Problem 1: (2 points each; 8 points total)

State the **type** of each of the following expressions, or if the expression is not valid, state why. For example, the type of the expression `3+4` is `int` and the type of the expression `length` is `'a list -> int`.

(1, [2,3], 4.0)

`int * int list * real`

(reduce op+)

`int list -> int`

explode o rev o implode

*Not valid – implode produces a string but rev requires an `'a list`*

[[[size]]]

`(string -> int) list list list`

Problem 2: (3 points each; 15 points total)

State the **type** of each of the following functions:

fun a (w,h) = w * h * 1

`int * int -> int`

fun f(x) = f(1) + f(2)

`int -> int`

fun f(L,(a,b)) = L = (a,b)

`(''a * ''b) * (''a * ''b) -> bool`

fun f(3,4) = (size,length)

`int * int -> (string -> int) * ('a list -> int)`

fun x y z = (y z) + z

`(int -> int) -> int -> int`
Problem 3: (3 points each, 9 points total)

Edit or rewrite the following functions to make better use of the facilities of ML:

\[
\text{fun } f(a,b,c) = [a-c, a+c] \\
\text{fun } f(a, \_, c) = [a-c, a+c];
\]

\[
\text{fun } f(x,y) = x::y::1::2::[] \\
\text{fun } f(x,y) = [x,y,1,2];
\]

\[
\text{fun } f(n) = \text{if } n = 10 \text{ then true else if } n = 5 \text{ then true else false;}
\]

\[
\text{fun } f2(10) = \text{true} \\
| f2(5) = \text{true} \\
| f2(\_) = \text{false}
\]

Problem 4: (5 points)

Write the \texttt{map} function. The type of \texttt{map} is \('a \rightarrow 'b) \rightarrow 'a \text{ list} \rightarrow 'b \text{ list}

\[
\text{fun } \text{map } F [] = [] \\
| \text{map } F (x::xs) = F(x)::(\text{map } F xs)
\]

Problem 5: (7 points)

Create a function \texttt{abslist}(L) of type \texttt{real list} \rightarrow \texttt{real list} that produces a copy of \texttt{L} with each value in the output list being the absolute value of the corresponding value in the input list. Assume there is NO function like Java's \texttt{Math.abs()} to compute absolute value—do the absolute value computation yourself.

\[
\text{fun } \text{abslist}(L) = \text{map (fn}(x) \Rightarrow \text{if } x < 0.0 \text{ then } \sim x \text{ else } x) \text{ L}
\]

Problem 6: (7 points)

Your instructor suffered the great embarrassment of distributing a version of \texttt{gather} that has a bug: If called with an empty list it should return \([\] \) but in fact it returns \([[]]\).

Example:

\[
- \text{gather}([], 10); \\
\text{val it = [[]]} : \text{int list list}
\]

\text{Change this:}

\[
\text{fun } \text{gather}(L, \text{limit}) =
\]

\text{to this:}

\[
\text{fun } \text{gather}([], \_) = [] \\
| \text{gather}(L, \text{limit}) =
\]
Problem 7: (15 points)

In this problem you are to create TWO functions, `doubler` and `quadrupler`. `doubler` is of type `string list list -> string list list` and "doubles" each letter in the strings. `quadrupler` is of the same type, but quadruples each letter.

```ml
fun doubler L = 
  let
    fun f([]) = []
    |  f(c::cs) = c::c::f(cs)
  in
    (map (map (implode o f o explode))) L
  end

val quadrupler = doubler o doubler
```

Problem 8a: (7 points)

Create a function `genlist` that takes a list of integers and for each integer N in the list, produces a list with N instances of the number 1. You may assume that all the values are non-negative.

```ml
val genlist = map ((map (fn(_) => 1)) o iota)
```

Problem 8b: (7 points)

Create a function `genlist_inv` that performs the inverse operation of `genlist`. The only value appearing in the lists will be the integer 1 (one).

```ml
val genlist_inv = map sum
```

An acceptable answer is to use `length` instead of `sum`,

```ml
val genlist_inv = map length
```

but if you try it out with the interpreter, you'll find that you get an error about type variables not being generalized.

Problem 8c: (2 points)

What is the type of `genlist_inv o genlist o genlist_inv` ?

```ml
int list list -> int list
```
Problem 9: (18 points)

Create a function tacdel(fname) that reads the file named by fname and prints (using the print function) the lines in the file in reverse order, and if a line contains the character "@", the line "<D>" appears in its place.

```ml
fun tacdel(fname) = 
  let
    val bytes = read_all_bytes(fname)
    fun lmapper(s) =
      if member(#"@", explode s) then "<D>" else s
    val lines = map lmapper (rev (split #"\n" bytes))
  in
    print(concat(ien(lines, 1, "\n")))
  end
```