Programming Paradigms 2022
Session 8: Interactive programming

Problems for solving and discussing

Hans Hüttel

1 November 2022

Problems that we will definitely talk about

1. ***(Everyone at the table together – 15 minutes)*** Here is a program.

```
main = do
  w <- getLine
  loop ( (read w) :: Int)
  where
      loop 1 = putStr (show 1)
      loop x = do
        putStr (show x)
        if even x
          then loop (x `div` 2)
          else loop (3*x + 1)
```

   Do not run it! Try to find out what it does.

2. ***(Work in pairs – 20 minutes)***

   Use recursion to define a Haskell value letter that is a sequence of actions which does the following:

   - Receive a string
   - Print out the characters of the string one by one, with each character followed by a linebreak

   As an example, we would expect the following:

```
*Main> letters
dingo
d
i
n
g
o
*Main>
```

3. ***(Everyone at the table together – 15 minutes)***

   Give another definition of letters that uses the sequence_ function from discussion problem 2.

4. ***(Work in pairs – 20 minutes)***

   Define an action hugorm :: IO() that reads a given number of integers from the keyboard, one per line, and then finally displays the sum of the integers[1]. As an example, we would expect the following:

[1] *Hugorm* is the Danish word for *adder*.

```
*Main> hugorm
How many numbers would you like to add? 5
1
2
3
4
5

 The sum is 15*Main>
```

You will need the functions read :: Read a =>String -> a and show :: Show a =>a -> String to get numbers from strings and to display numbers as strings, respectively. All types in the type class Num are also types in the type classes Read and Show.

More problems to solve at your own pace

a) Write a recursive function sumInts :: Integer -> IO Integer that repeatedly reads integer numbers from input until the number 0 is given. When that happens, the function will return the sum of all the numbers that were entered plus the original (default) value, which is given as the first parameter of sumInts.

b) We can generalize sumInts as a higher-order function whileIO which, for the given reading IO action getIO, termination condition condF, folding function foldF, and the original value, returns the required IO action.

Check that for some values of getIO, condF and foldF, we can redefine sumInts using whileIO.