Programming Paradigms

Lab 9. Revising Haskell. Wholemeal programming

Outline

- Side-scrolling game
- Rotating background
- Working with an infinite universe
- Generating random infinite universe

Inferring types

Exercise 6.1.
What is the type of the following definitions? Provide the most general type. Verify your answer against the compiler.

```
exercise_6_1_a = 2 + 3

exercise_6_1_b (x, y) = sqrt (x^2 + y^2)

exercise_6_1_c x y = [x, y]

exercise_6_1_d [] = ""
exercise_6_1_d (x:xs) = exercise4 xs ++ [x]

exercise_6_1_e x = (x, x)
```

Wrapper types

Exercise 6.2.
Consider the following type declarations representing cartesian and polar coordinates of a 2D point. Implement conversion functions.

```
data Cartesian = Cartesian Double Double
data Radians = Double
data Polar = Polar Double Radians


toPolar :: Cartesian -> Polar

fromPolar :: Polar -> Cartesian
```

Higher-order functions

Exercise 6.3.

Implement higher-order function concentric, that renders concentric shapes. Using concentric, render a picture of practice target:

```
main :: IO ()
main = drawingOf
  (concentric targetCircle 10)
```

Algebraic data types + stateful computation

Exercise 6.4.

Consider the following type of commands. Implement a function that follows a sequence of commands to render a picture.

```
type Radians = Double

data Command
  = Forward Double
  | Rotate Radians
  | TeleportTo (Double, Double)

runCommands :: [Command] -> Picture

main = drawingOf (runCommands
  [ Rotate (2*pi/3), Forward 2, Forward (-4)
  , TeleportTo (0, 0), Rotate (2*pi/3), Forward 2 ])
```

Input and output in Haskell

Exercise 6.5.

Implement an interactive program that records every user input and allows to search history via command **`/search <phrase>`**. Use `Data.List.isInfixOf` to search for previous inputs matching the search phrase. Example (user input is bold):

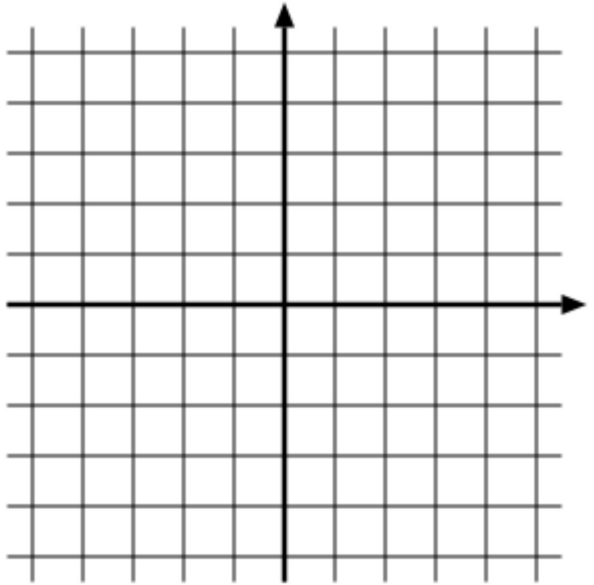
```
input> something to record
input> something else to record
input> Haskell is strongly statically typed
input> lazy evaluation is useful in purely-functional languages
input> /search something
Found 2 records:
- something to record
- something else to record
```

Generating lists

Exercise 6.6(a)
Implement function `renderCartesianGrid`:

```
renderCartesianGrid
  :: (Double, Double)
  -> (Double, Double)
  -> Picture

main :: IO ()
main = drawingOf
  (renderCartesianGrid
     (-5.5, 5.5)
     (-5.5, 5.5))
```



Generating lists

Exercise 6.6(b)

Implement function `renderPolarGrid`:

```
renderPolarGrid
  :: Double
  -> Int
  -> Picture


main :: IO ()
main = drawingOf
  (renderPolarGrid 5.5 5)
```