P. Thiemann, G. Radanne Wintersemester 2017/18

Functional Programming

http://proglang.informatik.uni-freiburg.de/teaching/functional-programming/2017/

Exercise Sheet 6 – First steps with monads - state

2017-12-20

In this exercise, we will be using the state monad. Documentation available at

https://hackage.haskell.org/package/transformers-0.5.5.0/docs/Control-Monad-Trans-State-Lazy.html

Along with the usual monad operations, the state monad allows to access the state contained in the monad. For example, the following function increments a counter and returns the old value.

```
incr :: State Int Int
incr = do n <- get
          put (n+1)
          return n
```

You then use the `runState` function to perform the operations and obtain the final state.

Exercise 1 (Pseudo random number generator)
A common way to generate (pseudo-)random numbers is through the use of a series. For example, Donald Knuth in the Art of Programming presents the following series:

$$x_n = (6364136223846793005 * x_{n-1} + 1442695040888963407) \bmod 2^{64}$$

This style of pseudo-random number generator is known as a linear congruential generator[1]. Implement the following API using the traditional State monad operations:

```
type Random a = State Int a
fresh :: Random Int
runPRNG :: Random a -> Int -> a
```

Exercise 2 (While – Evaluation)
Implement an interpreter for the miniwhile language presented in Ex05 using a State monad containing the memory. One could start with the following definitions:

```
type Value = ....
type Id = String
type Memory = Map Name Value

eval :: Program -> Memory
```

Here are some hints to make things simpler:

- First assume that variables always store `integer` values. At the beginning all variables have the value 0. (Since the While language also supports Boolean values, you must represent them as numbers, à la C)
- The module `Control.Monad` contains many useful functions for programming with Monads.

We will see how to add errors and printing in later exercises.

[1] https://en.wikipedia.org/wiki/Linear_congruential_generator