universität innsbruck

Functional Programming	SS 2020	LVA 703025

Exercise Sheet 4, 10 points — Deadline: Monday, March 30, 2020, 10am

- Watch the lecture of week 4 and study part 3 of the slides up to slide no. 43[1].
- Please write all the Haskell code into a single .hs-file and upload it in OLAT.
- Exercise 4.2 can be added as a comment to the .hs-file.
- You can use the template .hs-file that is provided on the proseminar page[2].
- Your .hs-file should be compilable with ghci.
- Don't forget to mark your completed exercises in OLAT.

Exercise 4.1 *Booleans* **3 p.**

Write code for the function `boolFun :: Bool -> Bool -> Bool -> Bool` as specified by

x	y	z	boolFun x y z
False	False	False	False
False	False	True	False
False	True	False	True
False	True	True	True
True	False	False	False
True	False	True	True
True	True	False	True
True	True	True	False

in three different ways:

1. using `if then else` and `&&`, `||`, `not`; (1 point)
2. as before but without using `if then else`; (1 point)
3. using pattern matching. (1 point)

Exercise 4.2 *Enumerations* **4 p.**

1. As of the 2019 elections the Austrian National Council houses 5 parties. In total there are 183 members of parliament (MPs), which are distributed as follows: Oevp - 71 MPs, Spoe - 40 MPs, Fpoe - 31 MPs, Gruene - 26 MPs and Neos - 15 MPs.

   Since no party holds more than half of the seats, there is need to form a coalition between at least two parties.

   Write a function `coalition :: Party -> Party -> Bool` that takes two parties and returns whether they can form a government together or not (purely mathematically, by holding at least 92 seats).

   As intermediate steps, define an enumeration type `Party` and a function `mps :: Party -> Integer`.

   (2 points)

[1] http://cl-informatik.uibk.ac.at/teaching/ws19/fp/slides/03x1.pdf
[2] http://cl-informatik.uibk.ac.at/teaching/ss20/fp/index.php#exercises

2. Define a type `Season` with constructors for the four seasons and write a Haskell function `daysInSeason` that returns the number of days in a given season. Make sure to also write down the type signature. (Assume the following durations: spring - 93, summer 94, fall - 90, winter - 89.)
   (a) Using pattern matching (1 point)
   (b) Using if-then-else and an `Eq` **`instance`** for `Season` (1 point)

Exercise 4.3 *Polymorphism* 2 p.

1. Write two different functions `foo` and `bar` of type `a -> a -> a`. Here, different means that for some input values `x` and `y` the result of `foo x y` is different from the result of `bar x y`. (1 point)
2. Is there a difference between the type signatures `a -> b -> a` and `c -> a -> c`? (1 point)

Exercise 4.4 *A simple recursive function* 1 p.

1. Implement the following functions g and c in Haskell:

$$g(n) = \begin{cases} 0 & \text{if } n < 0 \\ n/2 & \text{if } n \text{ is divisible by } 2 \\ 3n+1 & \text{otherwise} \end{cases}$$

$$c(x) = \begin{cases} 0 & \text{if } x \leq 1 \\ 1 + c(g(x)) & \text{otherwise} \end{cases}$$

   You can reuse the function `isDivisible` from the previous exercise sheet. Explain what the function c is counting? (Write a short comment about it in your Haskell code) (1 point)
2. BONUS: Is there an `n` where `c n` does not terminate[3]? (0 points)[4]

[3] See also `https://en.wikipedia.org/wiki/Collatz_conjecture`
[4] You can't earn any points with this exercise. Follow Paul Erdős' advice (see wiki article) and don't waste too much time on it ;)