COMPUTER SCIENCE TRIPOS Part IA – 2021 – Paper 1

1 Foundations of Computer Science (jdy22)

Sequences (lazy lists) and trees are fundamental types in functional programming. Here are definitions of sequences and trees with integer elements:

```
type iseq = Nil
          | Cons of int * (unit -> iseq)

type itree = Leaf of int
           | Branch of itree * itree
```

(*a*) In an *ascending* sequence such as $1, 3, 3, 7, \ldots$ each element is at least as large as the previous elements.

Given two ascending sequences, write a function `merge2` that produces a sequence of the elements of both in ascending order. For example, passing $1, 3, 3, 7, \ldots$ and $2, 4, 5, 9, \ldots$ to `merge2` should produce the sequence $1, 2, 3, 3, 4, 5, 7, 9, \ldots$. [5 marks]

(*b*) Sequences are considered to be equal if corresponding elements are equal.

(*i*) Define a function `equal_seq` that compares two sequences for equality. [5 marks]

(*ii*) Define sequences `s1` and `s2` for which `equal_seq s1 s2` does not terminate. [3 marks]

(*c*) The *fringe* of a tree is the left-to-right sequence of the values at the leaves. For example, the fringe of `Branch (Leaf 3, Branch (Leaf 10, Leaf 4))` is the sequence $3, 10, 4$.

(*i*) Define a function `fringe` that computes the fringe of a tree. Your function should have the following type:

```
val fringe : itree -> iseq
```

[5 marks]

(*ii*) Using the functions you have defined above or otherwise, write a function `equal_fringes` that determines whether two trees have equal fringes. [2 marks]