CSE341 Spring 2017, Midterm Examination
April 28, 2017

Please do not turn the page until 12:30.

Rules:

- The exam is closed-book, closed-note, etc. except for one side of one 8.5x11in piece of paper.
- Please stop promptly at 1:20.
- 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.
1. (21 points) This problem uses this datatype binding, where a value of type miniHTML describes the contents of an HTML page that uses a tiny subset of HTML. No knowledge of HTML is necessary.

```ml
datatype miniHTML =
    Text of string (* string holds text that appears on the page *)
  | Image of string * int * int (* string is the name of the file containing the image, first int is image width in pixels, second int is image height in pixels *)
  | BulletList of miniHTML list (* a bullet list of other items *)
```

For both problems below, you can use local helper functions or ML standard-library functions.

(a) Write a function `numPixels` of type `miniHTML -> int` that computes the count of all the pixels that are in any of the images anywhere in the argument. (Images are rectangles.)

(b) Write a function `noImages` of type `miniHTML -> miniHTML` where the result is like the argument except the result has no images. Each image should be replaced with text: If the file containing the image is `foo`, then the replacement text should be `removed image from file foo`.
2. (19 points) This problem uses the following ML code where an `int_tree` represents a binary tree of ints, but only internal nodes hold ints (leaves do not). Note values of type `int_tree` are not assumed to be sorted in any particular way (for example, they are not binary search trees).

```ml
exception Empty
datatype int_tree = Leaf | Node of int * int_tree * int_tree

fun min t =
    case t of
    (* 1 *)
        Leaf => raise Empty
    (* 2 *)
        Node(i,Leaf,Leaf) => i
    (* 3 *)
        Node(i,t1,Leaf) => let val m = min t1 in if i < m then i else m end
    (* 4 *)
        Node(i,t1,t2) => let
                        val m1 = min t1
                        val m2 = min t2
                        val m3 = if m1 < m2 then m1 else m2
                        in
                        if i < m3 then i else m3
                        end
    (* 5 *)

    (a) What is the type of the `min` function above?

    (b) Assuming `min` is supposed to compute the smallest number in a tree, it is wrong. Fix it by adding one more branch to the case expression in one of the five positions indicated by comments above. That is, write the code for one more branch and indicate where you are putting it.

    (c) Give an example argument to `min` such that `min` behaves differently before and after your change in part (b). Indicate the behavior of `min` before and after your change.

    (d) Now consider moving the code you added in part (b) to one of the other positions. For each position, give one of these answers:
    A. This position would work too.
    B. This position is actually the one you already picked in part (b).
    C. This position would not work: moving your part (b) answer here would either lead to a type-checking error or to `min` producing the wrong answer.

    Do not consider in your answer that the first branch has no `|` character and the others do. That is, assume we fix that syntactic issue as necessary.

    Position 1:    Position 2:    Position 3:    Position 4:    Position 5:
3. (13 points) In this problem, we ask you to give good error messages for why a short ML program does not type-check. A specific phrase or short sentence is plenty.

For example, for the program,

