Final Examination
Computer Science 312, Cornell University 15 May 2003

1. True/False [12 pts] (parts a–f; **4** points off for each wrong answer, 2 points off for each blank answer, minimum problem score 0.)

a. ____ With copying garbage collection, half of the heap storage is unused while the program executes.

b. ____ SML infers that the term `(fn(x,y)=>x, fn(x,y)=>y)` has a type equivalent to `('a*'b->'a)*('c*'d->'d)`.

c. ____ The shortest-path algorithm does not work correctly in general if any edges of the graph have zero weight.

d. ____ A string searching algorithm that looks for a pattern string in a larger text string (for example, the Boyer-Moore algorithm) must examine each character of the text string to avoid missing an occurrence of the pattern string.

e. ____ The treap data structure simultaneously satisfies both the heap ordering invariant and the binary search tree invariant.

f. ____ In the environment model an expression of the form `(fn x =>` *e*`)` evaluates to an unboxed value.

2. Zardoz [15 pts] (parts a–b)

For each of the following expressions, give an *expression* to replace the box $\boxed{}$, so that the entire containing expression evaluates to 42.

(a) [7 pts]

```
let val zardoz: ('a->'b->'a) * (string * int) = [ ]
    fun g(z: ('a->'b->'a) * ('b*'a)): 'a =
      let val (f, (m, n)) = z in f n m end
in
    g(zardoz) + String.size (#1 (#2 zardoz))
end
```

Your answer:

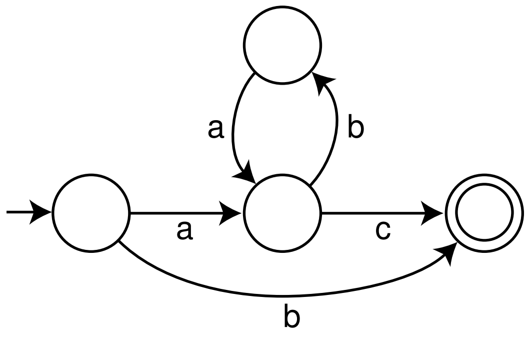
(b) [8 pts]

```
let
  fun zardoz (m: int -> int) (k: int) =
    if k > 13 then [    ]
    else m k - zardoz m (k + 1)
in
  zardoz (fn i => i + 3) 12
end
```

Your answer: $\boxed{}$

3. Strings [8 pts] (parts a–b)

Consider the following finite automaton:



Recall that a finite automaton accepts any strings consisting of the symbols traversed along the edges from the start state to an accept state. The start state is indicated by an incoming arrow and an accept state is indicated by a double circle.

(a) [3 pts] Give three distinct strings that are accepted by this automaton.

(b) [5 pts] Give a regular expression that matches exactly the strings that the automaton accepts.

4. Graphs [16 pts] (parts a–b)

(a) [8 pts]

Suppose that we have a simple priority queue implementation in which the queue is represented as a sorted list. Extracting the minimum element takes $O(1)$ time, but inserting a new element or changing the priority of an existing element both take $O(n)$ time. What would be the asymptotic run time of Dijkstra's shortest-path algorithm if implemented using this priority queue? Explain briefly.

(b) [8 pts]
Dijkstra's shortest-path algorithm finds the length of the shortest path from a single vertex of the graph to every vertex to the graph. Suppose that instead you have a *set* of source vertices V_0 and want to know the shortest distance to every vertex in the graph, starting from any vertex $v \in V_0$. Explain briefly how to use or modify the shortest-path algorithm obtain this distance instead. For full credit your solution should be asymptotically as efficient as Dijkstra's algorithm.

5. Trees [14 pts] (parts a–c)

Consider the following datatype for a binary search tree of elements of type `elem` ordered by an ordering function `compare: elem*elem->order`. No element occurs twice in the tree.

```
datatype tree = Nil | Node of {left: tree, value: elem, right: tree}
```

(a) [3 pts] What representation invariant does this data structure satisfy?

Consider a binary search tree whose root node contains the element x. Suppose that the tree contains at least one element greater than x. Then one of these elements greater than x must be the smallest such element. Deletion from a binary search tree works by replacing the element x with this smallest greater element (if it exists).

Let's consider binary search trees in which the elements are integers. For example, the following is a tree in which the smallest element greater than the root element (2) is at a leaf node (3) that is a right child of its parent:

```
  2
 / \
1   3
```

(b) [3 pts] Draw three more trees (and label them correspondingly):

i. Draw a tree in which the smallest element greater than the root element is at a leaf node that is a *left* child of its parent.

ii. Draw a tree in which the smallest greater element is not at a leaf node.

iii. Draw a tree containing at least two elements, in which there is no smallest greater element.

(c) [8 pts] Write the code *and* specification for a function `next: tree -> elem` that gives the smallest such element if it exists. You may assume that `compare` is in scope. Full credit will be given only to solutions that are $O(h)$ where h is the height of the tree.

6. Correctness [35 pts]   (parts a–j)

Consider the following data abstraction for whole (positive) integers and its implementation:

```
signature WHOLE = sig
  (* A whole is a whole number: for example, 1, 2, ... *)
  type whole
  (* one is 1 *)
  val one: whole
  (* succ(n) is n+1 *)
  val succ: whole -> whole
  (* plus(m,n) is m+n *)
  val plus: whole*whole -> whole
  val toInteger: whole -> int
end
structure Whole : WHOLE = struct
    (* AF: nil represents 1. false::n represents 2*AF(n).
     *     true::n represents 2*AF(n)+1.
     *)
    datatype whole = bool list
    val one = nil
    fun succ(n) = case n of
        false::n' => true::n'
    |   true::n' => false::succ(n')
    |   nil => [false]
    fun toInteger(n) = ...
    fun plus(m,n) = ...
end
```

(a) [1 pt] Explain in one sentence why no representation invariant is needed.

(b) [1 pt] Does every whole number have a unique representation? Answer yes or no.

(c) [2 pts] Give legal representations of the numbers 2, 4, 5, and 19.

(d) [3 pts] Give an implementation of `toInteger`.

```
    fun toInteger(n) =
```

(e) [5 pts] What is the asymptotic run time of `succ` as a function of n (interpreting n as a whole number, not a list!) You do not need to justify your answer. *Hint*: Consider $n = 2^k$.

(f) [5 pts] Complete the implementation of `plus`. It should use `succ`.

```
fun plus(m,n) = case (m,n) of
```

Now, use induction to prove that the implementation of `succ` (*not* `plus`!) is correct.

(g) [2 pts] Start by stating the proposition to be proved, in terms of the abstraction function.

Now, complete a proof by induction on the whole number $AF(n)$ being represented. Remember that the representations of whole numbers are not themselves numbers!

(h) [2 pts] Show that the base case holds.

(i) [2 pts]
State the induction hypothesis (in terms of n) and the induction step to be proved. Is this strong induction or ordinary induction?

(j) [12 pts] Show that the induction step follows, by considering the two possible structures of the representation. (Why are there only two possible structures?)