1 What Are Lists?

A list of integers (a value of type `int list`) is either

```
[], or
x :: xs where x : int and xs : int list.
```

And that's it!

[] is pronounced “nil” or “the empty list”; :: is pronounced “cons”. Therefore, the values of type `int list` are lists like these:

```
1 :: (2 :: (3 :: (4 :: [])))
```

This can also be written without the parens as

```
1 :: 2 :: 3 :: 4 :: []
```

because :: is right-associative. For a particular list with a fixed number of elements, you can also write the elements inside of square-brackets separated by commas, as in

```
[1,2,3,4]
```

This is just a convenient notation from SML; it’s short hand for the above form.

The operation on lists is case analysis (and recursion):

```club

case l of
   [] => <branch1>
| x :: xs => <branch2, with (x : int) and (xs : int list) in scope>
```

giving a branch for [] and a branch for ::. In the body of the :: branch, the variable x stands for the first element of the list, and the variable xs stands for the rest of the list.

Note that :: is being used both to create lists, in value declarations, and take lists apart, in the part of the :: branch to the left of =>.