Name:____

CSE 341, Spring 2008, Midterm Examination
30 April 2008

Please do not turn the page until everyone is ready.

Rules:

- The exam is closed-book, closed-note, except for one side of one 8.5x11in piece of paper.
- **Please stop promptly at 10:20.**
- You can rip apart the pages, but please staple them back together before you leave.
- There are **95 points** total, distributed **unevenly** among **5** questions (all with multiple parts).
- When writing code, style matters, but don't worry about indentation.

Advice:

- Read questions carefully. Understand a question before you start writing.
- **Write down thoughts and intermediate steps so you can get partial credit.**
- The questions are not necessarily in order of difficulty. **Skip around.**
- If you have questions, ask.
- Relax. You are here to learn.

Name:____

1. This problem uses this datatype definition:

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

   (a) (**6** points) Write a function `leftmost` that returns the `int` farthest to the left in a `int_tree`. (Given `Node(e1,e2)`, everything in `e1` is to the left of everything in `e2`.)

   (b) (**10** points) Write a function `max` that returns the greatest `int` in an `int_tree`.

Name:____

2. (a) (**7** points) Describe *what* function `m1` computes (not *how* it computes it):

```
fun m1 lst =
  case lst of
    [] => []
  | x::[] => x::[]
  | x::(_::z) => x :: (m1 z)
```

(b) (**6** points) Describe *what* function `m2` computes. Hint: It is *not* the same as what `m1` computes.

```
fun m2 lst =
  let fun loop (lst1,lst2) =
     case lst1 of
       [] => lst2
     | x::[] => x::lst2
     | x::(_::z) => loop(z,x::lst2)
   in
     loop(lst,[])
   end
```

(c) (**4** points) Briefly explain why `m2` might be more efficient than `m1`.

(d) (**4** points) Even though `m1` and `m2` are not equivalent, there are situations where it does not matter which you use, even if you do not know anything about the list they are called with. Describe an example of such a situation.

Name:____

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

(a) (**5** points)

```
val x = 2
fun f (z,w) = x + z + w
val z = 3
val x = 4
val ans = f(z,x)
```

(b) (**6** points)

```
val x = 2
fun f (z,w) = fn x => x + z + w
val g = f(3,4)
val z = 5
val ans = g z
```

(c) (**5** points)

```
fun f g x =
   case x of
     NONE => NONE
   | SOME y => SOME (g y)
val x = SOME 17
val ans = f (fn y => y+1) x
```

Name:____

4. (a) (**11** points) Write a function `map_alternate` that takes two functions `f` and `g` and a list `lst` and returns a `lst` produced by applying `f` to the elements in odd positions in the list (the first element, the third element, the fifth element, etc.) and applying `g` to the elements in even positions in the list (the second element, the fourth element, the sixth element, etc.)

   - `map_alternate` should takes its argument *in curried form* with `f` then `g` then `lst`.
   - Do not use any helper functions nor any ML library functions.
   - Hint: Think carefully about how to call `map_alternate` recursively to produce a short elegant solution.

   (b) (**5** points) What is the type of `map_alternate`?

   (c) (**4** points) Use a `val` binding and `map_alternate` to define `double_odds`, which should take a list of integers and return a list where the numbers in odd positions in the list are doubled and the numbers in even positions are unchanged. For example, `double_odds [5,7,8,6,1]` evaluates to `[10,7,16,6,2]`.

   (d) (**2** points) What is the type of `double_odds`?

Name:____

5. Consider this signature and structure definition for a module that implements "digits" — one-digit numbers that wrap around when you increment or decrement them.

```
signature DIGIT =
sig
type digit = int
val make_digit : int -> digit
val increment : digit -> digit
val decrement : digit -> digit
val down_and_up : digit -> digit
val test : digit -> unit
end

structure Digit :> DIGIT =
struct
type digit = int
exception BadDigit
exception FailTest
fun make_digit i = if i < 0 orelse i > 9 then raise BadDigit else i
fun increment d = if d=9 then 0 else d+1
fun decrement d = if d=0 then 9 else d-1
val down_and_up = increment o decrement
fun test d = if down_and_up d <> d then raise FailTest else ()
end
```

   (a) (**5** points) Give example client code (code outside the module) that can cause the `FailTest` exception to be raised.

   (b) Consider each of the following changes to the `DIGIT` signature separately. Answer "yes" if it is still possible to raise the `FailTest` exception and "no" if it is no longer possible. For each, **briefly explain why**.

      i. (**5** points) Remove the line `val test : digit -> unit`.

      ii. (**5** points) Remove the line `val down_and_up : digit -> digit`.

      iii. (**5** points) Replace the line `type digit = int` with `type digit`.

Name:____

This page intentionally blank.