Final Exam: Mon. Dec 17, 8:30-11:30

Various topics we’ve seen:
Recursion
Proofs: Lots of induction, proper proof style
Specifications
Continuations
Exceptions
Lazy programming
Modularity (structures and signatures)
Imperative programming
Concurrency
Type-checking

From Fall 2006 Final:
2(i)
fun foldr f z [] = z
  | foldr f z (x::L) = f(x,foldr f z L)

fun ins (x, []) = [x]
  | ins (x, y::R) = if x=y then y::R else y::ins (x, R)

We say L “has no repeats” if all its members are different.

Prove that, for all suitably typed lists L and values x, if L has no repeats then ins(x, L) has no repeats. You can use the fact that the members of ins(x, L) are x and the members of L.

4
signature GRAPH =
sig
  type "a graph
  val build : ("a * "a) list -> "a graph
  val roots : "a graph -> "a list
  val delete : "a * "a graph -> "a graph
  val isempty : "a graph -> bool
end;

complete:
structure Edges : GRAPH =
struct
  type "a graph = ("a * "a) list
  fun build L =
  fun roots L =

fun delete = 
  fun isempty L = null L
end;

6
datatype Token = Left | Right
E ::= <empty> | Left E1 Right E2

Write parse of type
  parse : Token list -> (Token List -> bool) -> bool
such that
  parse L k is true if there is a pair of lists L1 and L2 such that L=L1@L2, L1 conforms to the grammar and k(L2) = true
  parse L k returns false if there is no pair of lists L1, L2 such that L=L1@L2, L1 conforms to the grammar and k(L2) = true

Write balanced of type
  balanced : Token list -> bool
Such that for all token lists L, balanced L returns true if L conforms to the grammar, returns false otherwise

9
Write simul of type
  simul : 'a ref list * 'a list -> unit
such that for all n>=0, all suitably typed refs x_1,...,x_n and values v_1,...,v_n,
  simul([x_1,...,x_n],[v_1,...,v_n]) has the same effect as the sequence of assignments x_1:=v_1; ...; x_n:=v_n. If two lists have unequal length, the function should raise the exception Unequal.

10
Write a recursive function
  parfold : ('a * 'b -> 'b) -> 'b -> ('a chan * 'b chan) -> unit
such that, for suitably typed F, z, a and b, if channel a is supplied with the sequence x_1,...,x_n,... and b is a distinct channel, a thread executing parfold F z (a,b) will send z to be, receive x_1 from a, send F(x_1,z) to b, receive x_2 from a, send F(x_2,F(x_1,z)) to b, etc..
Do NOT use foldl. Do not store intermediate results.