Programming Paradigms

Lab 7. Input and output in Haskell

Outline

- Higher-order functions and lists in Haskell recap
- Adding user’s numbers
- Exercise: N-body simulation

Adding user’s numbers

Exercise 6.1.
Implement a program **`sumOfTwoInputs`** that reads two integers from standard input and prints their sum.

Exercise 6.2.
Implement a program **`sumOfManyInputs`** that reads a number N followed by N integers from standard input and prints their sum.

Running a one-question test

Exercise 6.3.
Design types **`Question`** and **`Answer`**, and implement a function

```
runQuestion :: Question -> IO Answer
```

that asks the question to the standard output and collects a **numeric** answer from the standard input. What should the program do when no answer is given? Modify the type of **`runQuestion`** accordinly.

Running a full test

Exercise 6.4.
Implement a function **`runQuestions`** that runs a series of questions and collects answers for all of them.

Exercise 6.5.
Design types **`Test`** and **`TestAnswers`**, and implement a function

```
runTest :: Test -> IO TestAnswers
```

that asks runs a test (a series of questions) and collects a user answers from the standard input.

Introducing more types of questions

Exercise 6.6.

Extend types `Question` and `Answer` to support True-or-False questions.

Exercise 6.7.

Extend types `Question` and `Answer` to support multiple choice questions.

Exercise 6.8.

Extend types `Question` and `Answer` to support free text responses.

Grading user’s answers

Exercise 6.9.
Design the types **`Checker`** and **`TestChecker`** to represent checkers for individual questions and entire test correspondingly.

Exercise 6.10.
Implement function **`gradeTest`** to grade **`TestAnswers`** using **`TestChecke`**r.