

Week 9 Tutorial Exercises
Reasoning about Haskell Functions

Objectives

The aim of these exercises is to provide you with *experience* of proving some simple facts about relationships that may exist between Haskell functions. The techniques that you will be learning are those that have been explained in the lectures in week 8.

You may not have time to complete all of these exercises during your lab class. If not, you should finish them in your own time, at least before the end-of-semester exam.

Exercise 1 (Properties of Multiplication)

The lectures on this material contained an inductive proof of the following simple property of natural numbers:

$$0 * m = 0$$

where the multiplication function was defined in terms of addition using a Haskell function, the clauses of which were transformed slightly to these equations:

$$m * 0 = 0$$

$$m * (n + 1) = m * n + m$$

The proof depended on these clauses and *just one* trivial fact about addition:

$$j + 0 = j$$

You should follow the example to prove first:

$$1 * m = m$$

and then:

$$m * 1 = m$$

What other simple fact about addition did you use?

Exercise 2 (Properties of Append)

Use the standard definition of the append operator:

$$++ :: [a] \rightarrow [a] \rightarrow [a]$$

$$[] ++ ys = ys$$

$$(x : xs) ++ ys = x : (xs ++ ys)$$

and prove the following properties of the append function:

$$xs ++ [] = xs$$

$$xs ++ (ys ++ zs) = (xs ++ ys) ++ zs$$

Exercise 3 (Membership of Appended Lists)

Prove the following property concerning membership of append lists.

$$\text{elem } z \text{ (xs ++ ys)} = \text{elem } z \text{ xs} \ || \ \text{elem } z \text{ ys}$$

The definition for the `elem` function that you should use is:

```
elem :: Eq => a -> [a] -> Bool
elem x [] = False
elem x (y:ys) = (x==y) || (elem x ys)
```

Exercise 4 (Filtering Appended Lists)

Prove the following property of the append function:

$$\text{filter } p \text{ (xs ++ ys)} = (\text{filter } p \text{ xs}) ++ (\text{filter } p \text{ ys})$$

The definition for the `filter` function that you should use is:

```
filter :: (a -> Bool) => [a] -> [a]
filter p [] = []
filter p (x:xs)
  | p(x) = x:(filter p xs)
  | not(p x) = filter p xs
```
