Functional Programming – Series 1

Exercise 1. Introduction-exercise. Type in the next function

$$f\ x \quad = \quad 2x^2 + 3x - 5$$

and evaluate it for a few values of x.

Remark. From now on every function definition should be preceded by mentioning its type.

Exercise 2. Import the module *Data.Char*:

import *Data.Char*

Now the following functions are avaiable:

ord :: *Char* → *Int*
chr :: *Int* → *Char*

which translate a character into its code and back.

- Use *ord* and *char* to define a function *code* which changes a letter (including capital letters) by cyclicly shifting it three positions further in the alphabet, i.e., after 'z' one should continue with 'a' again.

  For example:

  code `'a'` = `'d'`
  code `'P'` = `'S'`
  code `'y'` = `'b'`

  The function *code* should leave all other characters (digits, spaces, etc.) unchanged. Hint: the relations $<, \leqslant, \geqslant, >$ also work for characters.

- Evaluate the expressions

  map code `"hello"`
  map code `"Tomorrow evening, 8 o'clock in Amsterdam"`

  (*map* applies the function *code* to all characters in a string).

- Generalise your function *code* in such a way that it can be given a number n as additional argument, which indicates how many positions the character has to be shifted (note that above $n = 3$). Note that the order in which the arguments are given to *code* is relevant for using the function *map* with the generalised coding function.

Exercise 3. Define *recursively* a function *amount* which calculates how much money you have after n years, if you start with an amount a and receive r percent of interest per year. You have to take into account that you will have "interest over interest", where the interest only has to be computed once per year.

Exercise 4. Define two functions *root1* and *root2* which determine the roots of a quadratic equation of the form

$$ax^2 + bx + c = 0$$

where a, b, c are given, and $a \neq 0$. If the discriminant is negative, your function should give an error, to be defined as follows:

error `"negative discriminant"`

Write a function *discr* which calculates the discriminant and which can be used in the functions *root1* and *root2*.

Test your funktions for a number of values of a, b, c.

Exercise 5. A second order polynome is an expression of the form

$$ax^2 + bx + c$$

(assume $a \neq 0$).

- Write a function *extrX* which calculates the value of x at which the polynome has its extreme value.
- Write a function *extrY* which calculates this extreme value.

Exercise 6. Write recursive definitions for the following functions on lists (give the type for every function you define):

- *mylength* for the length of a list,
- *mysum* which adds the elements in a list of numbers,
- *myreverse* which reverses the order of the elements in a list,
- *mytake* which gives the first n elements of a list (in case n is greater than the length of a list, the whole list should be delivered),

– *myelem* which determines whether a given element is in a list,

– *myconcat* which glues together a list of lists into one long list,

– *mymaximum* which yields the maximum of a list of numbers,

– *myzip* which transforms two lists into a list of pairs (the shortest of the two lists determines the length of the resulting list).

Exercise 7. A sequence of numbers is *arithmetic* if, starting from some initial number a, every next number in the sequence can be determined from the previous number by adding a fixed difference d.

– Write a recursive function r which generates the arithmetic sequence starting with a, and using difference d. Note that the type of r is:

$$r :: Num\ a \Rightarrow a \to a \to [a]$$

– Write a function *r1* which selects the n-th number from the sequence above (hint: use the function r),

– Let i and j be two indices. Write a function *total* which calculates the sum of the i-th element upto (and included) the j-th element.

Exercise 8.

– Write a function *allEqual* which determines whether all elements in a list are equal.

– Write a function `isAS` which checks whether a sequence is arithmetical (hint: use the function *allEqual*).

Exercise 9. A matrix can be defined as a list of lists of numbers, where the inner lists are the rows of the matrix. Write for the following cases a function which:

a. checks whether all rows in a matrix are equally long,

b. yields the list of totals of every row in a matrix,

d. transposes a matrix, i.e., every n-th row is transformed into the n-th column,

b. yields the list of totals of every column in a matrix.