Data Constructors

We can construct more complex data types in Haskell by using the `data` constructor. A simple example is the `Bool` data type that you have already met. This can be defined as

```
data Bool = False | True
```

This says that the type `Bool` is defined as being either `False` or `True` (or is denoted by the `|` symbol). In other words, `Bool` is a type that can only be two values. Of course, data types can have more than two constructors, for example, the question sheet defines the data type of `Colour` and dictates that a `Colour` can only be one of seven colours of the rainbow. Equally if you wanted to create a data type to describe the different positions in Quidditch you could write something like this:

```
data QuidditchPlayer  =  Keeper
                      |  Chaser
                      |  Beater
                      |  Seeker
```

All this is saying is: "If you are a `QuidditchPlayer` you will either be a `Chaser`, a `Keeper`, a `Beater`, or a `Seeker`", or thinking about it the other way round: "If you are a `Keeper` you are a `QuidditchPlayer` (`Keeper :: QuidditchPlayer`)". The same could be done for ice cream flavours. You can define a data type for any discrete collection you want.

Here we are only scratching the surface of the power of making your own data types since there are many more possibilities that will take you beyond these trivial uses.

> ASIDE: Quidditch is a real sport! If you are interested join Bristol Quidditch Club