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

Exercises for Lecture: Functional Programming
Exercise Sheet 13 (Type Inference, Parsing)

Problem 1 (Type Inference)

Derive the most general type of the following function:

```
len xs = if null xs then 0 else 1 + len (tail xs)
```

For every subexpression, state its type currently known at the time it was derived. If necessary, also state the substitutions that needed to be carried out. You can assume that 0::`Int`, 1::`Int`, `(+)`::`Int`→`Int`→`Int`, `null`::$\forall\alpha.[\alpha] \to$ `Bool`, and `tail`::$\forall\alpha.[\alpha] \to [\alpha]$.

Problem 2 (Parser Combinators)

In Stud.IP you can find the file `Expr.hs`, which defines the abstract syntax of a simple language with λ-expressions together with an interpreter for the language (function `run`). In `Parser.hs` you can find some basic definitions for a parser and a simple grammar for a concrete syntax for the language defined in `Expr.hs`. Implement a parser for the language (following the grammar described in the file) as `expr`. The `main` function of `Main.hs` asks the user for an expression, runs the parser and then interprets the expression.

Note: You may need to start GHCi with `-package parsec` to make the Parsec library available. You can find the documentation of Parsec at `https://hackage.haskell.org/package/parsec`.

The modules `Text.Parsec.Prim` and `Text.Parsec.Combinator` provide some useful combinators to help with parsing.

Due date: Tuesday, July 18, 2023 at 16:00

Upload your solutions to the `Submissions` folder for this exercise in Stud.IP.