CvP - Programming Assignment 2

- Deadline: November 9, at the start of the werkcollege.
- Submit your solution electronically via liacscvp2018@gmail.com
- Clearly state your name and student number your solution file.
- Put your functions inside a .hs file and hand in this file.
- In this assignment you will practice Haskell programming. Your task is to define and test functions described below in this document.
- You may use Haskell specific features to produce the behaviour.
- For good tutorials check: http://learnyouahaskell.com/chapters

Question 1 Define and test a Haskell function `drop` that drops all occurrences of an element from a list. The expected behaviour is shown below.

```
ghci> drop ’b’ [’a’,’b’,’b’,’c’]
[’a’,’c’]
```

Question 2 Define and test a Haskell function `intersection` that takes two lists and returns their intersection. If an element appears more than once in one list or the other, it should appear in the output list only once. The expected behaviour is shown below.

```
ghci> intersection [1] [1]
[1]
ghci> intersection [1,2] [1]
[1]
ghci> intersection [3,3,4] [3,3,4,4,5,6]
[3,4]
```

Question 3 Define and test a Haskell functions `even` and `odd` that takes a list and returns the list of elements at the respective even and odd positions in the list. The expected behaviour is shown below.

```
ghci> even [1,2,3,4,5,6]
[2,4,6]
ghci> odd [1,2,3,4,5,6]
[1,3,5]
```

Question 4 Define and test a Haskell function `zip` that takes two lists and pairs up corresponding elements of the two lists. If one of the lists is longer than the other, the extra elements are ignored. The expected behaviour is shown below.

```
ghci> zip (odd [1,2,3,4,5,6]) (even [1,2,3,4,5,6])
[1,2,3,4,5,6]
ghci> zip [1,2,3] [4,5,6]
[1,4,2,5,3,6]
ghci> zip [1,2,3] [4]
[1,4,2]
```

Question 5 Define and test a Haskell function `flatten` that takes a nested list of integers of arbitrary depth and returns a list of all its primitive elements and the ones of its sub lists. The expected behaviour is shown below.

```
ghci> flatten []
[]
ghci> flatten [I 1]
[I 1]
ghci> flatten [T [I 1,I 2],I 3]
[I 1,I 2,I 3]
```

where `T` is a nested list of integers `I` of arbitrary depth.

Question 6 Write a Haskell function

```
indivisible :: Integer -> [Integer] -> [Integer]
```

such that `indivisibles n lst` returns the list of elements in `lst` that are not divisible by `n`. So, `indivisible 3 [1..10]` returns `[1, 2, 4, 5, 7, 8, 10]`.

Question 7 Use `indivisible` to write a Haskell function

```
eratosthenes :: [Integer] -> [Integer]
```

such that `eratosthenes lst` returns the list res such that

1. every element in `res` is an element in `lst`,
2. `res` does not contain 1 as an element, and
3. no element in `res` is divisible by any other element in `res`.

So, `eratosthenes [1..20]` returns `[2,3,5,7,11,13,17,19]`.

Question 8 Use `eratosthenes` to write a Haskell function

```
primes :: Integer -> [Integer]
```

such that `primes n` returns the prime numbers up to `n`. So, `primes 30` returns `[2,3,5,7,11,13,17,19,23,29]`.