Exercise Sheet 1

Functions, Types, and Design Recipes

STS Software Technology Systems

Functional Programming - Winter 2022/2023 - November 01, 2022 - Schupp/Lübke

How to succeed with the labs and exercises?

Labs and exercise sheets are published every week on the course homepage at StudIP. As described in the first lecture, each successfully completed lab and exercise earns you bonus points towards your final score in this semester's exam. Keep in mind that you only get bonus points if you would pass the exam without them. **Cheating does not help you - but we will!**

How to complete a lab successfully?

In the lab, you will solve the tasks on lab sheet. You are encouraged to talk to your neighbors and find solutions together, as well to ask the tutor for help. **Towards the end of the session, the tutor will briefly discuss your solutions with you. To pass the lab, you should complete two thirds of the tasks (rounding half up)**.

How to complete an exercise successfully?

In order to complete an exercise sheet successfully, you must upload your answers using INGInious **before the deadline** printed on the exercise sheet. We will not consider any solutions handed in after the deadline! Furthermore, you must solve and hand in the exercises **individually** and your Haskell code **must compile** and **pass certain amounts of tests** as specified. During the exercise session, we develop possible solutions together. Please participate! We encourage you to ask and answer questions from fellow students.

Technically, Haskell files you submit using INGInious must have the format as specified in the task sheets (usually ".hs", ".lhs", or ".txt"). Furthermore, INGInious will only consider your last submission. Therefore, if you first submit successfully (your code compiles and tests are passed) and afterwards unsuccessfully (your code does not compile or certain tests fail again), your last submission counts, and - if it does not compile - will therefore be ignored. Make sure your last submission was successful!

How to get additional information?

We encourage you to discuss past and present exercise sheets with us. Either approach us during the exercise session, or visit us during the weekly office hours. We are also available via e-mail or on the StudIP forum. We try to reply as quick as possible and in general, you should get a reply the next weekday, but we cannot guarantee this.

DEADLINE: 08:00, November 07, 2022

2022-11-07T08:00:00+01:00

In this exercise, you will play with strings. Open "Ex01.hs", supplied in the exercise's ZIP file, in a text editor and load it with GHCi. You should see something similar to the following:

```
root ➜ /workspaces/haskell $ ghci Ex01.hs
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
[1 of 1] Compiling Ex01             ( Ex01.hs, interpreted )
Ok, one module loaded.
*Ex01>
```

The "Ex01.hs" contains Haskell code that defines three string variables (`foo`, `bar`, and `baz`), and several functions over strings. You do not have to worry about how these definitions work yet. Whenever you enter a line into GHCi, it is evaluated for you and the result is printed to the screen (e.g `42 - 23`). Note that variables evaluate to their contents.

Task 1 Use GHCi to find out what strings the variables `foo`, `bar`, and `baz` contain. Document your findings by filling out the "Description" field in the "Ex01.hs" file.

Read through the "Purpose" and "Examples" fields of the functions in the "Ex01.hs" file. In the remainder of this exercise, you may only use functions and string variables defined in the "Ex01.hs" file.

Task 2 Find a Haskell expression that produces the string `"Hard"`.

Replace `undefined` by your expression in the definition of `question1` in the "Ex01.hs" file. After pressing the reload button or typing `:r` into GHCi, `question1 foo bar baz` should be equivalent to your expression.

Hint: Take a look at the syntax for function application in the lecture notes, which you can find on StudIP.

Task 3 Your next task is to find a Haskell expression that produces the string `"Cold World"`. Once you have it, replace the `undefined` in `question2` with your expression, reload, and test your definition.

Task 4 In standard mathematical notation, you can write $f(g(x))$ to apply a function f to the result of a function g applied to an argument x. In Haskell, you cannot write `f g x` since then, `f` would be applied to `g` and `x`. Similarly to math, one can use parentheses to group sub-expressions. In this case, one would write `f (g x)` to ensure that `f` is applied to the result of `g x`.

Replace the `undefined` in `question3` with a Haskell expression that produces the string `Hello`.

Task 5 Replace the `undefined` in `question4` with a Haskell expression that produces the string `Hello World`.

Upload your edited "Ex01.hs" file as a `zip` file using INGInious (*https://inginious.sts.tuhh.de*[1]). If your name contains any non-ASCII[2] characters (e.g., ä, ö, ü, ß), please replace them with the appropriate sequence of ASCII characters (ae, oe, ue, ss). The zip file should be named as follows:

`Ex01_[FirstName]_[LastName].zip`
(Example: `Ex01_Ole_Luebke.zip`)

DO NOT rename the files or any of the predefined functions or arguments. For INGInious to accept your submission as correct, it must at least pass the tests for:

- **Task 2**,
- **Task 3**,
- **Task 4**,
- **Task 5**

If any tests failed for a specific sub-task, you can get further information about a counterexample for these tests in the test results section of INGInious. Additionally, you also have to comply with the stated requirements for the uniqueness, file types, and naming conventions.

Please remember, that you are not guaranteed to get the bonus point when INGInious judges your submission as correct. We also perform manual checks (e.g., for free text answers, plagiarism, ...) that determine the final result. You will be informed about the outcome via e-mail.

[1] Only reachable from TUHH network or via VPN
[2] https://en.wikipedia.org/wiki/ASCII