The End of the Beginning

What Have We Learned?

- Programming
  - For some of you: first time
  - Make the computer do some useful tasks
- Programming *Language*
  - Haskell
  - Different from what most of you had seen before
- Programming *Principles*
  - ...

Programming Principles (I)

- Modelling
  - Create a new **type** that models what you are dealing with
  - Design and define **typed functions** around your types
  - Sometimes your type has an extra **invariant**
  - Invariants should be **documented** (for example as a property)

Programming Principles (II)

- Properties
  - When you define **functions** around your types...
  - Think about and define **properties** of these functions
  - Properties can be **tested** automatically to find mistakes
  - Mistakes can be in your functions (program) or in your properties (understanding)

Programming Principles (III)

- Breaking up problems into simpler parts, recursion
  - When you need to solve a **large, complicated problem**...
  - Continue breaking up until the parts are simple, or until you can use an existing solution
  - The parts can be solved **recursively**
  - Solve the whole problem by **combining** the solutions of all parts

Programming Principles (IV)

- Abstraction and Generalization
  - When you find yourself **repeating** a programming task
  - Take a step back and see if you can **generalize**
  - You can often define an **abstraction** (higher-order function) performing the old task *and* the new one
  - Avoid **copy-and-paste programming**

Programming Principles

- Important!
- Independent of *programming language*

Report from the front

“Läste kursen 2010 när jag började på D och lärde mig mycket, fast jag tyckte att jag kunde programmera innan. Fick 2012 jobb på Ericsson och programmerade då i Python, och använde då dagligen tekniker som jag lärde mig i kursen, framförallt då rekursion, operationer på listor och delar av det funktionella programmeringssättet som var nytt för mig 2010.”

Report from the front

“En vanlig fråga/missuppfattning som jag minns från början av Chalmers är just 'varför Haskell? Ingen använder det på riktigt i industrin', och det kan vara värt att påminna en extra gång om att <u>man lär sig metoder och tankesätt som är användbara oavsett vilket språk man sedan kodar i</u>.”

Why Haskell?

- What is easy in Haskell:
  - Defining types
  - Properties and testing
  - Recursion
  - Abstraction, higher-order functions
  - Pure functions
  - Separation (laziness)

Why Haskell (II)?

- What is harder in Haskell:
  - Ignoring types
    - Static strong typing
    - Expressive type system
      - Most advanced type system in a real-world language
  - Impure functions
    - All functions are pure
      - Unique among real-world languages
    - Instructions are created and composed explicitly
      - Makes it clear where the ”impure stuff” happens

Coming Programming Courses

D-line

- Grundläggande datorteknik
  - Assembler
- Objektorienterad programming
  - Java
- Inbyggda system
  - C

- Data structures
  - Java
  - Haskell

GU

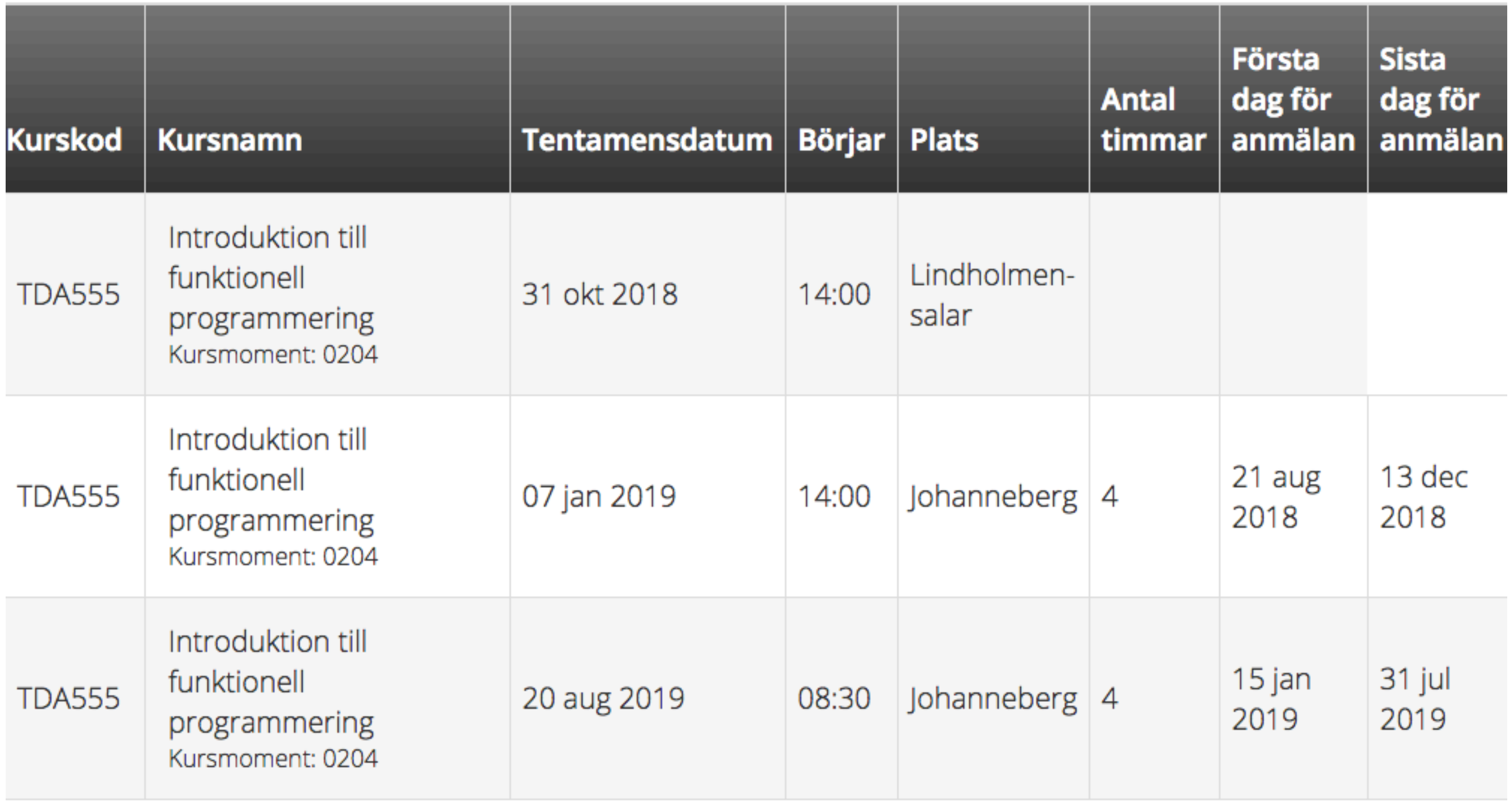
- Two programming courses
  - Both in Java
- Datastructures
  - Java
  - Haskell

