Programming Paradigms 2025
Session 9 : Interactive programming

Hans Hüttel

4 November 2025

Our plan for today

1. The learning goals
2. Presentations of the preparation problems
3. Problem no. 1
4. Problem no. 2
5. Break
6. Problem no. 3
7. Problem no. 4
8. If time allows: More problems at your own

Learning goals

- To understand the underlying idea of I/O in Haskell
- To be able to use the IO type construct in Haskell
- To be able to use sequencing with do blocks to write interactive programs
- To be able to write Haskell programs that combine the pure and impure features of Haskell

Preparation problem – “Hello”

Write a Haskell program that asks for the name of the user and greets the user with a ”Hello”. We would like to see the following behaviour:

```
*Main> hello
What is your name?
Graham
Hello Graham
*Main>
```

Preparation problem – What is going on?

Find out what the following expression does:

```
sequence_ [putStr "rip", putStr "rap",
    return ()]
```

and why Haskell will complain about

```
sequence_ [putStr "rip", putStr "rap",
    getChar]
```

Then give an explanation. (Page 135 is your friend.)

Problem/discussion – What does the code do?

```
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.

The Collatz conjecture

The Collatz conjecture is one of the most famous unsolved problems in mathematics and has led to a lot of important theoretical insights. It is named after the German mathematician Lothar Collatz, who introduced the idea in 1937.

We can build a sequence of integers as follows:

- Start with a seed x_0
- For every $i \geq 0$, we build the next term x_{i+1} as follows:
  - If x_i is even, then let x_{i+1} be $x_i/2$
  - If x_i is odd, then let x_{i+1} be $3x_i + 1$

The Collatz conjecture is now that every such sequence will eventually reach 1, no matter which seed we choose.

Problem 2 – Writing letters

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>
```

Break

Problem 3 – Writing letters again

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

Problem 4 – Hugorm

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:

```
*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. Types in Num are also members of Read and Show.

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

Evaluation

- What did you find difficult?
- What surprised you?
- What went well?