```
import Data.Char (digitToInt, isAlpha, toLower, toUpper)
```

1. Given that the function *inc* is defined as follows

   $$\begin{aligned} &inc :: Integer \to Integer \\ &inc = (+1) \end{aligned}$$

   evaluate the following expressions

   (a) $inc\ 2014$
   (b) $map\ inc\ [\,]$
   (c) $map\ inc\ [1, 2, 3, 4]$
   (d) $map\ (inc \circ inc)\ [1, 2, 3, 4]$

2. Assume that $sum :: [Integer] \to [Integer]$ is a function that sums a list of integers. Which of the following expressions are valid and why?

   (a) $sum \circ map\ inc = inc \circ sum$
   (b) $sum \circ map\ double = double \circ sum$
   (c) $sum \circ sort = sort \circ sum$
   (d) $sum \circ concat = sum \circ concat \circ map\ sum$

   where $inc = (+1)$, $double = (2*)$, *sort* is function that sorts a list in ascending order and *concat* is function that flat a list of lists.

3. The operator $+\!\!+$ concatenates two lists. Simplify the following expressions

   (a) $[\,] +\!\!+ [\,]$
   (b) $[[\,]] +\!\!+ [[\,]]$
   (c) `"Foo"` $+\!\!+$ `" "` $+\!\!+$ `['B', 'a', 'r']`
   (d) `['H', 'o', 'm', 'e']` $+\!\!+$ `['w', 'o', 'r', 'k']`

4. Let's write a song!

   One little elephant went out to play,
   Upon a spiders web one day.
   He had such enormous fun,
   That he called for another little elephant.

   Two little elephant went out to play,
   Upon a spiders web one day.
   He had such enormous fun,
   That he called for another little elephant.

   Write a function $song :: Int \to String$ such that $song\ n$ is the song when there are n elephants. Assume that $n < 10$.

5. An operator $\oplus$ is said to be associative if $x \oplus (y \oplus z) = (x \oplus y) \oplus z$. Futhermore an element e is said to be an identity of $\oplus$ if $e \oplus x = x \oplus e = x$. Which of the following operators is associative? Which have identity element?

   (a) $+$
   (b) $-$
   (c) $+\!\!+$
   (d) $\circ$

6. Some of the following expressions are ill-typed. Can you find them?

   (a) $inc - 1$

   (b) $inc\ (-1)$

   (c) $inc\ inc\ 0$

   (d) $[[\,]] +\!\!+ [[[\,]]]$

7. Suppose that Alice is using an eager evaluator and the White Rabbit is using a lazy one. How many times the White Rabbit's evaluator evaluate f in computing $head\ (map\ f\ xs)$ when xs is a list of length n? What altertative to $head \circ map\ f$ would Alice prefer?

   The White Rabbit would use $head \circ filter\ p$ to find the first element of a list that satisfy the predicate p. Why Alice would not do the same?

   Instead, Alice would probably prefer something like

```
first                     :: (a → Bool) → [a] → [a]
first p xs | null xs      = error "empty list"
           | p x          = ...
           | otherwise    = ...
  where x = head xs
```

   The function *null* returns *True* on an empty list, and *False* otherwise. Complete the right-hand side of Alice definition.

   What altertative might Alice prefer for $head \circ filter\ p \circ map\ f$.

8. Write a function that capitalize every word of a given text.

9. Error handling is a very important aspect of software development. One way to takle this problem is to be explicit about which computations might fail. The Haskell prelude define the datatype *Maybe* as follows

```
data Maybe a = Nothing | Just a deriving (Eq, Ord)
```

   With the help of this datatype we can write a safe version of the function *head*. (Recall that the function *head* fail when confronted with an empty list.) Let us call *safeHead* to the function

```
safeHead :: [a] → Maybe a
```

   Write a suitable definition of *safeHead*.

10. Consider the following function

```
exp                  :: Integer → Integer → Integer
exp x n | n ≡ 0      = 1
        | otherwise  = x ∗ exp x (n − 1)
```

   What do you think the function *exp* computes? How many multiplications does it take to evaluate $exp\ x\ n$?

