universität innsbruck

Functional Programming | WS 2019/2020 | LVA 703024+703025

Exercises 2 | Deadline: October 16, 2019, 6am

- Please write all your Haskell functions from this exercise sheet into a single .hs-file and upload it in OLAT.
- You can use a template .hs-file that is provided on the proseminar page.
- The file should compile with ghci.
- Once the file has been uploaded, it cannot be changed or resubmitted!

Exercise 2.1 *Parsing expressions* **4 p.**

Draw the abstract syntax tree of the following expressions according to the parsing algorithm described in the lecture, cf. part 2, 20–25:

1. `7 * 8 + 6 / 9` (1 point)
2. `7 - 8 - 6` (1 point)
3. `x || a == 3 && y` (1 point)
4. `7 + my_fun 9 (3 * 4) 90` (1 point)

Exercise 2.2 *Functional Decomposition* **2 p.**

The rulebook of a (fictional) tabletop RPG explains how damage is calculated: *The attacker adds their weapon's damage to the base-attack value to obtain total attack. The defender multiplies their armor's defense with the base-defense value to obtain total defense. The damage dealt is calculated by subtracting total defense from total attack, but it is at least zero.*
Consider functional decomposition as discussed in the lecture and write a function `calculateDamage`, that calculates damage for given attack and defense values, using at least 3 different and meaningful functions. You may do this in pseudocode or Haskell, but give type declarations in Haskell for all functions used. If you do it in Haskell, use the following functions for the *base-attack and base-defense values*:

```
baseAttack :: Integer
baseAttack = 4
baseDefense :: Integer
baseDefense = 3
```

(2 points)

Exercise 2.3 *Divide-and-Conquer* **4 p.**

1. Squaring a natural number can be solved by divide-and-conquer by noting that $n^2 = (n-1)^2 + 2n - 1$. That is, the task of squaring a positive natural number n can be *conquered* by first *dividing* it into squaring $n-1$ and then adding $2n-1$ to the result.

   Evaluate 3^2 by hand (the outcome should be 9) using that divide-and-conquer method and write the corresponding Haskell function `square :: Integer -> Integer`. (2 points)

2. The number of ways to choose a subset of k elements from a fixed set of n elements is known as the binomial coefficient and denoted by $\binom{n}{k}$. It can be computed on the one hand using functional decomposition as $\frac{n!}{k!(n-k)!}$, and on the other hand also by divide-and-conquer using $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$. Write the corresponding Haskell functions `binom_fd, binom_dc :: Integer -> Integer -> Integer`. Use the factorial function `fact` as on the slides for the former. For the latter first think about when and how to stop dividing, by first computing $\binom{3}{2}$ by hand (the outcome should be 3). (2 points)