CSc 372, Fall 2001
ML Examination
September 26, 2001

READ THIS FIRST

Do not turn this page until you are told to begin.

On problems that ask you only to write code, you need not include any explanation in your answer if you are confident it is correct. However, if an answer is incorrect, any accompanying explanation may help you earn partial credit.

If you have a question, raise your hand and the instructor or a teaching assistant will come to you. DO NOT leave your seat.

If you have a question that can be safely resolved by making an assumption, simply write down that assumption and proceed. Examples:

> "Assuming `length` is `string -> int`"
> "Assuming the homework function to insert values in a list is `insert_n`."
> "Assuming `ien(L,n,v)` and `repl(n,s)`" (if uncertain of argument order)

BE CAREFUL with assumptions that may significantly change a problem. For example, consider the expression `f [x 1, 2, 3]`. It may appear that a comma has been omitted in "`x 1`" but in fact there is no error.

You may use only language elements that have been studied in the class. You may use built-in functions such as `size`, `map`, and `rev`. You may use functions presented on the slides or written thereon, such as `member`, `filter` and `reduce`. You may use functions presented via the mailing list, such as `make_curried_r`. You may use functions that have appeared on the homework assignments this semester such as `gather` and `ien`. You may not use the `hd` or `tl` functions.

When you have completed the exam, enter your name on the exam sign-out log and then hand your exam to the instructor or a teaching assistant.

New DO NOT use your own paper. Do not write on the opposite side of the paper. If you need extra sheets, ask for them.

New Here is a list of functions that may be useful: `explode`, `filter`, `ien(L,n,v)`, `implode`, `map`, `print(s)`, `reduce`, `repl(s,n)`, `split d s`, `sum(L)`.

New There are no restrictions on the use of recursion and no deductions for poor style. Anything that works and doesn't use off-limits elements, such as `hd` and `tl`, is acceptable.

Print your name below **and when told to begin**, put your initials in the lower right hand corner of each page, being sure to check that you have all the pages.

This is a **forty-five** minute exam with a total of 100 points. There are nine problems on seven pages.

Seat row **and** number: __________ Name:____

Problem 1: (2 points each; 8 points total)

State the ***type*** of each of the following expressions, or if the expression is not valid, state why. For example, the type of the expression `3+4` is `int` and the type of the expression `length` is `'a list -> int`.

```
(1, [2,3], 4.0)
```

```
(reduce op+)
```

```
explode o rev o implode
```

```
[[[size]]]
```

Problem 2: (3 points each; 15 points total)

State the ***type*** of each of the following functions:

```
fun a (w,h) = w * h * 1
```

```
fun f(x) = f(1) + f(2);
```

```
fun f(L,(a,b)) = L = (a,b)
```

```
fun f(3,4) = (size,length)
```

```
fun x y z = (y z) + z
```

Problem 3: (3 points each, 9 points total)

Edit or rewrite the following functions to make better use of the facilities of ML:

```
fun f(a,b,c) = [a-c, a+c]
```

```
fun f(x,y) = x::y::1::2::[]
```

```
fun f(n) = if n = 10 then true else if n = 5 then true else false;
```

Problem 4: (5 points)

Write the `map` function. The type of map is `('a -> 'b) -> 'a list -> 'b list`

Problem 5: (7 points)

Create a function `abslist(L)` of type `real list -> real list` that produces a copy of L with each value in the output list being the absolute value of the corresponding value in the input list. Assume there is NO function like Java's `Math.abs()` to compute absolute value–do the absolute value computation yourself.

Examples:

```
- abslist([1.0, ~2.0, ~3.4, 100.0]);
val it = [1.0, 2.0, 3.4, 100.0] : real list

- abslist([]);
val it = [] : real list
```

Problem 6: (7 points)

Your instructor suffered the great embarrassment of distributing a version of `gather` that has a bug: If called with an empty list it should return `[]` but in fact it returns `[[]]`. Example:

```
- gather([], 10);
val it = [[]] : int list list
```

Here is the source for the instructor's faulty version of `gather`:

```
fun gather(L, limit) =
  let
    fun g([], _, _, result) = [result]
     |  g(x::xs, sum, limit, result) =
          if result = [] then
             g(xs, x, limit, [x])
          else
             if x + sum > limit then
                 result::g(x::xs, 0, limit, [])
             else
                 g(xs, sum+x, limit, result @ [x])
  in
    g(L, 0, limit, [])
  end
```

Produce a version of `gather` that has the correct behavior. Either mark changes to the above code or rewrite it. Note: You only need to fix that one bug–don't search for additional problems. Your changes should not introduce additional bugs, of course.

Problem 7: (15 points)

In this problem you are to create TWO functions, `doubler` and `quadrupler`. `doubler` is of type `string list list -> string list list` and "doubles" each letter in the strings. `quadrupler` is of the same type, but quadruples each letter.

Examples:

```
- doubler([["just"], ["a", "test"], ["h", "e", "r", "e", ""]]);
val it = [["jjuusstt"],["aa","tteesstt"],["hh","ee","rr","ee",""]]
  : string list list

- doubler([["hello"]]);
val it = [["hheelllloo"]] : string list list

- doubler([ ]);
val it = [] : string list list

- doubler([[ ]]);
val it = [[]] : string list list

- quadrupler([["an", "example"],[ ]]);
val it = [["aaaannnn","eeeexxxxaaaammmmpppplllleeee"],[]] : string
list list
```

Problem 8a: (7 points)

Create a function `genlist` that takes a list of integers and for each integer N in the list, produces a list with N instances of the number 1. You may assume that all the values are non-negative.

Examples:

```
- genlist;
val it = fn : int list -> int list list
- genlist [2,1,0,7,4];
val it = [[1,1],[1],[],[1,1,1,1,1,1,1],[1,1,1,1]] : int list list
- genlist [];
val it = [] : int list list
- genlist [10];
val it = [[1,1,1,1,1,1,1,1,1,1]] : int list list
```

Problem 8b: (7 points)

Create a function `genlist_inv` that performs the inverse operation of `genlist`. The only value appearing in the lists will be the integer 1 (one). Examples:

```
- genlist_inv [[1,1,1], [1], [1,1]];
val it = [3,1,2] : int list
- genlist_inv (genlist [2,1,0,7,4]);
val it = [2,1,0,7,4] : int list
- genlist_inv [[],[],[1],[]];
val it = [0,0,1,0] : int list
```

Problem 8c: (2 points)

What is the type of `genlist_inv o genlist o genlist_inv` ?

Problem 9: (18 points)

Imagine a function `read_all_bytes(fname)` that produces a string containing all the bytes in the file `fname`. For a file "x" containing the following five lines of text,

```
this
is @ some data
here to
test@
with
```

the string `"this\nis @ some data\nhere to\ntest@\nwith\n"` is returned by `read_all_bytes("x")`. Note that `\n` represents one character, a newline.

Create a function `tacdel(fname)` that reads the file named by `fname` and prints (using the `print` function) the lines in the file in reverse order, and if a line contains the character "`@`", the line "`<D>`" appears in its place. Assume that the file exists and is readable. Example:

```
- tacdel("x");
with
<D>
here to
<D>
this
val it = () : unit
```

Here is a function you may use:

```
fun member (v, []) = false
 |  member (v,x::xs) = if x = v then true else member(v,xs)
```