Programming Paradigms 2024
Session 4 : Functions and lists

Hans Hüttel

8 October 2024

Our plan for today

1. The learning goals
2. Presentations of the preparation problems
3. Problem no. 1
4. Break
5. Problem no. 2
6. Problem no. 3
7. Problem no. 4
8. Problem no. 5
9. If time allows: More problems at your own pace.
10. We evaluate today's session – please stay until the end!

Learning goals

- To understand the syntax and informal semantics of conditional expressions and guards and to be able to use these when programming in Haskell
- To understand and be able to use the different forms of patterns and pattern matching when programming in Haskell
- To understand anonymous functions (lambda expressions) in Haskell
- To understand the notion of list comprehension and how guards are used in comprehensions
- To be able to use list comprehension for defining functions in Haskell
- To understand how all these notions can be used in when building larger programs and to be able to apply this understanding when programming in Haskell

Discussion problem – onlytwo

Define, using pattern matching and *without using the length function*, a function onlytwo that tells us if a list has precisely two elements – in which case it must return True – or not, in which case it must return False. What is the type of onlytwo?

Discussion problem – alldots

The dot product of two pairs of numbers (a, b) and (c, d) is the number $a \cdot c + b \cdot d$. Define, *using list comprehension*, a function alldots that takes two lists of pairs of numbers and returns all the possible dot products of every pair from the first list and every pair from the second list. Find two good test case for testing your function definition and use them to test your code. What is the type of alldots?

Problem 1 – idhead (10 minutes)

Define a function idhead that will tell us if a list of pairs begins with a pair whose first and second components are identical.

We would like

```
idhead [(42,42),(3,4),(484000,5)]
```

to give us True but would like

```
idhead [("plip","mango"),("dingo","kpst")]
```

to give us False.

In your definition you are *not* allowed to use head or if-then-else-expressions! Is the function polymorphic? If so, in what way?

Discussion – Married at first sight

During breaks in the recording of *Married at first sight* one of the couples got into a heated argument about data structures in Haskell. Here is what they said.

- ”List” is just a clumsy word for array! Both are long data objects and we use square brackets to define both. That Hutton idiot does not know what he is talking about. He keeps getting the terminology wrong!
- Come on! Surely there must be a reason why one calls lists lists and not arrays. I want a divorce!!
- Oh yeah ?! Give me just *one good reason* why lists are not arrays! I will be the first of us to get a divorce!!

A fist fight broke out and the camera crew had to intervene and provide a correct explanation. What did the camera crew say?

Break

Problem 2 – Pyt (15 minutes)

A *Pythagorean triple* is a triple (a, b, c) of natural numbers a, b, and c, such that $a \leq b < c$ and $a^2 + b^2 = c^2$. In other words, a triple of this form gives us the length of the three sides of a right triangle for which all sides have integer length. The smallest Pythagorean triple is $(3, 4, 5)$.

Use list comprehension to define a function pyt that, when given an integer k, gives us a list of all Pythagorean triples whose largest element is at most k. Before you write the definition of pyt, find out what its type should be.

Problem 3 – Bighead (15 minutes)

Last week, we read that a famous influencer on Instagram has defined a Haskell function bighead that can tell us how many elements in a list xs are greater than (>) the head of xs. As an example of the behaviour of the function instance, the result of bighead [7,4,5,8,9] will be 2.

Now it is your turn to be a famous influencer. How would you define the bighead function? What should its type be?

Discussion – The silly colon

A famous YouTuber has posted a video about operations on list. The video got 3 million likes within the first 24 hours.

In the video, the YouTuber says

> *I wonder why anyone would even bother to use that strange colon symbol (it does not even have a name!!) It can only be used in patterns to write $x:xs$ in a function definition.*
> *If we want to prepend an element x to a list xs, we have to write $[x]$ ++ xs.*
> *Haskell is such a clumsy language! The designers of Haskell are incompetent and embarrassing, and if they have a job (which I doubt), they ought to be fired immediately. Everything is their fault.*

Is the YouTuber right?

Problem 4 – A serving of curry (10 minutes)

Show how the meaning of the following curried function definition can be given in terms of lambda expressions from Haskell.

```
plonk x y z = x+y+z
```

Figure out the type of plonk without asking the Haskell interpreter.

Problem 5 – Being perfect (10 minutes)

A perfect number n is a natural number that is the sum of its own divisors excluding n itself. 28 is a perfect number, since $1 + 2 + 4 + 7 + 14 = 28$.

Use list comprehension to define a function isperfect that will tell us if any given natural number is a perfect number.

Evaluation

- What did you find difficult?
- What surprised you?
- What went well?
- Is there a particular problem that we should follow up on with a short video?