COMP 212 : Functional Programming, Spring 2023

Homework 07

Name: ______________________________

Wes Email: ______________________________

Question	Points	Score
1	25	
Total:	25	

If possible, please type/write your answers on this sheet and upload a copy of the PDF to your google drive handin folder. Otherwise, please write the answers in some sort of word processor and upload a PDF. Please name the file `hw07-written.pdf`.

1. **Map Fusion**

Earlier in the course, we had a function `raiseBy : int list * int -> int list` that added its `int` argument to each element of the `int list`. We proved that for all values `l : int list`, `a: int`, `b: int`,

$$\texttt{raiseBy}(\texttt{raiseBy}(\texttt{l},\texttt{a}),\texttt{b}) \cong \texttt{raiseBy}(\texttt{l},\texttt{a}+\texttt{b})$$

This is a special case of a property called *map fusion*. Recall the `map` function:

```
fun map (f : 'a -> 'b, l : 'a list) : 'b list =
    case l of
        [] => []
      | x :: xs => f x :: map (f,xs)
```

Mapping `f` over some list `l` and then mapping another function `g` over the result gives a list that is equivalent to the one you would get if you map `fn x => g (f x)` ("g composed with f") over the original `l`. We can write this more concisely by defining an abbreviation for function composition, the function that applies `f` and then applies `g`:

```
fun (g : 'b -> 'c) o (f : 'a -> 'b) = fn x => g (f x)
```

The property we would like to prove is that for all lists `l`,

$$\texttt{map (g o f, l)} \cong \texttt{map(g, map f l)}$$

Unfortunately, this is *false* for certain `g` and `f`. We say that a function `f` is *total* if for all values `v`, `f v` is valuable: that is, a function is total iff it is valuable on all inputs.

(5) (a) Give functions `g` and `f` and a list `l` such that `map (g o f, l` and `map (g, map (f, l))` have different behaviors. Hint: consider `f` and `g` that are not total.

Solution:

However, we *can* prove this property for total `f` and `g`.

You may assume the following lemma:

Lemma 1. *For all types* `a`, `b` *and values* `f : a->b`, *if* `f` *is total then* `map(f,l)` *is valuable.*

Your job is to prove

Theorem 1. *For all types* `a`, `b`, `c`, *all values* `f : a -> b` *and* `g : b -> c`, *and* `l : a list` *if* `f` *and* `g` *are total, then*

$$\texttt{map(g, map(f,l))} \cong \texttt{map(g o f, l)}$$

Proceed by induction on the structure of `l`. **Be careful to note where you are using valuability, and explain why the expressions involved are valuable—where would your proof break for the non-total functions in your example above?**

(5) (a) **Solution:** Case for `[]`:

To show:

(15) (b) **Solution:** Case for `x::xs`, where `x` and `xs` are values:

IH:

To show:

Solution: (continue here if necessary)