Solution notes

Foundations of Computer Science 2005 – Paper 1 Question 6 (LCP)

(*a*) *(Lecture 7, datatypes and exceptions.)*

```
fun sum [] = 0 : int
  | sum (k::ks) = k + sum ks;

exception Fail;
fun find_path p t =
  let fun find vs (Twig v) = if p (v + sum vs) then rev (v::vs)
                             else raise Fail
    | find vs (Br(v,t1,t2)) =
        find (v::vs) t1
        handle Fail => find (v::vs) t2
  in find [] t
  end;
```

(*b*) *(Lectures 4–5, lists, and 10, functions as values.)*

```
fun all_paths p t =
  let fun find vs (Twig v) = if p (v + sum vs) then [rev (v::vs)] else []
    | find vs (Br(v,t1,t2)) =
        (find (v::vs) t1) @ (find (v::vs) t2)
  in find [] t
  end;
```

(*c*) *(Lecture 13, lazy lists.)*

The accumulating argument has to be a function in order for laziness to work.

```
fun all_pathq p t =
  let fun find vs (Twig v) xf =
           if p (v + sum vs) then Cons(rev (v::vs), xf) else xf()
    | find vs (Br(v,t1,t2)) xf =
        find (v::vs) t1 (fn () => find (v::vs) t2 xf)
  in find [] t (fn () => Nil)
  end;
```