Programming Paradigms 2025
Session 6 : Higher-order functions

Preparing for the session

Hans Hüttel

14 October 2025

Where nothing else is mentioned, chapters and page numbers refer to *Programming in Haskell*.

The video podcast

You can watch the podcast on YouTube via the course page on Moodle.

Tuesday 14 October 2025 – Higher-order functions

The text is Chapter 7 of *Programming in Haskell*.

Learning goals for the session

- To be able to explain precisely what a higher-order function is and what the type of a higher-order function is
- To be able to explain precisely higher-order functions on lists including map, filter , foldr and foldl
- To be able to explain and understand the recursive definitions of common higher-order functions on lists
- To be able to use higher-order functions, including map, filter , foldr and foldl , for solving programming problems in Haskell
- To be able to explain and use function composition in Haskell

How you should prepare before we meet on Tuesday

Before we meet, watch the podcast and read the text. You can do this in any order you like. Also see if you can solve the following two small discussion problems. We will talk about them in class.

1. Every letter in the lowercase English alphabet has a position. ”a” has position 1, ”c” has position 3 and ”h” has position 8.

   In Haskell, every string is a list of characters. So String is the same type as [Char].

   We can define a function positions that, given a string of lowercase letters str gives us the list of positions of the characters in str.

   As an example, positions ”abba” gives us [1,2,2,1] . Use the higher-order functions in Chapter 7 to define positions.

Here it useful to remember that the ordinal value of a character can be computed using the function fromEnum found in the prelude. We have that fromEnum 'a' is 97 and that fromEnum 'b' is 98.

2. The function sumsq takes an integer n as its argument and returns the sum of the squares of the first n integers. So sumsq n returns the sum

$$1 + \ldots + n^2$$

As an example, sumsq 4 gives us 30 and sumsq 9 gives us 285 . Use foldr to define sumsq – and do not use map.