq key \leq y\), where \(x\) and \(y\) are the given endpoints.

\(b\) Write an OCaml function \texttt{takeSlice} to return a slice – specified by a given pair of endpoints – from a binary search tree. \([4 \text{ marks}]\)

\(c\) Write an OCaml function \texttt{dropSlice} to \textit{remove} a slice from a binary search tree: given a tree and a pair of endpoints, it should return the binary search tree consisting of precisely the nodes such that \(x > key\) or \(key > y\). \([\text{Hint: First consider the simpler task of deleting a node from a binary search tree.}]\) \([8 \text{ marks}]\)

\(d\) The tree \(t\) need not be identical to that returned by

\[
\text{union (takeSlice (x, y) t)} \\
\text{(dropSlice (x, y) t)}
\]

Briefly explain how such an outcome is possible. \([2 \text{ marks}]\)

\([\text{Note: All OCaml code must be explained clearly and should be free of needless complexity.}]\)