1999 p1q1

Foundations of Computer Science (Small Questions)

~~**Tiny questions (Paper 2)**~~ p2q1

~~1. Consider the ML lists (`fn x => [x,x]`)(`ref 0`) and `map ref [0,0]`. What do they have in common, and how do they differ?~~

~~2. What does it mean for a function to be *tail-recursive*?~~

~~3. How can an ML program form a cyclic list? (Your answer need not present any code.)~~

~~4. Outline the key principles of loop design, considering the simple case of a `while` loop.~~

10-mark question (Paper 1) p1q1

A *permutation* of a list is any list that can be derived from it by re-arranging the elements. Write an ML function that returns the list of all the permutations of its argument. Explain your code clearly and carefully.

For example, applied to the list [1,2,3], your function should return the list whose elements are [1,2,3], [2,1,3], [2,3,1], [1,3,2], [3,1,2] and [3,2,1].

You may assume that the elements of the argument are distinct. The elements of the result may appear in any order.

Model Answers

~~**Tiny questions**~~ p2q1

~~1. Each list consists of two reference cells that are initialized to zero. But the first list holds two identical reference cells, while the second holds two distinct ones. Updating the head of the first list would also update its second element.~~

~~2. (From the notes.) A function is tail-recursive if evaluating the recursion does not result in nesting. The effect is to save space (on the stack) compared with a similar function that is not tail-recursive.~~

~~3. A cyclic list can only form if the program defines its own mutable list type using type `ref`. The only way to form a cycle is by updating an existing `ref` cell, e.g. by using a destructive concatenate function to join a list to itself.~~

~~4. Loop design centres on an invariant, a variant and a termination condition. The *invariant* is a property that should hold upon entry and be preserved at each iteration; in particular it holds upon termination, along with the termination condition itself. The variant is typically an expression that denotes a positive integer and that decreases with each iteration; this ensures that the loop will eventually terminate.~~

10-mark question p1q1

See the attached code. The empty list has only one permutation: itself. For a non-empty list, the function removes the head and calls its recursively on the tail, inserting the head in each permutation in every possible position.

```
fun insertall x front [] = [rev (x::front)]
  | insertall x front (y::ys) =
        (rev front @ (x::y::ys)) :: insertall x (y::front) ys;

fun flat [] = []
  | flat (l::ls) = l @ flat ls;

fun perms [] = [[]]
  | perms (x::xs) = flat (map (insertall x []) (perms xs));
```