Curiosities

A mock assignment

Quick overview:

Context: This is a Haskell testing framework, which is my licence thesis project.
Objectives: This framework attempts to improve the testing of assignments for the Functional Programming lab by being easy to use, providing feedback for mistakes, and not requiring you to download external libraries.
What I ask from you:

1. To try and solve some tasks in a mock assignment (you don't have to fully solve the tasks)
2. To test your solutions using the framework and see if it helps you solve the tasks (feedback is separate and can be found in the detailed output)
3. To provide some feedback about the framework in this form: `https://forms.gle/oP6SRXQkpsoAPkbL8`

What you need: To have at least version 9.2 of Haskell installed, but you should already have it from the Functional Programming lab if you haven't uninstalled it.

---

1 Submission instructions

1. Unzip the Curiosities.zip folder. You should find 1 folder and 1 file:
   - src folder - your workspace
   - .gitignore - if you want to use version control
2. Edit the `src/Misc/ListOperations.hs`, `src/Math/BasicOperations.hs`, `src/Math/BaseTranslations.hs` and `src/Misc/FileOperations.hs` files with your solutions.
3. When done, submit the `src/Misc/ListOperations.hs`, `src/Math/BasicOperations.hs`, `src/Math/BaseTranslations.hs` and `src/Misc/FileOperations.hs` files and your feedback on the following form: `https://forms.gle/oP6SRXQkpsoAPkbL8`

2 Project resources

Resource	Link
The System.IO module	`https://hackage.haskell.org/package/base-4.14.0.0/docs/System-IO.html`
The Data.List module	`https://hackage.haskell.org/package/base-4.18.0.0/docs/Data-List.html`

3 Getting started with the development

Starting code

You will have to work in the following files:

- `src/Misc/ListOperations.hs` - Here you will implement the list processing task
- `src/Math/BasicOperations.hs` - Here you will implement the mathematical operations tasks
- `src/Math/BaseTranslations.hs` - Here you will implement the base processing and translation tasks
- `src/Misc/FileOperations.hs` - Here you will implement the file reading function

At the end you may run `runhaskell.exe .Main.hs base4to10.txt` to see your solutions working in tandem.

Development process

First, you should run `runhaskell.exe .TestTests.hs` to confirm that the tests fail.

Then you should choose a test group, because groups contain related tests for a given aspect of the application and try to implement a solution such that (some of) the tests pass. Once you are satisfied, you can move on to the next test group, repeating this procedure.

If your Haskell extension for VSCode works, you might also find evaluating the examples placed above function helpful.

4 Tasks

4.1 List Operations

Task 4.1 0.5p

Implement the `findCommonElements` function, which will receive two lists and should return a **sorted** list of the elements they have in common.

4.2 Mathematical Tasks

4.2.1 Mathematical Operations

Task 4.2.1 0.25p

Implement the `#^^` function, which will receive two Integers `a` and `n`, and will return "a tetrated to n".
The acceptable range for `a` is `0 to 4` and for `n` is `1 to 3`, if the values provided are outside that range 0 will be returned instead.

Tetration is a mathematical operation which repeats an exponentiation, much like the exponentiation repeats a multiplication.
Tetration can be defined recursively as:

$$a^{\#\hat{}\hat{}}n = \begin{cases} 1 & \text{if } n = 0, \\ a^{a^{\#\hat{}\hat{}(n-1)}} & \text{if } n > 0, \end{cases}$$

Task 4.2.2 0.2p

Implement the `populateOperations` function, which will receive two Integers `a` and `n` and will return an `Operations` value populated with the operations applied to the inputs ($a + n$, $a \times n$, a^n, $a^{\#\hat{}\hat{}}n$).

4.2.2 Base Translations

Task 4.2.3 0.25p

Implement the `translateFrom4` function, which will receive a String of a number in `base 4`, and should return another String of the received number translated into `base 10`.
If the input String contains characters other than 0, 1, 2, or 3, they should be ignored.

Task 4.2.3 0.35p

Implement the translateFromInto function, which will receive two numbers x and y, and a String of a number in base x, and should return another String of the received number translated into base y.
If the input String contains digits not supported in base x, or other characters, they should be ignored.

4.3 File Operations

Task 4.2.3 0.3p

Implement the readBaseTranslationFile function, which will receive a file name of type Maybe String at the input, referencing a file with contents in the following format "x y numeric_string", and should return a tuple of the contents in the form of (x, y, numeric_string) or an error.

- If at the input the file name provided cannot be found, Error FileNotFound will be returned instead.
- If the input file's contents are not compliant with the provided format, Error ParseFailed will be returned instead.
- If the input file is successfully interpreted, the resulting tuple will be wrapped in Sucess.

5 Testing your implementation

To run all tests, use:

powershell session

```
PS> runhaskell.exe .\Test\Tests.hs
```

To see detailed output for failed tests (i.e. why did a test fail), you can use the -d or –detailed flag:

powershell session

```
PS> runhaskell.exe .\Test\Tests.hs -d
```

To run tests selectively by test name, use the -n or –name flag, followed by the name of the test:

powershell session

```
PS> runhaskell.exe .\Test\Tests.hs -n tetration
```

To run tests only for the list task, mathematical tasks or file reading you can use:

powershell session

```
PS> runhaskell.exe .\Test\Tests.hs list
```

powershell session

```
PS> runhaskell.exe .\Test\Tests.hs math
```

powershell session

```
PS> runhaskell.exe .\Test\Tests.hs io
```