Harry Langford
hjel2@cam.ac.uk

Paper 1 Question 1

(a) The findSix program counts the number of sixes which are reachable from the root without passing any other sixes. It returns unit if it can find none and raises an exception gotIt(n) if it finds n $\neq$ 0 6's. It has a type of tree $\rightarrow$ unit.

```
let t1 = Br( 4, Lf 6 , Br(6, Lf 3 , Lf 6) );;
let t2 = Br( 4, Lf 7 , Br(8, Lf 3 , Lf 1) );;
```

When run on the first tree, it checks 4. Realises that 4 $\neq$ 6. So it recursively calls itself on the left subtree – Lf 6. This raises GotIt(1). So v1 = 1. findSix is then called on the right subtree. v = 6 so this raises gotIt(1). So v2 = 1. v = v1 + v2 = 2. So the first call raises GotIt(2).

When run on the second tree, it checks 4. 4 $\neq$ 6. So findSix is called on (Lf 7). 7 $\neq$ 6. So () is returned. v1 = 0. findSix is then called on the right subtree – Br(8, Lf 3 , Lf 1). 8 $\neq$ 6. So findSix is called on both subtrees. 3 $\neq$ 6. 1 $\neq$ 6. So both return (). This means findSix Br(8, Lf 3 , Lf 1) returns (). Hence the whole function returns ().

(b)

```
let rec compute n = function
        | Lf(x) -> if x = 6 then 1 else 0
        | Br(v, l, r) -> if v = 6 then 1
                         else
                         let v1 = compute n l in
                         let v2 = compute n r in
                         v1 + v2
;;
```

(c)

```
let containsroot t =
    let rec predicate p = function
        | Lf(x) -> p x
        | Br(v, l, r) -> p v || predicate p l || predicate p r
    in
    match t with
        | Lf(_) -> false
        | Br(root, l, r) -> predicate ((=) root) l || predicate ((=) root) r
;;
```

(d)

```
let doublepath t =
    let rec contains x = function
        | [] -> false
        | hd::tl -> (x = hd) || contains x tl
    in
    let rec twoinpath acc = function
        | Lf(v) -> contains v acc
        | Br(v, l, r) -> (contains v acc) || (twoinpath (v::acc) l) ||
                                             (twoinpath (v::acc) r)
    in
    twoinpath [] t
;;
```