# 99 questions/Solutions/73

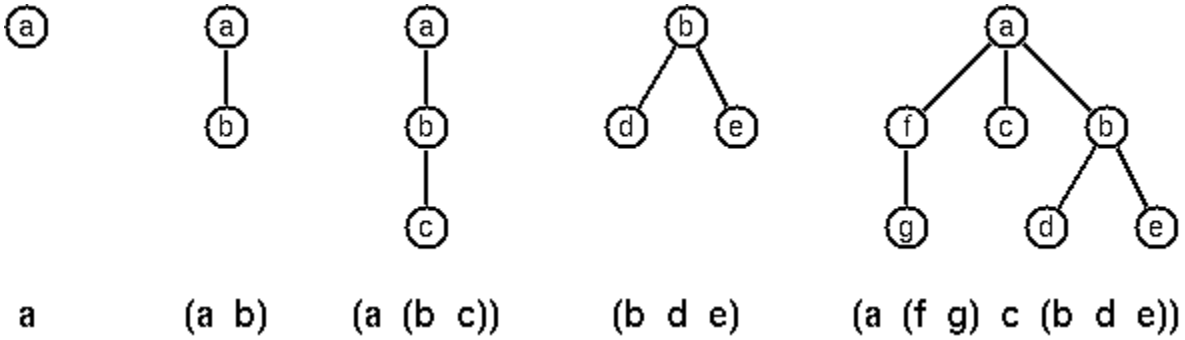
## From HaskellWiki

< 99 questions | Solutions

(**) Lisp-like tree representation.

There is a particular notation for multiway trees in Lisp. Lisp is a prominent functional programming language, which is used primarily for artificial intelligence problems. As such it is one of the main competitors of Prolog. In Lisp almost everything is a list, just as in Prolog everything is a term.

The following pictures show how multiway tree structures are represented in Lisp.




Note that in the "lispy" notation a node with successors (children) in the tree is always the first element in a list, followed by its children. The "lispy" representation of a multiway tree is a sequence of atoms and parentheses '(' and ')', which we shall collectively call "tokens". We can represent this sequence of tokens as a Prolog list; e.g. the lispy expression (a (b c)) could be represented as the Prolog list ['(', a, '(', b, c, ')', ')']. Write a predicate tree_ltl(T,LTL) which constructs the "lispy token list" LTL if the tree is given as term T in the usual Prolog notation.

Solution using the `Syntax` type used in problem 70 above:

```
lisp :: Syntax (Tree Char)
lisp = iso toTree fromTree
        (literal '(' *> (char <*> list (space *> lisp) (literal ')')) <|> char)
  where toTree (Left (x, ts)) = Node x ts
        toTree (Right x) = Node x []
        fromTree (Node x []) = Right x
        fromTree (Node x ts) = Left (x, ts)
```

Here we use the isomorphism between `Tree a` and `Either (a, [Tree a]) a`, where the

list is non-empty.