```
fun f1 (x,y) = if x then y + 1 else x
```

a fine answer would be, “the then-branch-expression and the else-branch-expression do not have the same type.”

Give good error messages for each of the following:

(a) fun f2 x y =
    if x = 0
    then y + 1
    else 2 * f2 (x-1,y)

(b) fun f3 x y = x + y + a
    val a = 17
    val b = f3 6
    val c = b

c) fun f4 (xs,ys) =
    case (xs,ys) of
    ([],_) => 0
    | (_,[]) => 0
    | ([],[]) => 0
    | (_::xs,_::ys) => 1 + f4 (xs,ys)

d) fun f5 = List.map (fn x => x + 3)
4. (20 points)

(a) Without using any helper functions (except :), write a function choose_map of type
\((\forall a \quad (a \times a \to \text{bool}) \to (a \to b) \to a\ \text{list} \to a\ \text{list} \to b\ \text{list})\) as follows:

- It takes four arguments in curried form.
- The length of the result is equal to the length of the longer of the third and fourth arguments.
- The \(i^{th}\) element of the output is the second argument applied to the \(i^{th}\) element of either the third argument or the fourth argument.
- To choose which, we use the first argument on the two \(i^{th}\) elements — a result of true means use the third argument’s element and a result of false means use the fourth argument’s element.
- But if the lists have different lengths, then for positions past the end of the shorter list we just use the element of the longer list and, for such elements, the first argument to choose_map is irrelevant.

(b) Use a val binding and a partial application of choose_map to define a function pick_bigger of type \((\forall n \quad \text{int list} \to \text{int list} \to \text{int list})\) where, for example,
\[
pick\_bigger \ [1,7,9] \ [0,10,9,4,2] = [1,10,9,4,2]
\]

(c) Fill in the three blanks below such that \(t2\) evaluates to true.

\[
\text{fun} \ \text{zero\_floor} \ \text{xs} = \text{pick\_bigger} \ (\text{List.map} \ \_ \ \_ \ \_) \ \_
\]
\[
\text{val} \ t2 = \text{zero\_floor} \ [\sim2,\sim4,0,5,\sim3,7] = [0,0,0,0,7]
\]

(d) What is the type of zero_floor?
5. (8 points)

(a) Your friend proposes replacing the implementation of ML’s List.map with this tail-recursive implementation:

```ml
fun new_map f xs =
  let
    fun loop acc xs =
      case xs of
        [] => acc
      | x::xs => loop ((f x)::acc) xs
  in
    loop [] xs
  end
```

Explain to your friend in roughly 1 English sentence why this is a bad idea.

(b) Now your friend proposes adding this function of type

(`a * `b -> `c) -> (`a * `b) list -> `c list to ML’s list library as a useful iterator over lists of pairs:

```ml
fun map_pair f xs =
  case xs of
    [] => []
  | (a,b)::xs => (f (a,b)) :: (map_pair f xs)
```

Explain to your friend in roughly 1 English sentence why this is a bad idea.
6. (19 points) This problem considers this ML module definition:

```ml
structure Circ :> CIRC =
struct
  type circle = real * real * real (* center x-coordinate, center y-coordinate, radius *)
  val unitCircle = (0.0, 0.0, 1.0) (* center at origin, radius 1 *)

  fun moveX ((x,y,r), dx) = (x+1.0*dx, y, r)
  fun moveY ((x,y,r), dy) = (x, y+1.0*dy, r)
  fun scaleR ((x,y,r), f) = (x, y, 1.0*r*f)
  fun intersect ((x1,y1,r1),(x2,y2,r2)) = let
    val dx = x2 - x1
    val dy = y2 - y1
    in
      Math.sqrt(dx*dx + dy*dy) < r1 + r2
    end
end
```

(a) Complete this signature definition so that clients can use all the variable and function bindings in Circ for circles but the type of circles is abstract.

```ml
signature CIRC =
sig
  type circle
end
```

(b) Consider a function `area` for computing the area of a circle (hints: (1) \(\pi r^2\), (2) `Math.pi`).

i. Write an implementation of `area` that works inside the Circ module.

ii. Extend CIRC to make `area` available to clients.

iii. What type does your `area` implementation have inside the Circ module?

iv. Yes or no: If `area` is not provided by the module, is it possible for clients to write an equivalent function outside the module?

(c) Consider a function `funnyCircleMaker` that takes one real (call it a) and produces a circle with center-x-coordinate, center-y-coordinate, and radius that are all a.

i. Write an implementation of `funnyCircleMaker` that works inside the Circ module.

ii. Extend CIRC to make `funnyCircleMaker` available to clients.

iii. What type does your `funnyCircleMaker` implementation have inside the Circ module?

iv. Yes or no: If `funnyCircleMaker` is not provided by the module, is it possible for clients to write an equivalent function outside the module?
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.