Functional Programming for BDA - List 0

Marcin Michalski, FPAM WUST 2021/2022

Before proceeding with the exercises below, download Haskell Platform or at least Glasgow Haskell Compiler (briefly: GHC).

Exercise 1. Play with the command `:t` and check the type of various expressions, for example:

a) 98,

b) $5 + 3$,

c) (+),

d) (2^),

e) (truncate pi),

f) (sqrt 25),

g) (round 1.8),

h) False,

i) $(4 < 5)$,

j) (> 45),

k) ... .

Explain the results.

Exercise 2. Calculate the following expressions in GHC: 2^3^2, 2^(3^2), (2^3)^2. Determine the associativity of ^ using the command `:i`. In a similar way check some other known to you 2-argument functions.

Exercise 3. Enter $f\ x\ y = x + 2{*}y$ and $g = f\ 3$ in GHC. How does the function g work?

Exercise 4. Enter $x = [1, 2, 3]$, $y = [1, 3, ..8]$ and $z = [1..]$ in GHC.

a) Test functions `head`, `init`, `last`, `tail` on x and y, e.g. `head y`.

b) Test ++ function which concatenates two lists into one, e.g. `u=y++z`.

c) Test function `take n` on z, where n is a natural number. What happens if you try to print z (`show` z) or take the last element of it?

d) Enter the command `:show bindings` and explain what do you see.

Exercise 5. Implement the function `signum :: Double -> Int` using patter matching in several different ways.

Exercise 6. Consider the following function:

```
tell_bmi :: Double -> Double -> String

tell_bmi h w = if w/(h^2)<=18.5 then "you are underweight" else
if w/(h^2)<25 then "you have normal weight" else "you are overweight"
```

Rewrite this function using pattern matching and expressions `let` and `where`.

Exercise 7. Test the following lines of code:

```
True || (length ([1..] > 0))
```

and

```
(length ([1..] > 0)) || True.
```

Explain the difference in behavior.

Exercise 8. Write a function that mimics `if ... then ... else ...` expression.