Fakultät für Informatik und Mathematik
Prof. Dr. Martin Griebl
Summer semester 2023

Exercises for Lecture: Functional Programming
Exercise Sheet 2 (Partial Application in Java, First Steps in Haskell)

Problem 1 (Functions with Several Arguments)

Consider the interfaces `Function` for functions with one argument and `BiFunction` for binary functions (functions with two arguments) from the package `java.util.function`.

(a) Compute the value of `3+5` in two different ways using `Function` and `BiFunction`:

1. implement a function using `Function` that adds 3 to its argument,
2. implement a function for binary addition using `BiFunctino`.

Use these two functions to compute `3+5` by suitable function applications.

(b) When a function with two arguments is given its first argument, the result is another function with one argument (the remaining argument).

This can be expressed by an interface for binary functions that extends the interface `Function` in the following way:

```
interface MyBiFunction<A,B, R> extends Function<A, Function<B,R> > { ... }
```

Implement the `apply` method from `Function` as a default method in `MyBiFunction`.

Example: When `add` represents binary addition, then 3+5 can not only be expressed by `add.apply(3,5)` but also by `add.apply(3).apply(5)`.

(c) Implement function *map* using `MyBiFunction` in a class `Map`. This function applies a unary function to a list (more generally, a collection or, in Java, any `Iterable`). It corresponds to the operator α (apply-to-all) in John Backus's language FP.

Test your implementation of `Map` with a function to square values and a short list of numbers.

Problem 2 (Polynomials of Second Degree)

Consider polynomial functions $a \cdot x^2 + b \cdot x + c$ with $a \in \mathbb{R}\backslash\{0\}, b, c, x \in \mathbb{R}$.

(a) Define a function `optimalX` in Haskell that takes a, b and c as inputs and computes the x coordinate for which the function value of the polynomial is minimal or maximal, respectively (dependent on a), i.e., `optimalX a b c` returns the x coordinate of the apex of the parabola defined by the polynomial's graph.

(b) Define a function `optimalF` in Haskell that computes from given a, b and c the corresponding optimal function value, i.e., the y coordinate of the apex.

(c) Define a function `optimal` in Haskell that takes a, b and c as inputs and returns the x and y coordinates of the parabola's apex as a pair.

Problem 3 (`Int`, `Integer`, `Double`)

In Haskell one can define lists of numbers by giving an enumeration, e.g., `[1..10]`. It is also possible to specify a stride (step unit) for the enumeration by giving the value of the second entry, e.g., `[2,4..10]`. Additionally, it is possible to define infinite lists by omitting the upper bound, e.g., `[2,4..]`.

Evaluate the following expressions using GHCi (use `Ctrl+C` to abort a computation if necessary):

(a) `[0::Integer,2^60 ..]`

(b) `[0::Int,2^28 ..]`

(c) `[0::Int,2^31 ..]`

(d) `[0::Int,2^32 ..]`

(e) `[0::Int,2^60 ..]`

(f) `[0::Int,2^63 ..]`

(g) `[0::Int,2^64 ..]`

(h) `[0::Double,10^305 ..]`

(i) `[0.0,0.1 .. 1.0]`

What do you notice? How do you explain the observed behavior?

Due date: Tuesday, May 02, 2023 at 16:00

Upload your solution for Problem 2 as Haskell source file (`*.hs`) to the `Submissions` folder for this exercise on Stud.IP.