Programming Paradigms 2023
Session 6: Higher-order functions

Problems for solving and discussing

Hans Hüttel

10 October 2023

How you should approach this

- Some problems are meant to be solved in groups. Please work together at the table for those. Screen sharing at your table is best.
- Other problems are meant to be solved in pairs. Please work together two by two, not in groups. Screen sharing at your table is not a good idea for these problems.
- We set a limit on how much time you are going to spend on each of the numbered problems. *If you do not manage to solve a problem* within the time set aside, do not worry about that. We will discuss solutions afterwards. *If you finish early*, work on one of the "lettered problems" in the problem set instead of working on the next numbered problem.

Problems that we will definitely talk about

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

   The within function takes a list of numbers and a pair of numbers, returns a list of numbers which are in the input list and within the range (inclusive) given by the input pair.

   The elements in the output list appear be in the same order they appeared in the input list. If the input pair is (n1,n2), assume that n1 is the lower bound of the range and n2 is the upper bound of the range.

   As an example, within [1,3,4,5,2] (1,3) should give us [1,3,2] and within [1,3,4,5,2] (3,1) should give us [] .

   Define within using the higher-order functions in Chapter 7.

2. ***(Work in pairs – 10 minutes)***

   Implement the sumrows function. The function takes a list of number lists returns a one-dimensional list of numbers with each number equal to the sum of the corresponding row in the input list. If a list is empty, its sum is 0.

   As an example, sumrows [[1,2], [3,4]] should give us [3, 7], and sumrows [[],[],[1]] should give us [0,0,1] .

   Define sumrows using the higher-order functions in Chapter 7.

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

   The base of the natural exponential function $e = 2.718\ldots$ can be written as the limit of the infinite series

   $$\sum_{k=0}^{\infty} \frac{1}{k!}$$

   The function approx should give us the approximation of e that we find by adding the first n terms of this infinite series. That is,

   $$\text{approx } n = \sum_{k=0}^{n} \frac{1}{k!}$$

   Define approx using the higher-order functions in Chapter 7; the factorial function $k!$ is defined by

```
fact k = product [1..k]
```

4. ***(Work in pairs – 10 minutes)***

   Here is a function. What does it do? And why? Use the explanation at the very end of Section 7.3 in the book to help you answer the "why"-question.

   ```
   fingo :: [a] -> [a] -> [a]

   fingo xs ys = foldr (:) xs ys
   ```

5. ***(Everyone at the table together – 10 minutes)***

   The function map can be applied to any function, so we can write map map. What is the type of map map? Figure this out *without* asking the Haskell interpreter – try to justify your answer and only then ask the interpreter.

More problems to solve at your own pace

In your solutions, remember the learning goals of this session! You must use the higher-order functions from Chapter 7 – **not** recursion or list comprehension!!

a) The partition function generalizes the isolate function from Session 5. The partition function takes a predicate p and a list xs and returns the pair of lists of elements which do and do not satisfy the predicate, respectively.

   a) Define partition using filter .
   b) Define partition using foldr .

b) How can we implement the filter function using foldr ?

c) Use foldr to define a function remove which takes two strings as its arguments and removes every letter from the second list that occurs in the first list. For example, remove "first" "second" should give us "econd". First find out what the type of the function should be.

d) The function min2 takes a list of numbers and returns the second-smallest number of the input list. If a list contains duplicates, the second-smallest number and the smallest number can be identical; then the function should return that number. Assume that every input lists contains at least two numbers.

   As an example, min2 [2110, 4820, 3110, 4120] should give us 3110 and min2 [2110, 4820, 2110, 4120] should give us 2110.

   Define the min2 function using the higher-order functions in Chapter 7.

Bibliography

[1] Wikipedia. Continued fractions. `https://en.wikipedia.org/wiki/Continued_fraction`.