Future Programming Courses

- Concurrent Programming
- Compiler Construction
- Advanced Functional Programming
- Parallel Functional Programming
- Software Engineering using Formal Methods
- Language Technology
- (Programming Paradigms)
- ...

All use Functional Programming in some way

Exam: 31 October 14:00



Kurskod	Kursnamn	Tentamensdatum	Börjar	Plats	Antal timmar	Första dag för anmälan	Sista dag för anmälan
TDA555	Introduktion till funktionell programmering Kursmoment: 0204	31 okt 2018	14:00	Lindholmen-salar			
TDA555	Introduktion till funktionell programmering Kursmoment: 0204	07 jan 2019	14:00	Johanneberg	4	21 aug 2018	13 dec 2018
TDA555	Introduktion till funktionell programmering Kursmoment: 0204	20 aug 2019	08:30	Johanneberg	4	15 jan 2019	31 jul 2019

student.portal.chalmers.se/sv/chalmersstudier/tentamen/Sidor/Tentamensdatum.aspx

What if…

You are not done with the labs in time?

- Next year: this course runs again
- complete the missing labs according to the deadlines and rules given

You do not pass the exam?

- January: re-exam
- August: re-exam

EXAM
Introduction to Functional Programming
TDA555/DIT440

DAY: 2016-10-29 TIME: 14:00–18:00 PLACE: M-salar

Responsible: David Sands, D&IT, Tel: 0737 20 76 63

Aids: An English (or English-Swedish, or English-X) dictionary

Grade: Completing Part I gives a 3 or a G;
Part I and Part II are both needed for a 4, 5, or VG

This exam consists of two parts:

Part I (5 small assignments)

- Give good enough answers for 4 assignments here and you will get a 3 or a G
- (Points on Part II can be counted towards Part I if needed, but this is very unlikely to happen in practice.)

Part II (2 larger assignments)

- You do not need to solve this part if you are happy with a 3 or a G!
- Do Part I and one assignment of your choice here and you will get a 4
- Do Part I and both assignments here and you will get a 5 or a VG

Please read the following guidelines carefully:

- Answers can be given in Swedish or English
- Begin each assignment on a new sheet
- Write your number on each sheet
- Write clearly; unreadable = wrong!
- You can make use of the standard Haskell functions and types given in the attached list (you have to implement other functions yourself if you want to use them)
- You do **not** have to import standard modules in your solutions

Good Luck!

What you should know to pass

In general

- Do not expect to pass by learning old exam questions!
- Do **not** assume that old exam questions will come up again this year.
- show that you **understand** by writing more-or-less correct Haskell code for some small problems

Basic Programming Techniques

- Definition by recursion (lists and numbers)
- Definition using list comprehensions

  - Write simple functions
  - Understand definitions
  - Rewrite definitions written in one style using another

example: af function in 2016 exam

Combining functions

- Give definitions which combine the use of other standard functions

example: urls function in 2016

Simple higher-order functions

Understand and use simple higher-order functions

for example: map, filter, takeWhile, dropWhile,
  zipWith, all, any

Define a simple higher-order function e.g. to simplify cut-and-paste code.

Predicates

Writing functions that return something of type Bool

Show that you understand and can formulate properties of functions (e.g. quickCheck properties)

e.g. prop_Lookup from 2016 exam

Simple Data types

- Define simple data types to model a problem domain (both with and without recursion)

- Define functions using given recursive or non-recursive data type

Examples:

  – prop_lookup (exam 2016) uses the Maybe type
  – Defining a data type for expressions (2016)

“Instructions”

Defining simple functions using IO or Gen

- small definitions using do-notation
- understand/simplify definitions that use do notation

To pass (2017 onwards)

Correct answers to 5 of 7 simple questions

How did it go this year?

New things: lab 3 spread over three weeks

  Too slow?

New lab 4 (A + B)

  Too hard/easy?

Course communication via Slack

  Good for us! Good for you?

Official Course Survey

- Sent by email Monday after the exam
- What is the point?
- We are open to suggestions for questions that we should put on the survey.