Assignment 5

Monads

Deadline: Monday, December 21, 23:55

5.1 Submission instructions

1. Unzip the `A5.zip` folder. It should contain one file:
   - `Solutions.hs` - for the Haskell exercises
2. Edit the first line of each of the source files as described in the comments.
3. Edit the source files with your solutions.
4. When done, zip (not rar renamed as zip!) this `A5` folder and name the zip archive with the following format:

$$A5\_\langle FirstName\rangle\_\langle LastName\rangle\_\langle Group\rangle$$

   Examples of valid names:
   - `A5_John_Doe_30432.zip`
   - `A5_Ion_Popescu_30434.zip`
   - `A5_Gigel-Dorel_Petrescu_30431.zip`

   Examples of invalid names:
   - `Solutions.zip`
   - `A5.zip`
   - `Solutii_A5_Ion_Popescu.zip`

5.2 Assignment exercises

5.2.1 Haskell

Exercise 5.2.1 3p

Implement a function `passwords` that enumerates all 8 character passwords containing digits (0 - 9) lowercase letters (a - z) and uppercase letters (A - Z) using the list applicative.

Haskell REPL

```
> take 5 passwords
["00000000","00000001","00000002","00000003","00000004"]
> take 5 (drop 10000 passwords)
["000002Bi","000002Bj","000002Bk","000002Bl","000002Bm"]
```

Hints:
Re-read section 10.2 of Lab 10 about the list applicative.
You should consider using (some of) the following functions: `replicate`, `sequenceA`.

Exercise 5.2.2 3p

Given the following data definition `data Password = Password String`, implement a function `validatePassword :: String -> Maybe Password` that takes a string and returns a valid password wrapped in `Just` or `Nothing`, if the string isn't a valid password.
A valid password should have at least 8 characters and should contain:

- At least one lowercase character
- At least one uppercase character
- At least one digit

Haskell REPL

```
> validatePassword "123"
Nothing
> validatePassword "abc"
Nothing
> validatePassword "MyStrOngPassword2"
Just (Password "MyStrOngPassword2")
> validatePassword "abcDE12"
Nothing
> validatePassword "Abcd1234"
Just (Password "Abcd1234")
```

Hints:
You should consider using (some of) the following functions: `any`, `all`, `elem`, `notElem`

Exercise 5.2.3 5p

Given the following data definition `data Email = Email {username :: String, domain:: String}`, write a function `validateEmail :: String -> Maybe Email` that takes a string a returns a valid email address wrapped in `Just` or `Nothing`, if the string isn't a valid email address.
A valid email address (`jonny@example.com`) should:

- Contain a username (the part before the "`@`" character) that is at least 3 characters long (`johnny`)
- Contain exactly one "`@`" character
- Contain a domain name (the part after the "`@`" character) that contains:
  - A non-empty hostname (which can contain "." characters) (`example`)
  - A "." character
  - A top level domain (`com`)

Haskell REPL

```
> validateEmail "johnny@example.com"
Just (Email {username = "johnny", domain = "example.com"})
> validateEmail "johnny@.com"
Nothing
> validateEmail "johnny.com"
Nothing
> validateEmail "johnny@domain@.com"
Nothing
> validateEmail "x@domain.com"
Nothing
validateEmail "johnny.john@domain.com"
Just (Email {username = "johnny.john", domain = "domain.com"})
> validateEmail "johnny.john@domain.example.com"
Just (Email {username = "johnny.john", domain = "domain.example.com"})
```

Hints:
You should consider using (some of) the following functions: `span`, `break`, `null`

Exercise 5.2.4 — 4p

Complete the `main` function in `Solutions.hs` to create a Haskell program for checking whether an email and password are valid. The program should prompt the user to enter their email address and password, check if they are a valid email address and password and show a message whether the input was valid or not.

Grading:

- 2 points for implementing the `validateUser :: String -> String -> Maybe User` function that uses `validateEmail` and `validatePassword` to validate an email address and a password
- 2 points for implementing the main function