Solution notes

Foundations of Functional Programming 2005 – Paper 5 Question 12 (ACN)

(*a*) *Give a lambda-expression that can be used to form the composition of two functions.* *[1 mark]*

$$\lambda f.\lambda g.\lambda x.f(gx)$$

(*b*) *Suppose that the lambda expression you have given above can be referred to using the name* `B`*, one way of representing the natural numbers as lambda expressions involves for instance having the number "3" represented by a term* $\lambda f.\mathtt{B}f(\mathtt{B}ff)$ *so that a numeral when applied to an argument* f *composes* f *with itself the given number of times.*

In this scheme, write out lambda-expressions that will serve as 0, 1 and 2. *[3 marks]*

$$\lambda f.\lambda x.x$$

$$\lambda f.f$$

$$\lambda f.Bff$$

(*c*) *Present and explain lambda-expressions that find the successor to a number and that add two numbers together.* *[6 marks]*

$$\lambda m.\lambda f.Bf(mf)$$

$$\lambda m.\lambda n.B(mf)(nf)$$

A simple trace giving these terms sample args will show why they work.

(*d*) *If* m *and* n *are two lambda expressions that both represent numbers in this style, what interpretation can be places on the term* $(m\ n)$*? Explain and justify your claim.* *[4 marks]*

The term mn will behave as the number corresponding to n^m. This is because mn is the same as $n \circ n \circ \ldots \circ n$ with m copies of n composed. Each application of n multiplies by n, so we do that m times and get n^m.

(*e*) *Explain how it is possible to produce a lambda expression that, give then representation of a non-zero number* k *produces an expression that behaves like* $k - 1$. *[6 marks]*

The idea here is to try to manage *n*successorminusone and find something that will act as the "minusone" there. Earlier we had

```
fun suc n f = B f (n f);
```

Here I will introduce

```
fun suc1 n f = n (K (B f (n f))) 1;
```

as a "trick" version of successor that applies its arg. If the arg is zero it returns 1. If it is non-zero it returns the successor of its arg. But by its construction it is now trivial to invent an arg such that when handed to `suc1` you get 0.

The sample script has an alternative way.