CMSC 330 Quiz 2 Fall 2022

Q1. OCaml Typing

Q1.1. Write an expression of the following type: `float -> int -> float`

```
fun a b -> a +. float_of_int b
```

Q1.2. Write an expression of the following type: `'a -> 'b -> 'c -> ('a -> 'c -> 'b list) -> 'b list`

```
fun w x y z -> x::(z w y)
```

Q2. Type Check

The following expression does not type check:

```
fun f a b -> if a+1=2 then a else if 3 then b+.1.0 else (f b)
```

Identify the type error(s):

Unbound variables, Mismatched return types, Incorrect type for the if condition, Mismatched types when applying b to f

Q3. OCaml Coding

Consider the following type:

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

Now consider the following functions:

```
let rec fun_a acc t =
  match t with
  | Leaf ->
      (match acc with
       | (s, [])     -> acc
       | (s, t::ts) -> fun_a (s,ts) t)
  | Branch(l,v,r) ->
      (match acc with
```

```
        | (s, ts) -> fun_a (v+s, r::ts) l)

let rec fun_b acc t =
  match t with
  | Leaf -> acc
  | Branch(l,v,r) ->
      let l_fun = fun_b acc l in
      fun_b (l_fun + v) r
```

Which functions have *all* of the recursive calls in a tail position?

fun_a, fun_b

Q3. Fill In The Blanks

Given the following `collapse_tree`, type `tree` where it has `int`, `left_tree`, `right_tree` as tree data structure. Implement a function called `biggest_Node` that finds the largest value in the tree.

```
type tree =
  | Leaf of int
  | Node of int * tree * tree

let rec collapse_tree f t =
  match t with
  | Leaf(x) -> x
  | Node (i, l, r) -> f i (collapse_tree f l) (collapse_tree f r)
```

Make sure to thoroughly read and understand `collapse_tree` before implementing the function. The two blanks below refer to the parameters passed in for the `collapse_tree` function.

Example:

```
biggest_Node (Node(8, Node(4, Leaf(1), Leaf(2)), Node(6, Leaf(7), Leaf(6))))
= 8

biggest_Node (Node(4, Node(6, Leaf(2), Leaf(3)), Node(7, Leaf(5), Leaf(6))))
= 7

biggest_Node (Node(6, Node(4, Node(2, Leaf(1), Leaf(-2)), Leaf(0)), Node(6,
Leaf(-0), Node(4, Leaf(1), Leaf(-2))))) = 6
```

Prompt:

```
let biggest_Node t = collapse_tree (_Blank 1_) (_Blank 2_);;
```

Blank #1:

```
fun x l r -> if x > l && x > r then x else if l > r then l else r
```

Blank #2:

```
t
```