Programming Paradigms 2013-2014
Reeks 2 - Haskell
Recursion and Higher Order Functions

Naomi Christis
Office G028
naomi.christis@uantwerpen.be

1. Define `takeWhile2`, which selects elements from a list while they satisfy a predicate.
2. Define the function `replicate` which will produce a list of identical elements (`replicate :: Int -> a -> [a]`).
3. Check whether all members of a list of integers are even, or all are odd.
4. Find all the digits in a string.
5. A pythagorean triad is a triple (x, y, z) of positive integers such that $x^2 + y^2 = z^2$. Using a list comprehension, define a function `triads :: Int -> [(Int, Int, Int)]` that maps a number n to the list of all triads with components in the range $[1..n]$.
6. A positive integer is perfect if it equals the sum of all of its factors, excluding the number itself. Using a list comprehension; define a function `perfects :: Int -> [Int]` that returns the list of all perfect numbers up to a given limit.
   (e.g. `perfects 500` gives $[6, 28, 496]$)
7. Express the comprehension `[f x | x <- xs, p x]` using the functions map and filter.
8. Define `zipWith2` which takes a function and two lists as parameters and then joins the two lists by applying the function between the corresponding elements.
9. Show how the following functions could be redefined using `foldr`:
   (a) `length`
   (b) `reverse`
   (c) `map f`
   (d) `filter p`
10. Define the following functions:
    (a) the function `curry` that converts a function on pairs into a curried function
    (b) the function `uncurry` that converts a curried function with two arguments into a function on pairs

    Define your own version of `fst` and `snd` using the `curry` and `uncurry` functionality (hint: `const` and `flip`).