1. Write a function to compute the sum of the squares of elements in the list?

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

3. True or False: The function below has the type signature: `(int * int) list -> int list:

   ```
   fun mystery lst = map (op /) lst;
   ```

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

   ```
   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;
   ```

5. Given the following function:

   ```
   fun check _ [] = []
   | check b (x::xs) = if b then
     x :: (check (not b) xs)
   else
     check (not b) xs;
   ```

   What would be returned if we call it with:
check true ["I", "really", "love", "coding", "SML"];

6. Write a function called check2 that is equivalent to check true (note, that since check is curried, this is a function), but that does not use booleans.
Solutions

1. fun squareSum [] = 0
   | squareSum (x::xs) = x*x + (squareSum xs);

2. 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.)

   fun unzip [] = ([], [])
   | unzip ((u,v)::uvs) = 
     let
       val (x,y) = unzip uvs;
     in
       (u::x,v::y)
     end;

   (* or *)

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

3. False. Remember, that / is for reals!

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

5. ["I","love","SML"]

6. fun check2 [] = []
   | check2 [x] = [x]
   | check2 (x::y::xs) = x::(check2 xs);