SOLUTION NOTES

Foundations of Computer Science 2003 Paper 1 Question 5 (LCP)

This question assesses basic ML programming skills, and specifically Lecture 13 (Lazy Lists).

(a) Here is the datatype of lazy lists:

```
datatype 'a seq = Nil
                | Cons of 'a * (unit -> 'a seq)
```

A function represents the list's tail. Thus, it is not evaluated until its value is required, so some infinite lists can be expressed.

(b) and (c) These functions was given in Lecture 13 and should be easy for candidates to reconstruct anyway. If the first argument to `appendq` is an infinite list then the result is just the first list.

```
fun appendq (Nil,    yq) = yq
  | appendq (Cons(x,xf), yq) = Cons(x, fn()=>appendq(xf(),yq))

fun interleave (Nil,    yq) = yq
  | interleave (Cons(x,xf), yq) =
        Cons(x, fn()=> interleave(yq, xf()))
```

(d) The code is brief, but perhaps hard to find. Given the list `xs`, we interleave the result of attaching a zero with that of attaching a one. Now `binaryLists []` is the solution.

```
fun binaryLists xs =
    Cons(xs, fn()=> interleave(binaryLists(0::xs),
                               binaryLists(1::xs)))
```

(e) This function takes the output of part (d) and generates palindromes by the following ideas. An even palindrome consists of a list joined to its reverse. An odd palindrome consists of a list joined to its reverse about a centre element, which is either zero or one.

```
fun palify (Cons(x,xf)) =
    Cons(x @ rev x,
         fn()=> Cons(x @ (0 :: rev x),
                     fn()=> Cons(x @ (1 :: rev x),
                                 fn()=> palify (xf()))));
```

An alternative solution is to take the output of part (d) and simply discard all non-palindromes. This solution will get slightly less credit, because it is highly inefficient: exponentially many elements will be discarded.