Requirements

- GHC 8.6.5
  - base >=4.10
  - array >=0.5.2.0
  - QuickCheck ==2.13.2
- Module-declaration must be present.
- All type-declations *must* be specified according to the assignment specification, otherwise the submission is negative due to technical reasons.

Submission

It is sufficient to submit the single file containing the solved assignment, e.g. `Assignment1.hs`. Note that this file must contain a module-declaration. A module declaration can be defined as follow:

```
module Assignment1 where
```

Reference: https://www.haskell.org/onlinereport/haskell2010/haskellch5.html

The submission can be tested on the server via GHC using the command:

```
ghci Assignment1.hs
```

We want to emphasise that in the new submission system it is still necessary to use the exact type-signature from the assignment in your submission, otherwise, if the signature does not match the assignment's the submission will be negative due to technical reasons. *Therefore write the type-signature above each function, required in the assignment.*

Project Template Overview

The project template has the following structure:

```
./
│   .gitignore
│   cabal.project
│   Setup.hs
│   stack.yaml
│   template-ffp.cabal
│
├───src
│       Assignment1.hs
│       Assignment2.hs
```

```
│       Assignment3.hs
│       Assignment4.hs
│       Assignment5.hs
│       Assignment6.hs
│       Assignment7.hs
│
└───test
        TestMain.hs
```

To solve your assignments, it is sufficient to edit and submit only the files in the `src` directory.

Tests

You can define automated tests in `test/Main.hs`. The folder is not collected for grading. Therefore, there are no restrictions on how to write or structure your tests. We recommend the hspec library and this tutorial for it: https://hspec.github.io/getting-started.html

Project Files

The following files may not be edited nor submitted for grading.

- `cabal.project` is used by the tool `cabal-install`. It contains an `index-state` to ensure a reproducible build.
- `template-fpp.cabal` describes the dependencies of a single project. It contains compiler options and other meta-data. It is comparable to a `pom.xml` for `maven`.
- `stack.yaml` is similar to `cabal.project` and serves the same purpose, but is specific to the `Stack` tool.
- `.gitignore` is a file for git-project that defines which files should not be tracked for changes.
- `Setup.hs` defines how a project is actually built. This file must not be altered and you do not have to submit it.

Supported Tools

The project template can be built and executed using multiple project management tools. If you decide to use a project management tool, it is sufficient to use only one of them,

- Stack
  - Version: 2.1.3.1
- cabal-install
  - Version: 3.0.0.0

If you do not want to use either of these tools, you can use a plain `GHC` installation with `ghci`.

Basic Introduction

cabal-install

The project management tool cabal-install only takes care of building the project. You have to perform the `GHC` installation yourself. Some ways to install `GHC`:

Linux

- Package Manager (e.g. Ubuntu eoan (19.10))
  - `apt install ghc`
- ghcup

MacOs

- Brew
  - `brew install ghc`
- ghcup

Windows

- Chocolatey
  - Recommendation: https://hub.zhox.com/posts/introducing-haskell-dev/
- ghcup

Basic Commands

Build the project:

```
> cabal build
```

To open the project in an interactive shell, similar to `hugs`:

```
> cabal repl src/Assignment1.hs
```

If you write any tests, you can execute them via:

```
> cabal run tests
```

Stack

The project management tool Stack takes care of the `GHC` installation. Therefore, it is sufficient to only install `Stack`, which takes care of the `GHC` installation for you. Afterwards, the following commands can be used:

Basic Commands

Build the project:

```
> stack build
```

To open the project in an interactive shell, similar to hugs:

```
> stack repl src/Assignment1.hs
```

If you write any tests, you can execute them via:

```
> stack test
```