These examples are intended to give you an idea of the kind of questions on the midterm. They may be a little easier, a little harder, or a little less clear than actual midterm problems. As you read them, emphasize “mastering the concept” over “getting the answer.” Sample solutions follow the examples. *Work the problems and think them through completely before reading the solutions.*

1. The function `zip` was on an assignment. Write the corresponding function `unzip`.

2. Determine the type signatures of the functions `e`, `f`, and `g`.

   ```haskell
   fun f x [] 0 = x
   | f x [] z = e z
   | f x (y::ys) z =
     if x then
       f x (y::ys) (z-1)
     else
       f x ys z;
   fun g u v = v u;
   ```

3. There is a built-in function `List.filter`. Look up online what it does and then write your own version of filter. The call `filter isGreen lst` returns a list of all the green elements of `lst`.

4. The boolean version of `change` takes a list of coins and an amount, and tells if it is possible to make change in that amount from the given coins. Write `change` in the form of a boolean-valued function, and deduce its type signature.

5. (a) Write a function `ssums` that takes a list of integers and returns a list of the sums of all subsets of the elements of the list. Do not worry about eliminating duplicates.

   (b) What is the type signature of the function `ssums`?

   (c) If you were asked for a version of `ssums` that did not return a list with duplicate elements, how would you do it?

6. Recall from algebra that the composition of two functions `f` and `g` is a function `f ◦ g` defined by `(f ◦ g)(x) = f(g(x))`. 
(a) Write a curried function \texttt{comp} that takes two functions and produces their composition. [There is a built-in infix operator \texttt{o} (lowercase 'oh?) which composes functions.]

(b) What is the type signature of \texttt{comp}?

7. (a) Write a function \texttt{toList} that takes two integers, a value and a base, and represents the value as a list in the base. (You may assume that the value is non-negative and the base is greater than one. Do not worry about arithmetic overflow.)

(b) Write a function \texttt{fromList} that takes a list and a base and returns the integer represented by the list.
Solutions

1. There are many ways to write the function. This one is contorted to avoid a let construction and a named helper function. (FYI, there is a built-in function ListPair.unzip.)

   ```ml
   fun unzip [] = ([], [])
    | unzip ((u,v)::uvs) = (fn (x,y) => (u::x,v::y)) (unzip uvs);
   ```

2. e : int -> bool
   f : bool -> 'a list -> int -> bool
   g : 'a -> ('a -> 'b) -> 'b

3. fun filter p [] = []
   | filter p (s::ss) = 
      if p s then 
        s::(filter p ss)
      else
        filter p ss;

   filter: ('a -> bool) -> 'a list -> 'a list

4. fun change [] amt = amt=0
   | change (c::cs) amt = 
      (change cs amt) orelse 
      ((c <= amt andalso change (c::cs) (amt - c));

   change : int list -> int -> bool

5. (a) fun ssums [] = [0]
   | ssums (v::vs) = 
      let
        val recSums = ssums vs;
      in
        recSums @ (map (fn u => u + v) recSums)
      end;
   (b) ssums : int list -> int list
   (c) One way would be to use a version of the function uniquify; see Part IV, Section 7 of A Brief Introduction to SML.

6. fun comp f g = fn x => f(g x);

   comp : ('b -> 'c) -> ('a -> 'b) -> ('a -> 'c)

7. (a) fun toList value base = 
   if value = 0 then
     []
   else
     (value mod base) :: (toList (value div base) base);
(b) fun fromList nil _ = 0
    | fromList (d::ds) base = d + base * (fromList ds base);