CSE 341, Autumn 2018, Midterm Review
October 25, 2018

Selected problems in order:

- Spring 2013 #1
- Spring 2013 #3
- Spring 2016 #4
- Sprint 2016 #6

Name:_____

1. This problem uses this datatype binding for *ternary trees*, where a ternary tree is a tree where all non-leaves have exactly three children:

```
datatype int_ternary_tree = Leaf of int
                          | Node of int
                                    * int_ternary_tree
                                    * int_ternary_tree
                                    * int_ternary_tree
```

   (a) (**8** points) Write an ML function `to_list` of type `int_ternary_tree -> int list`. The result should have every number that appears anywhere in the argument (and no other numbers). If a number appears n times in the argument, then it also appears n times in the result. The order of numbers in the result does not matter.

   Use no helper functions other than `::` *and* `@`.

   (b) (**10** points) Write a second version of `to_list` that:

   - Does *not* use `@` (and not your own reimplementation of it)
   - *Does* use a locally-defined helper function of type `int_ternary_tree * int list -> int list`
   - Does *not* need to produce a list in the same order as your answer in part (a).

   (c) (**3** points) Is your answer to part (a) tail-recursive? Explain in 1-2 sentences.

   (d) (**3** points) Is your answer to part (b) tail-recursive? Explain in 1-2 sentences.

Name:_____

3. For each of the following programs, give the value `ans` is bound to after evaluation.

   (a) (**5** points)

```
fun f x y z = if z > 0 then (fn w => w + x + y) else (fn w => w + x - y)
val a = 1
val b = 2
val c = f b a
val d = c ~7
val ans = d 4
```

   (b) (**5** points)

```
fun f p =
    let
        val x = 3
        val y = 4
        val (z,w) = p
    in
        (z (w y)) + x
    end
val x = 1
val y = 2
val ans = f((fn z => x + z), (fn x => x + x))
```

   (c) (**5** points)

```
fun f x = x + 7

fun g y =
    if y > 0
    then (f (y-1)) + 1
    else 4
and f y = (* notice the keyword and on this line *)
    if y > 0
    then (g (y-1)) + 2
    else 5

val ans = f 3
```

Name:_____

4. (**20** points)

   (a) Without using any helper functions (except `::` and `=`), write a function `nonempty_for_x` of type `int -> ((int -> string) list) -> (string list)` as follows:

   - It takes two arguments `x` and `flist` in curried form.
   - The output list contains no empty strings (i.e., `""`).
   - The i^{th} element of the output list is the i^{th} non-empty string produced by calling each element of `flist` in order with `x`.

   Hint: You can see if a string is empty by comparing it to `""` using `=`.

   (b) Create a function `nonempty_for_x'` that is equivalent to `nonempty_for_x` by filling in these blanks with anonymous functions:

```
fun nonempty_for_x' x = (List.filter ____________________________________)

                        o (List.map ____________________________________)
```

   (c) Does your `nonempty_for_x` actually have a more general type than the type specified? If so, what is it?

   (d) Does your `nonempty_for_x'` actually have a more general type than the type specified? If so, what is it?

Name:_____

6. (**18** points) This problem considers two ML structures and two ML signatures, all related to intervals (also known as ranges) of integers where we consider a range like "3 to 7" to *include* both endpoints.

```
signature INTERVAL1 =
sig
type t = int * int
val make : int * int -> t
val contains : t * int -> bool
val size : t -> int
end
```

```
structure IntervalA =
struct
type t = int * int
fun make (x,y) = (Int.min(x,y), Int.max(x,y))
fun contains ((x,y),i) = x <= i andalso i <= y
fun size (x,y) = y - x
end
```

```
signature INTERVAL2 =
sig
type t
val make : int * int -> t
val contains : t * int -> bool
val size : t -> int
end
```

```
structure IntervalB =
struct
type t = int * int
fun make (x,y) = (Int.min(x,y), abs (y - x))
fun contains ((x,len),i) = x <= i andalso i <= x + len
fun size (_,len) = len
end
```

(a) Does `IntervalA` have signature `INTERVAL1` (i.e., would `structure IntervalA :> INTERVAL1 ...` typecheck)?

(b) Does `IntervalA` have signature `INTERVAL2` (i.e., would `structure IntervalA :> INTERVAL2 ...` typecheck)?

(c) Does `IntervalB` have signature `INTERVAL1` (i.e., would `structure IntervalB :> INTERVAL1 ...` typecheck)?

(d) Does `IntervalB` have signature `INTERVAL2` (i.e., would `structure IntervalB :> INTERVAL2 ...` typecheck)?

(e) Suppose a program has two structures `S1` and `S2` both with signature `INTERVAL1`. Further suppose `S1`'s `make` is the same as in `IntervalA` and `S2`'s `size` is the same as in `IntervalB`.

i. Would `S2.size (S1.make (5,~5))` type-check?

ii. Regardless of whether it type-checks, if we assume we can evaluate it, what would `S2.size (S1.make (5,~5))` evaluate to?

(f) Repeat the previous question assuming `S1` and `S2` both have signature `INTERVAL2`.

(g) What is the type of `size` *inside* `IntervalA`? (Do not use type `t` in your answer.)

(h) What is the type of `size` *inside* `IntervalB`? (Do not use type `t` in your answer.)