Function Composition

Function composition is a fancy way of doing one function followed by another. You can think of it like a production line, where you feed the value in the far right and it gets operated on by each function in order from right to left and the final result is spat out.

```
(.) :: (b -> c) -> (a -> b) -> (a -> c)
```

It utilises the `(.)` function. It may seem annoying that it is `(b -> c)` then `(a -> b)` but there is a reason for this. It is because the second function is done first. For example:

```
productionLine :: Int -> Int
productionLine = (+7) . (*2)
```

This function takes in an `Int`, which is fed first to the `(*2)` machine, which spits out and answer that is in turn fed to the `(+7)` machine. So if I feed `3` to the `productionLine` it would get multiplied by `2`, giving `6`, then `6` would have `7` added to it resulting in `13`.