1 COMP304 Week 3 Exercises

You can find these notes at `https://github.com/jb567/comp304-tutorials/`
I'll upload some model solutions at a later point

On Running your haskell code

As has been said in the lectures, you should use the `ghc` haskell toolkit.

In order to test a program, the easiest way is to do the following

```
[user@computer] ~$ ghci haskellfile.lhs
haskellfile> testfunction 1 2 3 4
```

Then, within this environment it can be tested whether or not the function is correct.

Literate Haskell file

Throughtout this course, you will be asked to submit literate haskell files for any assigned work.

Literate programming is the idea that code should be written in long form, with a high level of commenting to maintain proper understanding.

In order to do this literate haskell uses a system of anything that is not in a "comment" is a comment, and all the code is "commented" in

In order to write code you have two options, either
`> codeHere:: Int`
"bird" syntax, this must have a whitespace line between the code and the comments

You can also flip the bird to be `<` to comment it out, this is generally used to show where code that isn't used is held

Similarly, there is a LaTeX variant,

```
\begin{code}
haskellCode :: Int
\end{code}
```

1.1 Ex 1

We have provided the following function to count how many items are in a list recursively.
E.g.

```
count [] => 0
count (1:[]) => 1
count (2:[]) => 1
count (3:2:[]) => 2
```

However, this program doesn't work.

```
count :: [Int] -> Int
count (x:z) = 1 + count z
count (x:[]) = 1
```

Describe a solution to make this function work

1.2 Ex 2

Convert the following function to not use guard statements (Hint: Use pattern matching)

```
fibb :: Int -> Int
fibb a | a == 0 = 0
       | a == 1 = 1
       | otherwise = fibb (a-1) + fib (a-2)
```

1.3 Ex 3

Write a simple guess my number game answer checking function. Give it using both pattern matching and guards.

The number guessing should be hardcoded, and the same for each attempt.

You should return "`You have found my number`" on a correct guess

On an error you should return "`That's not my number`".

For an extension you should return whether or not the fixed number is higher or lower than the guess.

```
guess :: Int -> String
```

1.4 Ex 4

Write a function that returns the nth element in a list

```
findN :: [a] -> Int -> a
```

1.5 Ex 5 - Hard

Write a function to return the maximum value of a list using recursion. Hint: you might need to keep things in the list

```
max :: [Int] -> Int
```

1.6 Ex 6

Write a function to reverse the order of a list

```
reverse :: [a] -> [a]
```