Exercises
Higher-order functions

Exercise 1

Show how the list comprehension

```
[f x | x <- xs, p x]
```

can be re-expressed using the higher-order functions ***map*** and ***filter***.

*** Exercise 2

Without looking at the definitions from the standard prelude, define the following higher-order library functions on lists.

1. Decide if all elements of a list satisfy a predicate:

```
all :: (a -> Bool) -> [a] -> Bool
```

2. Decide if all elements of a list satisfy a predicate:

```
any :: (a -> Bool) -> [a] -> Bool
```

3. Select elements from a list while they satisfy a predicate:

```
 takeWhile' :: (a -> Bool) -> [a] -> [a]
```

4. Remove elements from a list while they satisfy a predicate

```
dropWhile :: (a -> Bool) -> [a] -> [a]
```

**** Exercise 3

Redefine the functions

```
map f
```

and

```
filter p
```

using

```
foldr
```

Exercise 4

Noting that *String* is the same as $[Char]$. Define a function *capitalises*, of type $String \to String$, which takes a list of characters as its argument and returns the same list as its value except that each lower-case letter has been replaced by its upper-case equivalent. Thus,
capitalises ”Bohemian Rhapsody” = ”BOHEMIAN RHAPSODY”.
Hint: Use *toupper* which returns the uppercase of a letter and *map*. This should be written as a function-level definition.

*** Exercise 5

Define a function $squareall :: [Int] \to [Int]$ which takes a list of integers and produces a list of the squares of those integers. For example,
$squareall[6, 1, (-3)] = [36, 1, 9]$.
Hint: Using map, this should be written as a function-level definition.

**** Exercise 6

Define a function *nestedreverse* which takes a list of strings as its argument and reverses each element of the list and then reverses the resulting list. Thus,
nestedreverse [”in”, ”the”, ”end”] = [”dne”, ”eht”, ”ni”].

Hint: Using map, this should be written as a function-level definition.

**** Exercise 7

Define a function $atfront :: a \rightarrow [[a]] \rightarrow [[a]]$ which takes an object and a list of lists and prepends the object at the front of every component list. For example,
$atfront\ 7\ [[1, 2], [], [3]] = [[7, 1, 2], [7], [7, 3]]$.
Hint: Using map, this should be written as a function-level definition.

*** Exercise 8

Define a function lengths which takes a list of strings as its argument and returns the list of their lengths. For example,
lengths [”the”, ”end”, ”is”,”nigh”] = [3, 3, 2, 4].
Hint: Using map, this may be written as an object-level definition.

*** Exercise 9

Using the higher-order function map define a function *sumsq* which takes an integer n as its argument and returns the sum of the squares of the first n integers. That is to say, $sumsq\ n = 1^2 + 2^2 + 3^2 + \ldots + n^2$

**** Exercise 10

The function filter can be defined in terms of *concat* and *map*:

```
filter p = concat.map box where box x = ...
```

Write down the definition of box x

*** Exercise 11

Define a function *wvowel* (without vowels) which removes every occurrence of a vowel from a list of characters.

**** Exercise 12

Define a function wiv (without internal vowels) which takes a list of strings as its argument and removes every occurrence of a vowel from each element. For example,

```
wiv ["the", "end", "is", "nigh"] = ["th", "nd", "s", "ngh"]
```