— *Solution notes* —

COMPUTER SCIENCE TRIPOS Part IA – 2013 – Paper 1

1 Foundations of Computer Science (LCP)

datatypes, pattern-matching

(*a*) Write brief notes on ML datatypes and pattern-matching in function declarations. [6 marks]

Answer: Solutions should include examples of datatype declarations and mention the concept of a constructor. Examples of pattern-matching should be non-trivial, with nested constructors and (preferably) overlapping patterns.

---

programming, binary trees

(*b*) A binary tree is either a *leaf* (containing no information) or is a *branch* containing a label and two subtrees (called the *left* and *right* subtrees). Write ML code for a function that takes a label and two lists of trees, returning all trees that consist of a branch with the given label, with the left subtree taken from the first list of trees and the right subtree taken from the second list of trees. [6 marks]

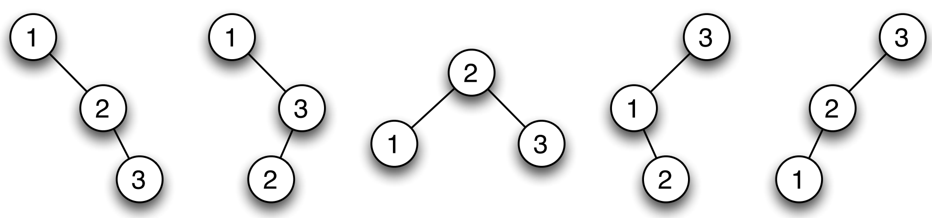
Answer: The datatype declaration is not required as part of the answer, but sets the stage. Students are unlikely to know about `List.concat`, but it can be coded in two lines with the help of @ (append).

```
datatype 'a tree = Lf
                 | Br of 'a * 'a tree * 'a tree;

fun make_trees v t1 = map (fn t2 => Br(v, t1, t2));

fun make_trees2 v t1s t2s =
      List.concat (map (fn t1 => make_trees v t1 t2s) t1s);
```

---

programming, binary trees

(*c*) Write ML code for a function that, given a list of distinct values, returns a list of all possible binary trees whose labels, enumerated in inorder, match that list. For example, given the list [1,2,3] your function should return (in any order) the following list of trees:



[8 marks]

Answer:

```
fun anti (l1, []) = []
  | anti (l1, v::l2) =
      make_trees2 v (anti_inorder (rev l1)) (anti_inorder l2) @
      anti (v::l1, l2)
```

```
and anti_inorder [] = [Lf]
  | anti_inorder xs = anti ([],xs);
```

Note that the question refers to binary trees, not to binary *search* trees, and it does not impose an ordering constraint on the labels of these trees.

All ML code must be explained clearly and should be free of needless complexity.