universität innsbruck

Functional Programming	SS 2020	LVA 703025
Exercise Sheet 1, 10 points		Deadline: Monday, March 9, 2020, 10am

- Please write the Haskell function from exercise 1.3 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.
- Don't forget to mark your completed exercises in OLAT.

Exercise 1.1 *Basic Haskell Expressions* 1 p.

Evaluate the following expressions in `ghci`.

```
9 + 3
5 / 2
(10 * 100) - 999
10 * 100 - 999
10 * (100 - 999)

1 == 1
1 == 0
5 /= 5
5 /= 10
5 <= 10
5 >= 10

not False
True && False
True && True
False || True
not (True && True)
not (1 /= 1)

if True then 5 else 10
if 10 <= 5 then 5 else 10
```

To what logical operations are the Haskell operations `not`, `||` and `&&` equivalent?

Exercise 1.2 *Functions* 4 p.

Consider the following Haskell definitions.

```
double x = x + x
doubleSmallNum x = if x <= 100 then double x else x
subTwoY x y = x - (2 * y)

f x = if x <= 0 then 0 else 1 + f (x - 1)
```

1. Evaluate the expression `doubleSmallNum 5` step by step on paper. (1 point)
2. Evaluate the expression `subTwoY 8 (doubleSmallNum 111)` step by step on paper. (1 point)
3. Evaluate the expression `f 3` step by step on paper. (2 points)

Exercise 1.3 *Implementing Functions* **2 p.**

1. Implement the following function in Haskell:

$$h(x, y) = \sin x \sin y + \cos(x + y)$$

(1 point)

2. Implement the following function in Haskell:

$$g(x) = \begin{cases} \exp(x) & \text{if } x \geq 0 \\ \cos(x) & \text{if } x < 0 \end{cases}$$

(1 point)

Exercise 1.4 *Evaluation Strategies* **3 p.**

1. The function `subTwoY x y = x - (2 * y)` subtracts two times the value of `y` from `x`, and the function `double x = x + x` doubles the value of `x`. Stepwise evaluate with pen and paper the expression `subTwoY (double 5) 4` according to call-by-value, call-by-name and the lazy strategy until a normal form is reached, cf. part 2, 13–14. (2 points)
2. Consider the functions `foo x y = xPowYPowZ x y x` and `xPowYPowZ x y z = x ^ y ^ z`. Stepwise evaluate with pen and paper the expression `foo (1+1) 3` according to call-by-value/strict, call-by-name/non-strict and call-by-need/lazy until a normal form is reached. (1 point)