Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Datenbanksysteme · Prof. Dr. Grust

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Functional Programming

SS 2017

Benjamin Dietrich

Assignment #0

Submission Deadline: Thu, 27.4.2017

Please observe the following rules for this and all future assignments:

1. Please hand in your assignments as teams of two. Teams should stay fixed over the course of the semester. If you have not yet found a team partner, please use the forum to find one.
2. In general, the only acceptable file format is plain text (`*.txt`, `*.hs` for Haskell code). Files in other formats are not graded, unless explicitly stated differently.
3. Your code has to work with GHC 8.0.2. We recommend to install GHC using the *Haskell Platform* (see below).
4. All code you submit must compile. Code that does not compile (in particular: does not typecheck) might not be graded.
5. You should aspire to code that does not produce any warnings when compiled with the GHC option `-Wall`.
6. Please submit code that is nicely and consistently formatted and well-documented.[1] Every top-level function definition has to include a type signature and a comment.
7. The usual rules for plagiarism and academic integrity apply. Work only in your team and don't share code and solutions with fellow students. Reference any external sources you use in your solution.

This first assignment should help you getting started with `git` and *Haskell*. However, it will **not be graded**. Regular, graded assignments will start next week.

[1] To have an idea of "nicely formatted code", you can find a short style guide here: https://github.com/tibbe/haskell-style-guide/blob/master/haskell-style.md.

Exercise 1: How to use `git` (0 Points)

Since you are reading these lines you obviously managed to set up and check out a repository for your team at *GitHub*. The first exercise for each of you (individually) will be, to update a file in this repository.

Check out the latest version of the file `README.md` – which may be the initial one or an updated version of your team partner – and complete it with your name, your "Matrikelnummer" and your study program. This information will be the basis of your grading at the end of the lectures. Remember that your changes are only visible to others (in particular the tutors), if you *pushed* them to *GitHub*.

Now that you basically know how `git` works you can – and should – use it, not only to hand in your solutions every week, but also to work on the exercises together with your team partner. **Exercise 2: Haskell**

Tools (0 Points)

First, we need a Haskell implementation. The Haskell Platform bundles the Glorious Haskell Compiler (GHC) with a set of libraries and tools that are considered a standard Haskell environment.

1. Please install the Haskell Platform 8.0.2 from https://www.haskell.org/platform/ or your system's package manager. To check that things work, compile and run the "Hello World" program shown in the lecture (`chewie.hs` is available on the course website).

2. Open the file `isPrime.hs` within the GHCi REPL and enter the following statements:

```
1 *Main> primes = [ p | p <- [1..15], isPrime p ]
2 *Main> :sprint primes
3 *Main> head primes
4 *Main> :sprint primes
5 *Main> primes == [1..15]
6 *Main> :sprint primes
7 *Main> last primes
8 *Main> :sprint primes
```

   Inspect the values bound to `primes` as printed in lines 2, 4, 6 and 8. Explain your observations.

3. Arithmetic operators `(+)` and `(-)` have lower precedence than `(*)` and `(/)` ("*Punkt vor Strich*"). Implement equivalent operators `(+~)`, `(-~)`, `(*~)`, `(/~)` that behave in the opposite way, *i.e.* "*Strich vor Punkt*", such that *e.g.*

   ```
   8 *~ 2 +~ 6 ≡ 64
   ```

   The new operator's type and associativity should be the same as for their regular equivalents.