CMSC 330 Quiz 2 Fall 2022

Q1. OCaml Typing

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

\[
\text{fun } a \ b \rightarrow a +. \ \text{float_of_int} \ b
\]

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

\[
\text{fun } w \ x \ y \ z \rightarrow x::(z \ w \ y)
\]

Q2. Type Check

The following expression does not type check:

\[
\text{fun } f \ a \ b \rightarrow \text{if } a+1=2 \ \text{then } a \ \text{else if } 3 \ \text{then } b+.1.0 \ \text{else } (f \ b)
\]

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 =
    \text{match } t \ \text{with}
    | \text{Leaf} \rightarrow 
      \text{(match } acc \ \text{with}
        | \{s, []\} \rightarrow \text{acc}
        | \{s, t::ts\} \rightarrow \text{fun_a (s,ts) t)
      | \text{Branch}(l,v,r) \rightarrow 
        \text{(match } acc \ \text{with}
```

let rec fun_a (v+s, r::ts) l =
    (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.

```ocaml
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. The two blanks below refer to the parameters passed in for the 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