

# Programming Paradigms 2022

## Session 3: Functions and lists

### Problems for solving and discussing

Hans Hüttel

27 September 2022

#### 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.
- This time we will try to put a time bound on the problems to see if this will improve the flow of the session. *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.

#### Problems that we will definitely talk about

##### 1. (*Everyone at the table together – 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.

##### 2. (*Solve in pairs – 10 minutes*)

Use list comprehension to define a function `sevens` that given an integer  $k$  gives us a list of all natural numbers that are divisible by 7 and are less than  $k$ . First find out what its type should be.

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

During breaks in the recording of the reality TV show *Paradise Hotel* a contestant decided to write a function `headsup` that can tell us if the two first elements of a list are identical. Here is what the contestant wrote.

```
headsup x = if head x == head (tail x) then True else False
```

The contestant added that the type of `headsup` is

```
[Num] -> Bool
```

The other contestants argued that this solution was clumsy and that there seemed to be something very wrong with the type. What is your opinion and why?

##### 4. (*Solve in pairs – 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.

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

We also saw this problem in Session 2 but it is worth re-visiting! Find a Haskell expression whose type is

```
(Ord a1, Eq a2) => a2 -> a2 -> (a1, a1) -> a1
```

*Hint:* Remember what operators are required for types in the type classes that are mentioned here. And use if-then-else!

## More problems to solve at your own pace

- a) Use list comprehension to define a function `flop` that, when given a list of pairs returns a list of pairs whose components are reversed. The list can be empty.

For example, `flop [(1,'a'),(3,'r'),(9,'e')]` should return the list `[('a',1),('r',3),('e',9)]`.

What is the type of `flop`?

- b) Write a function `dupli` that will duplicate the elements of any given list. As an example, `dupli [1, 2, 3]` must give us `[1,1,2,2,3,3]`. What should the type of `dupli` be? *Hint:* The `concat` function from Chapter 5 will be useful for stitching everything together.

- c) A perfect number  $n$  is a natural number that is the sum of its own divisors that are not  $n$ . 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.

- First figure out what the type of `isperfect` should be.
- Now write the code. *Hint:* Section 5.2 of the book is useful.

- d) 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?

- e) Here is a function `sums` whose definition has one single use of list comprehension.

```
sums m n = [ x+y | x <- [1..m], y <- [1..n] ]
```

The list comprehension in this definition uses two generators. Write an alternative definition of `sums` that only uses list comprehensions (so you may need more than one instance of list comprehension) with one generator each. *Hint:* The `concat` function from Chapter 5 will also be useful here.