SOLUTION NOTES

Introduction to Functional Programming 2001 Paper 13 Question 11 (AD)

```
fun append [] l2   = l2
  | append h::t l2 = h::(append t l2);
```

[3 marks]

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

[3 marks]

```
fun applistq [] s   = s
  | applistq h::t s = Cons(h, fn () => applistq(t, s));
```

[6 marks]

To prove,

```
applistq l1 (applistq l2 s) = applistq (l1@l2) s
```

We use structural induction on the first list $l1$, to prove the stronger induction hypothesis:

$\forall l2 \forall s$ `applistq l1 (applistq l2 s) = applistq (l1@l2) s`

In the base case:

```
applistq [] (applistq l2 s)
       = applistq l2 s
       = applistq []@l2 s
```

where the first line is true from the base case in the definition of `applistq`, and the second line follows from the base case of the definition of `append`.

Induction Step:

```
applistq h::t (applistq l2 s)
          = Cons(h, fn () => applistq (t, applistq l2 s))
          = Cons(h, fn () => applistq t@l2 s)
          = applistq h::(t@l2) s
          = applistq (h::t)@l2 s
```

where the first identity follows from the recursive step in the definition of `applistq`. The second line is an application of the induction hypothesis. The third line also follows from the recursive step in the definition of `applistq` (this time applied in reverse), and the final line follows from the definition of `append`.

[8 marks]