Name:____

CSE341 Spring 2016, Midterm Examination
April 29, 2016

Please do not turn the page until 10:30.

Rules:

- The exam is closed-book, closed-note, etc. except for *one* side of one 8.5x11in piece of paper.
- **Please stop promptly at 11:20.**
- There are **100 points**, distributed **unevenly** among **6** questions (all with multiple parts):
- **The exam is printed double-sided.**

Advice:

- Read questions carefully. Understand a question before you start writing.
- Write down thoughts and intermediate steps so you can get partial credit. But clearly indicate what is your final answer.
- The questions are not necessarily in order of difficulty. Skip around. Make sure you get to all the questions.
- If you have questions, ask.
- Relax. You are here to learn.

Name:____

1. (**24** points) This problem uses this datatype binding, where a value of type `pipe` describes the shape of a pipe system (e.g., for carrying water).

   ```
   datatype pipe = Straight of int
                 | Curve of int * real
                 | Tee of pipe * pipe * pipe
                 | Sequence of pipe * pipe
   ```

   - `Straight i` represents a straight pipe of length `i` centimeters.
   - `Curve (i,r)` represents a curved pipe of length `i` centimeters with an arc of `r` radians, meaning the "curve" occupies $r/(2\pi)$ of a circle.
   - `Tee (p1,p2,p3)` is a tee (also known as a fork?) that connects the three pipes together, with `p2` and `p3` being in a line that is at a right angle to `p1`.
   - `Sequence (p1,p2)` connects the two pipes together.

   (a) Write a function `check_pipe` of type `pipe -> bool` that returns `true` if and only if all lengths anywhere in the argument are positive and all arcs in curves are strictly between 0 and 2π. (You can use `Math.pi`, which has type `real`.)

   (b) Write a function `scale_model` of type `pipe * int -> pipe` that creates a pipe of the same shape as the first input but with all lengths scaled (multiplied) by the second input. (Arcs stay the same.)

   (c) Consider this code that uses your answer to part (b);

   ```
   val little_p = Sequence (Straight (3+4), Curve (4+5, 1.5))
   val big_p = scale_model (little_p, 10)
   ```

   i. What value is bound to `little_p`?

   ii. What value is bound to `big_p`?

   Solution:

   (a)
   ```
   fun check_pipe p =
     case p of
         Straight i => i > 0
       | Curve (i,r) => i > 0 andalso r > 0.0 andalso r < 2.0 * Math.pi
       | Tee(p1,p2,p3) => check_pipe p1 andalso check_pipe p2 andalso check_pipe p3
       | Sequence(p1,p2) => check_pipe p1 andalso check_pipe p2
   ```

   (b)
   ```
   fun scale_model (p,s) =
     case p of
        Straight i => Straight (i*s)
      | Curve(i,r) => Curve(i*s,r)
      | Tee(p1,p2,p3) => Tee(scale_model (p1,s),
                             scale_model (p2,s),
                             scale_model (p3,s))
      | Sequence(p1,p2) => Sequence(scale_model (p1,s),
                                    scale_model (p2,s))
   ```

   (c) i. `Sequence (Straight 7,Curve (9,1.5))`

   ii. `Sequence (Straight 70,Curve (90,1.5))`

Name:____

2. (**17** points) This problem uses this ML code:

```
fun foo (xs,ys) =
  case (xs,ys) of
     ([],[]) => []                    (* branch 1 *)
   | ([],_) => ys                     (* branch 2 *)
   | (_,[]) => xs                     (* branch 3 *)
   | (x::xs',_) => x::(foo(ys,xs'))  (* branch 4 *)
```

(a) Give *three different* inputs to `foo` that all lead to the output `[1,2,3,4]`. Each of your answers should already be a value (i.e., not contain other expressions like addition or function calls).

(b) Is `foo` tail-recursive?

(c) What is the type of `foo`?

(d) For each of the following, give exactly one of these answers:

A. It leads to a "match nonexhaustive" warning.

B. It leads to no warning and the resulting function is equivalent to `foo` (the branch was unnecessary).

C. It leads to no warning but the resulting function is not equivalent.

i. What happens if we remove just **branch 1** (and, for parsing purposes, the `|` character that follows)?

ii. What happens if we remove just **branch 2**?

iii. What happens if we remove just **branch 3**?

iv. What happens if we remove just **branch 4**?

Solution:

(a) There are many solutions, including any 3 of the following:

```
([],[1,2,3,4])
([1],[2,3,4])
([1,3],[2,4])
([1,3,4],[2])
([1,2,3,4],[])
```

(b) No

(c) `('a list * 'a list) -> 'a list`

(d) i. B

ii. A

iii. B

iv. A

Name:____

3. (**12** points) For each of the following programs, give the value `ans` is bound to after evaluation.

(a)
```
val c = 12

fun f a =
  let
      val b = a - 1
      val a = b - 1
      val b = a - 1
  in
      c - b
  end

val c = 10
val ans = f c
```

(b)
```
fun f p =
  let
      val q = p 1
      val r = q 2
  in
      (r 3) + (p 0 0 0)
  end

fun g x =
  let
      val y = 6
  in
      f (fn z => fn w => fn t => z + w + t + y)
  end

val ans = g 7
```

(c)
```
exception E

fun h a =
  case a of
     NONE => raise E
   | SOME a => a

val a = 12
val ans = h (h (SOME (SOME a)))
```

Solution:

(a) 5

(b) 18

(c) 12

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?

Solution:

(a)
```
fun nonempty_for_x x flist =
  case flist of
      [] => []
   | f::flist' => let val s = f x in
                    if s = ""
                    then nonempty_for_x x flist'
                    else s :: nonempty_for_x x flist'
                 end
```

(b) A few ways:

```
fun nonempty_for_x' x = List.filter (fn s => String.size s > 0) o List.map (fn f => f x)
fun nonempty_for_x' x = List.filter (fn s => s <> "") o List.map (fn f => f x)
fun nonempty_for_x' x = List.filter (fn s => not (s = "") o List.map (fn f => f x)
```

(c) Yes, `'a -> ('a -> string) list -> string list`

(d) Yes, `'a -> ('a -> string) list -> string list`

Name:____

5. (**9** points)

   (a) What is `x` bound to after this ML code evaluates?

   ```
   val x = List.filter (fn i => i > 32 andalso i < 39) [0,99,35,36,14]
   ```

   (b) What is `y` bound to after this ML code evaluates?

   ```
   fun filterish f xs = List.foldl (fn (i,acc) => if f i then i::acc else acc) [] xs

   val y = filterish (fn i => i > 32 andalso i < 39) [0,99,35,36,14]
   ```

   (c) In approximately one English sentence, explain the general difference between `List.filter` and `filterish`.

Solution:

(a) `[35,36]`

(b) `[36,35]`

(c) One returns the reverse of the list the other returns.

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

signature INTERVAL2 =                structure IntervalB =
sig                                  struct
type t                               type t = int * int
val make : int * int -> t            fun make (x,y) = (Int.min(x,y), abs (y - x))
val contains : t * int -> bool       fun contains ((x,len),i) = x <= i andalso i <= x + len
val size : t -> int                  fun size (_,len) = len
end                                  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.)

Solution:

(a) yes

(b) yes

(c) yes

(d) yes

(e) i. yes

 ii. 5

(f) i. no

 ii. 5

(g) int * int -> int

(h) ’a * ’b -> ’b

Name:____

Here is an extra page in case you need it. If you use it for a question, please write "see also extra sheet" or similar on the page with the question.