Preliminary Examination I
Computer Science 312, Cornell University 6 March 2003

Before starting the exam, write your name on this page and your netid on both this page and the next page.

There are 5 problems on this exam. It is 10 pages long; make sure you have the whole exam. You will have 90 minutes in which to work on the problems. You will likely find some problems easier than others; read all problems before beginning to work, and use your time wisely. The prelim is worth 100 points total. The point breakdown for the parts of each problem is printed with the problem. Some of the problems have several parts, so make sure you do all of them!

This is an closed-book examination; you **may not** use outside materials, calculators, computers, etc.

Do all written work on the exam itself. If you are running low on space, write on the back of the exam sheets and be sure to write (OVER) on the front side. It is to your advantage to show your work — we will award partial credit for incorrect solutions that are headed in the right direction. If you feel rushed, try to write a brief statement that captures key ideas relevant to the solution of the problem.

If you finish in the last ten minutes of the exam, please remain in your seat until the end of the exam as a courtesy to your fellow students.

Name and NetID ______________________________

Problem	Points	Score
1	10	
2	30	
3	20	
4	30	
5	10	
Total	100	

1. True/False [10 pts]

   (parts a–e; **4** points off for each wrong answer, 2 points off for each blank answer, minimum problem score 0.)

a. ____ SML variables have lexical scope.

b. ____ A value is polymorphic if it has a single type.

c. ____ Weak specifications should be avoided.

d. ____ Both 'requires' and 'checks' clauses state preconditions.

e. ____ Higher-order functions are functions that return types rather than ordinary values.

2. Abstraction [30 pts] (parts a–e)

   The following is the interface and implementation of a data abstraction:

```
signature SORTED_SEQUENCE = sig
  (* A "sequence" is a sorted sequence: a possibly empty
   * sequence of integer elements [m1, m2, ... mn] in
   * ascending (nondecreasing) order.
   * Examples: [~1,2,3], [0,2,3,3], []
   *)
  type sequence
  (* empty() is an empty sequence. *)
  val empty : unit -> sequence
  (* add(s,n) is the sequence containing n and all the
   * elements in s. *)
  val add: sequence*int -> sequence
  (* prepend([m1,...,mn], m') is the sequence [m', m1,...mn]. *)
  val prepend: sequence*int -> sequence
  (* nth(s,n) is the nth element of s (indices start at zero). *)
  val nth: sequence*int -> int
  (* size(s) is the number of elements in s. *)
  val size: sequence -> int
end
structure SortedSeq : SORTED_SEQUENCE = struct
  type sequence = int * int list
  fun empty() = (0, [])
  fun add((sz, lst), m) = raise Fail "Implement me!"
  fun prepend((sz, lst), m) = (sz+1, m::lst)
  fun nth((sz, lst), n) = List.nth(lst, n)
  fun size(sz,lst) = sz
end
```

   (a) [8 pts] Two of these methods need requires clauses. Which are they? For each method, explain why a requires clause is needed, and give a requires clause that addresses the problem you have identified.

(b) [4 pts] The implementer has failed to specify an abstraction function. Give a specification of the abstraction function that this implementation uses.

(c) [4 pts] The implementer has failed to write a representation invariant. Give a representation invariant that is strong enough to prove that all of the implemented functions work correctly. (Do not give the proofs.)

(d) [8 pts] The `add` function is unimplemented. Write an implementation that satisfies the given specification and preserves the representation invariant.

(e) [6 pts] One test case for the function `add` is `add(`$[1, 2, 4, 5]$`, 3)`. Give up to eight more good black-box test cases for the function `add`, which collectively provide good coverage. For each test case, write three or four words explaining how it improves coverage. You will lose points for providing redundant test cases.

Note: in the example test case, $[1, 2, 4, 5]$ represents a sequence, not an SML list, which is hinted at by its font. For brevity, use this list-like notation in your test cases.

3. Zardoz [20 pts] (parts a–d)

For each of the following expressions, give a *value* that causes the expression to evaluate to 42 if the box $\boxed{}$ is replaced by that value. A list of values `[`$v_1, \ldots, v_n$`]` is considered a value.

(a) [5 pts]

```
let val f: string list = [    ]
in
 case List.map(fn(x) =>
               case Int.fromString(x) of
                 SOME n => n
               | _ => 0) f of
   x::y::_ => x*10 + y
 | _ => 41
end
```

Your answer: ______________________

(b) [5 pts]

```
let fun zardoz(x:int):int list =
      if x = 0 then [] else x::zardoz(x-1)
in
  case zardoz( [    ] ) of
    w::x::y::z => x*y
end
```

Your answer: ______________________

(c) [5 pts]

```
let fun zardoz(b: bool list):int =
      case b of
        nil => 0
      | true::b' => 1 + 2*zardoz(b')
      | false::b' => 2*zardoz(b')
in
  zardoz [    ]
end
```

Your answer: ______________________

(d) [5 pts]

```
let val f = $\boxed{\phantom{xx}}$ in
  17 + (#2 (f(12, "hello"))) + (#1 (f("goodbye", 13)))
end
```

Your answer: $\boxed{}$

4. Correctness and Evaluation [30 pts] (parts a–e)

   Consider the following implementation of a factorial function:

```
fun f(m,n) = if n = 1 then m else f(m*n, n-1)
(* fact(n) is n! *)
fun fact(n) = f(1,n)
```

   (a) [3 pts] Critique the specification of the function `fact`. How would you change this specification to make it better?

   (b) [6 pts] Show each of the evaluation steps involved in the evaluation of `fact(3)` according to the substitution model.

   (c) [6 pts] The function `f` has no specification above. Complete the following specification in such a way that `f` satisfies it *and* it is strong enough to argue that `fact` computes the factorial of its argument.

```
(* f(m,n) is
```

   (d) [3 pts] Using the specification you just wrote in 4(c), argue that `fact` satisfies its own specification. In your argument, rely only on the specification of `f` and do not use any other information about the implementation of `f`.

(e) [12 pts] Prove by induction that `f` satisfies the specification of 4(c). Make sure to perform all of the steps involved in a proof by induction.

5. Complexity [10 pts] (parts a–e)

   For each of the following pairs of functions $f(n)$ and $g(n)$, circle the one that is of the asymptotically largest order of growth, or write SAME if they are of the same order of growth.

   (a) [2 pts]

   $$f(n) = n, \qquad g(n) = n \lg n$$

   (b) [2 pts]

   $$f(n) = 1000n^2 + 10000, \qquad g(n) = n^3 - 5n^2$$

   (c) [2 pts]

   $$f(n) = n^{2.001}, \qquad g(n) = n^2 \lg n$$

   (d) [2 pts]

   $$f(n) = n^2 + n \lg n, \qquad g(n) = 2n^2 + n$$

   (e) [2 pts]

   $$f(n) = \lg^2 n, \qquad g(n) = \lg n$$