11. The International Standard Book Number (ISBN) is an identification system for books. An ISBN number consist of 13 digits. The last digit of the sequence is called the check digit. For instance, the following sequence is an ISBN number

    978-0-393-04002-9

But not all sequences of 13 digits is a valid ISBN number. According to the ISO 2108, "each of the first 12 digits of the ISBN is alternately multiplied by 1 and 3. The check digit is equal to 10 minus the remainder resulting from dividing the sum of the weighted products of the first 12 digits by 10, with one exception. If this calculation results in an apparent check digit of 10, the check digit is 0".

Write a function

$$\mathit{isbn} :: \mathit{String} \rightarrow \mathit{Bool}$$

such that given a string returns *True* if the string represent a valid ISBN, and *False* otherwise.

12. A palindrome is a sequence of symbols that reads the same forward or reversed, after removing punctuation symbols. For example, the phrase

    Madam I'm Adam!

    is a palindrome. Write a function

    $$\mathit{palindrome} :: \mathit{String} \rightarrow \mathit{Bool}$$

    such that returns *True* if the given string is a palindrome and *False* otherwise.

13. Suppose that you want to generate the list of distinct pairs of natural numbers. It doesn't matter in what order the pairs are enumerated as long all they are there. Say whether or not the following definition is correct

    $$\mathit{allPairs} = [(x, y) \mid x \leftarrow [0\,..], y \leftarrow [0\,..]]$$

    If you think it doesn't can you give one definition that is correct?

14. Give a definition of the function

    $$\mathit{disjoint} :: (\mathit{Ord}\ a) \Rightarrow [a] \rightarrow [a] \rightarrow \mathit{Bool}$$

    that takes two lists is ascending order, and determines whether or not they have an element in common.

15. Under what conditions do the following two lists comprenhension deliver the same result?

    $$[e \mid x \leftarrow xs, p\ x, y \leftarrow ys]$$
    $$[e \mid x \leftarrow xs, y \leftarrow ys, p\ x]$$

    Compare the costs of evaluating the two expressions.

16. When the great mathematician Srinivasan Rammanujan was ill a London Hospital, he was visited by the English mathematician G.H. Hardy. Trying to find a subject of conversation, Hardy remarked that he had arrived in a taxi with the number 1729, a rather boring number it seemed to him. Not at all, Rammanujan instantly replied, it is the first number that can be expressed as two cubes in essentially different ways: $1^3 + 12^3 = 9^3 + 10^3 = 1729$. Write a program to find the second such number.

    In fact, define a function tha returns a list of all essentially different quadruples (a, b, c, d) in the range $0 < a, b, c, d \leqslant n$ such that $a^3 + b^3 = c^3 + d^3$.

17. The dual view of lists is to construct them by adding elements to the end of the list:

    data *List a* = *Nil* | *Snoc* (*List a*) *a*

    Snoc is, of course, Cons backwards. With this view of lists [1,2,3] would be represented by

    Snoc (*Snoc* (*Snoc Nil* 1) 2) 3

Exactly the same information is provided by the two views but it is organized differently. Give the definitions of *head* and *last* for the snoc-view of lists, and define two functions

$$\begin{aligned}&\mathit{toList} :: [a] \to \mathit{List}\ a\\&\mathit{fromList} :: \mathit{List}\ a \to [a]\end{aligned}$$

for converting efficiently from one view of lists to the other. (Hint. *reverse* is efficient, taking linear time to reverse a list.)

18. How much space is required to evaluate *length xs*? Consider the following alternative definition of *length*:

$$\begin{aligned}&\mathit{length} :: [a] \to \mathit{Int}\\&\mathit{length}\ \mathit{xs} = \mathit{loop}\ (0, \mathit{xs})\\&\quad \textbf{where}\ \mathit{loop}\ (n, [\,]) = n\\&\quad\qquad \mathit{loop}\ (n, x : \mathit{xs}) = \mathit{loop}\ (n+1, \mathit{xs})\end{aligned}$$

Does the space requirement change? Does it change if we switched to eager evaluation?

19. The prelude function *take n* takes the first *n* elements of a list, while *drop n* drops the first *n* elements. Give recursive definition for these two functions.

Which of the following equations are valid for all integers *m* and *n*? You don't have to justify your answers, just try to understand what they claim to say.

