

Name: _____

**CSE 341, Spring 2004, Midquarter Examination
28 April 2004**

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 1:20.**
- You can rip apart the pages, but please write your name on each page.
- There are a total of **60 points**, distributed unevenly among five questions.
- 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. Consider these 3 functions. `append` appends two lists, as discussed in lecture.

```
fun append (l1,l2) =  
  case l1 of  
    [] => l2  
  | hd::t1 => hd::(append(t1,l2))  
fun f1 (l1,l2,l3) = append(l1, append(l2,l3))  
fun f2 (l1,l2,l3) = append(append(l1,l2), l3)
```

For parts (c) and (d), let `a1` be a list with `n1` elements, `a2` be a list with `n2` elements, and `a3` be a list with `n3` elements.

- (a) **(3pts)** What do `f1` and `f2` compute?
- (b) **(3pts)** What type do `f1` and `f2` have?
- (c) **(5pts)** How many times does evaluation of `f1(a1,a2,a3)` call the `::` constructor?
- (d) **(5pts)** How many times does evaluation of `f2(a1,a2,a3)` call the `::` constructor?

Name: _____

2. This problem considers copying pairs of integers.
 - (a) **(2pts)** Write an ML function `copy_pair` that takes a pair of integers and returns a new pair with the same field values as the argument. Make sure your function builds a new pair.
 - (b) **(6pts)** Suppose we take a program using `copy_pair` and replace some of the uses of the `copy_pair` function with the identity function (`fn x => x`). Is the resulting program contextually equivalent to the original one? If so, why? If not, under what circumstances is it not equivalent?

Name: _____

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

(a) (3pts)

```
val x = 1
val y = x+1
val x = y+1
val ans = x+y
```

(b) (3pts)

```
val x = 1
fun f y = x
val x = (f 3) + (f 2)
val ans = f x
```

(c) (3pts)

```
exception E
val f = (fn y => raise E) handle E => (fn z => z + 1)
val ans = (f 37) handle E => 14
```

Name: _____

4. Suppose we add a new construct to ML called `awhile` with this definition:

Evaluating `awhile(g,f,acc)` produces $\underbrace{f(f\dots(f\ acc)\dots)}_{n\ \text{times}}$ where n is the *minimum* number such that `g` applied to the result is *false*.

For example, if `g acc` is false, then n is 0 and the result is `acc`.

(a) (**7pts**) Show that we do not need to extend ML with `awhile` because you can implement it as a function. In other words, provide a function body for the incomplete binding below.

```
fun awhile(g,f,acc) =
```

(b) (**8pts**) This incomplete function uses `awhile`. Complete the function such that it computes $base^{exp}$ when exp is positive. (Your solution can be wrong for $exp \leq 0$.) Hint: `base` is in scope throughout the body of `pow`.

```
fun pow(base,exp) =
```

```
  let val _____ =
      awhile(fn (acc,exp) => _____,
            fn (acc,exp) => _____,
            (base,exp))
  in
    _____
  end
```

Name: _____

The next page is blank so you have room for answers

5. Consider this structure:

```
structure L :> LSIG =
struct
  datatype my_int_list = Empty | Cons of int * my_int_list
  exception BadList
  fun makeOne i = Cons(i,Empty)
  fun my_hd lst = _____
  fun my_tl lst = _____
end
```

- (a) **(3pts)** Write bodies for `my_hd` and `my_tl` such that each takes a value of type `my_int_list`, raises `BadList` if the value is `Empty`, and evaluates to the first (for `my_hd`) or second (for `my_tl`) field of the pair that `Cons` carries.
- (b) **(9pts)** For each of the following `LSIG` definitions, determine if a client of `L` can make `BadList` get raised. If so, give an example client for which `BadList` is raised. If not, briefly explain why not. (Examples may name available structure elements directly. For example, in the first two parts you can write `my_hd` instead of `L.my_hd`.)

i. signature `LSIG =`
sig
 datatype `my_int_list = Empty | Cons of int * my_int_list`
 val `my_hd : my_int_list -> int`
 val `makeOne : int -> my_int_list`
end

ii. signature `LSIG =`
sig
 type `my_int_list;`
 val `Cons : int * my_int_list -> my_int_list`
 val `my_hd : my_int_list -> int`
 val `makeOne : int -> my_int_list`
end

iii. signature `LSIG =`
sig
 type `my_int_list;`
 val `Cons : int * my_int_list -> my_int_list`
 val `my_tl : my_int_list -> my_int_list`
 val `makeOne : int -> my_int_list`
end

Name: _____

This page is intentionally blank.