**DelftX:** FP101x Introduction to Functional Programming

**<DISCLAIMER>**
**For this lab, we provide a template file which contains the signatures of the functions to be implemented, as well as some definitions that will be needed in order to answer the final question. We suggest you make use of this template when solving the exercises.**
**</DISCLAIMER>**

## EXERCISE 0 (1 point possible)

Using a list comprehension, define a function that selects all the **even** numbers from a list.

```
evens :: [Integer] -> [Integer]
```

*Example:* evens [2, 5, 6, 13, 32] = [2, 6, 32]

What is the value of the expression: `sum . evens $ [827305 .. 927104]`

43772529500

43772529500

*You have used 0 of 1 submissions*

## EXERCISE 1 (1 point possible)

What is the value of the expression: `sum . evens $ []`

0

0

*You have used 0 of 1 submissions*

## EXERCISE 2 (1 point possible)

What is the value of the expression: `sum . evens $ [1,3..]`

- `0`
- `[]`
- `[2, 4..]`
- (selected) The computation does not terminate

*You have used 0 of 1 submissions*

## EXERCISE 3 (1 point possible)

Using a list comprehension, define a function squares that takes a non-bottom Integer `n >= 0` as its argument and returns a list of the numbers `[1..n]` squared.

*Example:*

```
squares 4 = [1*1, 2*2, 3*3, 4*4]
squares 0 = []
```

Choose a **valid** type signature for squares:

- `Num -> [Num]`
- (selected) `Integer -> [Integer]`
- `a -> [a]`
- `Integer a => a -> [a]`

*You have used 0 of 1 submissions*

## EXERCISE 4 (1 point possible)

Using the `squares` function that you have implemented, we can define a function `sumSquares :: Integer -> Integer` as follows:

```
sumSquares n = sum (squares n)
```

What is the value of: `sumSquares 50`?

42925

$$42925$$

**Answer:** 42925

*You have used 0 of 1 submissions*

---

## EXERCISE 5 (1 point possible)

Modify the previous definition of `squares` such that it now takes two non-bottom Integer arguments, `m >= 0` and `n >= 0` and returns a list of the `m` square numbers that come after the first `n` square numbers.

*Example:*

```
squares' 4 2 = [3*3, 4*4, 5*5, 6*6]
squares' 2 0 = [1*1, 2*2]
squares' 0 2 = []
squares' 0 0 = []
```

We can define a new `sumSquares'` function as follows:

```
sumSquares' x = sum . uncurry squares' $ (x, x)
```

What is the value of: `sumSquares' 50`?

295425

$$295425$$

*You have used 0 of 1 submissions*

---

## EXERCISE 6 (1 point possible)

Using the `squares'` function that you've defined in the previous exercise, what is the value of: `sum $ squares' 10 0`:

385

385

*You have used 0 of 1 submissions*

## EXERCISE 7 (1 point possible)

Again using the `squares'` function that you've defined previously, what is the value of: `sum $ squares' 0 10`:

0

0

*You have used 0 of 1 submissions*

## EXERCISE 8 (1 point possible)

Using a list comprehension, define a function `coords :: Integer -> Integer -> [(Integer, Integer)]` that returns a list of all coordinate pairs on an `[0..m] × [0..n]` rectangular grid, where `m` and `n` are non-bottom Integers `>= 0`.

*Example:*

```
coords 1 1 = [(0,0), (0,1), (1,0), (1,1)]
coords 1 2 = [(0,0), (0,1), (0,2), (1,0), (1, 1), (1, 2)]
```

What is the value of: `foldr (-) 0 . map (uncurry (*)) $ coords 5 7`

-60

−60

*You have used 0 of 1 submissions*

## EXERCISE 9 (1 point possible)

The expression `a f b g c` is equivalent to:

- a (f b) (g c)
- (a f b) (g c)
- ((a f) b) g c
- (a f) (b g) c

*You have used 0 of 1 submissions*

## EXERCISE 10 (1 point possible)

The type `a -> f -> b -> g -> c` is equivalent to:

- a -> (f -> b) -> (g -> c)
- a -> (f -> b -> (g -> c))
- (a -> f -> b) -> (g -> c)
- (a -> f) -> (b -> g) -> c

*You have used 0 of 1 submissions*

## EXERCISE 11 (1 point possible)

The type `(a, f, b, g, c)` is equivalent to:

- (a, (f, b), (g, c))
- ((a, f, b), (g, c))

- ((a, f), (b, g), c)
- ((a, f, b, g, c))

*You have used 0 of 1 submissions*

## EXERCISE 12 (1 point possible)

The expression `(a, f, b, g, c)` is equivalent to:

- (a, (f, b), (g, c))
- ((a, f, b), (g, c))
- ((a, f), (b, g), c)
- ((a, f, b, g, c))

*You have used 0 of 1 submissions*