(a) $\mathit{take}\ n\ \mathit{xs} \mathbin{+\!\!+} \mathit{drop}\ n\ \mathit{xs} = \mathit{xs}$

(b) $\mathit{take}\ m \circ \mathit{drop}\ n = \mathit{drop}\ n \circ \mathit{take}\ (m+n)$

(c) $\mathit{take}\ m \circ \mathit{take}\ n = \mathit{take}\ (m\ \text{`}\mathit{min}\text{`}\ n)$

(d) $\mathit{drop}\ m \circ \mathit{drop}\ n = \mathit{drop}\ (m+n)$

20. Here are some equations; at least one of them is false. Which are the true ones, and which are false? Once again, you do not have to provide any justification for your answers, the aim is just to look at some equations and appreciate what they are saying.

(a) $\mathit{map}\ f \circ \mathit{take}\ n = \mathit{take}\ n \circ \mathit{map}\ f$

(b) $\mathit{map}\ f \circ \mathit{reverse} = \mathit{reverse} \circ \mathit{map}\ f$

(c) $\mathit{map}\ f \circ \mathit{sort} = \mathit{sort} \circ \mathit{map}\ f$

(d) $\mathit{map}\ f \circ \mathit{filter}\ p = \mathit{map}\ \mathit{fst} \circ \mathit{filter}\ \mathit{snd} \circ \mathit{map}\ (\mathit{fork}\ (f, p))$

(e) $\mathit{filter}\ (p \circ g) = \mathit{map}\ (\mathit{invertg}) \circ \mathit{filter}\ p \circ \mathit{map}\ g$

(f) $\mathit{reverse} \circ \mathit{concat} = \mathit{concat} \circ \mathit{reverse} \circ \mathit{map}\ \mathit{reverse}$

(g) $\mathit{filter}\ p \circ \mathit{concat} = \mathit{concat} \circ \mathit{map}\ (\mathit{filter}\ p)$

In the fifth equation assume that *invertg* satisfies $\mathit{invertg} \circ g = \mathit{id}$. The function *fork* in (d) is defined by

$$\begin{aligned}&\mathit{fork} :: (a \to b, a \to c) \to a \to (b, c)\\&\mathit{fork}\ (f, g)\ x = (f\ x, g\ x)\end{aligned}$$

21. Define *unzip* and *cross* by

$$\begin{aligned}&\mathit{unzip} = \mathit{fork}\ (\mathit{map}\ \mathit{fst}, \mathit{map}\ \mathit{snd})\\&\mathit{cross}\ (f, g) = \mathit{fork}\ (f \circ \mathit{fst}, g \circ \mathit{snd})\end{aligned}$$

What are the types of these functions?

Prove by simple equational reasoning that

$$\mathit{cross}\ (\mathit{map}\ f, \mathit{map}\ g) \circ \mathit{unzip} = \mathit{unzip} \circ \mathit{map}\ (\mathit{cross}\ (f, g))$$

You can use the functor laws of *map* and the following rules:

(a) $\mathit{cross}\ (f, g) \circ \mathit{fork}\ (h, k) = \mathit{fork}\ (f \circ h, g \circ k)$

(b) $\mathit{fork}\ (f, g) \circ h = \mathit{fork}\ (f \circ h, g \circ h)$

(c) $\mathit{fst} \circ \mathit{cross}\ (f, g) = f \circ \mathit{fst}$

(d) $\mathit{snd} \circ \mathit{cross}\ (f, g) = g \circ \mathit{snd}$

22. Continuing from the previous exercise, prove that

$$\mathit{cross}\ (f, g) \circ \mathit{cross}\ (h, k) = \mathit{cross}\ (f \circ h, g \circ k)$$

We also have $\mathit{cross}\ (\mathit{id}, \mathit{id}) = \mathit{id}$ (Why?). So it looks like *cross* has functor like properties, except that it takes a pair of functions. Yes, it's a *bifunctor*. That suggest a generalization

```
class Bifunctor f where
  bimap :: (a → b) → (c → d) → f a c → f b d
```

The arguments to *bimap* are given one by one rather than paired. Express *cross* in terms of *bimap* for the instance *Pair* of *Bifunctor*, where

```
type Pair a b = (a, b)
```

Now consider the datatype

```
data Either a b = Left a | Right b
```

Construct the instance *Either* of *Bifunctor*.