HW 11 **Prof. Caldwell**
Due: 30 September 2010 **COSC 3015**

On pages 32 and 40 Bird discusses instances of the EQ type class and in Hutton there is discussion on pp 111.

Suppose we wanted to model sets using lists. We might do the following.

```
data Set a = Set [a] deriving (Show)
```

So now, `Set ['a','b','c']` would have type `Set Char` – we have used the name `Set` to name both the type and the constructor.

```
Set :: [a] -> Set a
```

We can define a set equality for lists — the order and multiplicitly of the elements do not contribute to wether two sets are equal or not. We instantiate the Eq type class as follows:

```
instance (Eq a) =>  Eq (Set a) where
 (Set s) == (Set t) = (filter (notin t) s == []) && (filter (notin s) t == [])
     where notin xs x  =  not (elem x xs)
```

Exercise 0.1. Instantiate the finite function type `FinFun` from the previous homework as an instance of the Eq type class. Two finite functions are equal when they are equal as sets of pairs or when their domains are equal (as sets) and when the all the domain elements map to the same range element.