

Worth: 10% (together with part II of this assignment)

Due: By 1pm on Wednesday **25 March**

For this part of the assignment, you will implement ML functions for a mutable binary search tree of strings, *i.e.*, a binary search tree that stores references to strings, so the values can be updated “in place”. The basic datatype you must use for such trees is the following:

```
datatype strBST = Leaf | Node of string ref * strBST ref * strBST ref;
```

1. Write a function `search = fn : strBST ref -> string -> bool` such that for all trees (given by their root `T : strBST ref`) and for all strings `s : string`, the value of `search T s` is `true` if the tree rooted at `!T` contains some node that stores a string equal to `s`; `false` otherwise.
2. Write a function `insert = fn : strBST ref -> string -> unit` such that for all trees (given by their root `T : strBST ref`) and for all strings `s : string`, the call `insert T s` inserts `s` into the tree rooted at `!T` (making no change if `s` already belongs to the tree)—*i.e.*, `insert T s` modifies the data in `T` to ensure that the tree rooted at `!T` contains one node that stores a string equal to `s`.
3. Write a function `remove = fn : strBST ref -> string -> unit` such that for all trees (given by their root `T : strBST ref`) and for all strings `s : string`, the call `remove T s` removes `s` from the tree rooted at `!T` (making no change if `s` does not belong to the tree)—*i.e.*, `remove T s` modifies the data in `T` to ensure that the tree rooted at `!T` contains no node that stores a string equal to `s`.
4. Write a function `strBST2string = fn : string -> strBST ref -> string` such that for all trees (given by their root `T : strBST ref`) and strings `s : string`, the value of `strBST2string s T` is a string representation of the tree rooted at `!T`, drawn “sideways” (*i.e.*, with right subtrees above their root node and left subtrees below), with an indentation in front of each node’s string equal to exactly `d` copies of `s`, where `d` is the node’s depth in the tree (with the root at depth 0).

For example, your functions should produce the following results.

<pre>- val T = ref Leaf; val T = ref Leaf : strBST ref - insert T "hello"; val it = () : unit - insert T "blah"; val it = () : unit - insert T "hi"; val it = () : unit - insert T "bye"; val it = () : unit - print (strBST2string " " T); hi hello bye blah val it = () : unit</pre>	<pre>- insert T "hi"; val it = () : unit - print (strBST2string " " T); hi hello bye blah val it = () : unit - remove T "hello"; val it = () : unit - print (strBST2string " " T); hi bye blah val it = () : unit</pre>
--	---

As for the previous assignment, you must use good functional style in your code (*i.e.*, make appropriate use of patterns, local declarations, and higher-order functions), including good external and internal comments (to explain the purpose of your code and your decisions on how to implement it), as well as good visual layout (formatting, indenting, blank lines) to make your code easy to read and understand.

Submit your code for this part in file “`strBST.sml`” (include the code for `datatype strBST` in your file).