In contrast to the previous exercise on lists, try to implement these functions using list comprehensions (e.g., `[ a+1 | a <- as ]`) as well as list ranges: `[1..n]`, `[1,5,..,n]`. **Do not use explicit recursion!**

Note that many of these functions are available in the standard library,[1] but the goal of this exercise is to practice by implementing them from scratch.

- In mathematics, the factorial of a non-negative integer number `n` is defined recursively as follows:

$$factorial\ (n) = \begin{cases} 1 & \text{, if } n = 0 \\ n * factorial\ (n-1) & \text{, if } n > 0 \end{cases}$$

  Alternatively, we can also express it as: $factorial\ (n) = 1 * 2 * \ldots * (n-1) * n$

  Write a function `factorial :: Integer -> Integer`, such that `factorial n` is the factorial of `n`. If `n` is a negative number, `factorial n` should result in 1. **Hint:** You can use the standard library function `product`, to compute the product of every element in a list.

```
Main> factorial 5
120

Main> factorial 0
1

Main> factorial (-10)
1
```

- Write a function `myRepeat :: Int -> Int -> [Int]` such that `myRepeat n x` returns a list with `n` times the number `x`. If `n` is less than zero return the empty list.

```
Main> myRepeat 4 5
[5,5,5,5]

Main> myRepeat (-1) 5
[]

Main> myRepeat 0 5
[]
```

- Write a function `flatten :: [[Int]] -> [Int]` which converts a list of lists to a single list.

```
Main> flatten [[1,2],[3,4],[5,6]]
[1,2,3,4,5,6]
```

[1] For example, see module `Data.List`, which can be found on `http://downloads.haskell.org/~ghc/7.6.3/docs/html/libraries/base/`.

```
Main> flatten []
[]
```

- Write a function `range :: Int -> Int -> [Int]` which returns a list of the consecutive numbers between the two given numbers, both numbers included. If the first number is greater than the second you should return the empty list.

```
Main> range 1 10
[1,2,3,4,5,6,7,8,9,10]

Main> range (-10) (-5)
[-10,-9,-8,-7,-6,-5]

Main> range 10 1
[]
```

- Write a function `sumInts :: Int -> Int -> Int`, which takes an integer `low` and an integer `high` and computes the sum:

$$\texttt{sumInts low high} = \texttt{low} + (\texttt{low} + 1) + (\texttt{low} + 2) + \ldots + (\texttt{high} - 1) + \texttt{high}$$

If $\texttt{low} > \texttt{high}$ then the sum should be zero. **Hint:** You can use the standard library function **sum**, to compute the sum of every element in a list.

```
Main> sumInts 3 5
12

Main> sumInts 5 3
0

Main> sumInts 5 5
5
```

- Write a function `removeMultiples :: Int -> [Int] -> [Int]` which removes all multiples of a number from the list. Use `n `mod` d` or `mod n d`. Assume that the first argument will never be zero.

```
Main> removeMultiples 2 (range 1 10)
[1,3,5,7,9]

Main> removeMultiples 5 []
[]
```