Course Information

2024/2025 2nd Semester

COMP3258
Functional Programming

Basic Information

- Instructor
    - Dr. Bruno Oliveira (bruno@cs.hku.hk, Chow Yei Chin Building room 420). Consultation hour: Tuesday 4:30pm
- Demonstrators
    - LUO Yicong (yicong.luo@connect.hku.hk). Consultation hour: Thursday 11:30AM

About me

- Associate Professor at HKU
    - Language: English (and Portuguese)
- Research Interests
    - Programming Languages
    - Functional Programming (especially Haskell and Scala)
    - Object-Oriented Programming
    - Modularity

More about me

- I come from Portugal
  - Best known these days for football.
  - Macau was administered by Portugal until 1999.

Functional Programming

- What is functional programming? Some possible answers:
  - Programming with first-class functions
    - map (\x -> x + 1) [1,2,3]   ~> [2,3,4]
  - Programming with mathematical functions
    - No side-effects (no global mutable state, no IO)
    - Calling a function with the same arguments, always returns the same output (not true in most languages!)
  - The main means for computation is function application

Pure Functional Programming

Functional Programming Languages

Traditionally focused on Functional Programming

- Impure Functional Languages
  - Statically Typed: ML,OCaml,Scala …
  - Dynamically Typed: Scheme, Lisp …
- Pure Functional Languages
  - Statically Typed: Haskell

Functional Programming Languages

- Impure Functional Languages
    - Statically Typed: ML,OCaml,Scala, Java 8, C#, C++11, Swift …
    - Dynamically Typed: Scheme, Lisp, Python, Ruby …
- Pure Functional Languages
    - Statically Typed: Haskell, Agda, Idris

Haskell in the course

- Haskell is going to be used in the course as the language to teach Functional Programming

  http://www.haskell.org/haskellwiki/Haskell

Why Haskell?

- Reasons for using Haskell in a Functional Programming course are:
  - Haskell is purely functional! You won't be able to use impure/imperative features
    - There are many more Functional languages, but they are usually not pure.
  - Haskell is a state-of-the-art (functional) programming language

What is this course good for?

- Learning Functional Programming and Functional Programming Techniques
  - Programming without (ab)using side-effects and mutation
  - Recursive Programming
  - Reuse with higher-order functions
  - Programming with parametric polymorphism and strong type systems
- Learn to think differently about programming
  - Functional Programming vs Imperative Programming
  - The functional programming techniques learned in this course apply to any languages
- To make you a better programmer!
- Because **you will probably need FP in your career**!

Functional Programming adoption in Industry

- In the last 15-20 years Mainstream languages, and the industry have been adopting FP:
  - Java 8 adopted lambdas, Java 9 adopted pattern matching
  - Swift, from Apple, is mostly a Functional Programming Language
  - .Net languages support lambdas for many years now
  - C++11 and later versions have lambda expressions

Functional Programming in Java

```java
final BigDecimal totalOfDiscountedPrices =
  prices.stream()
        .filter(price -> price.compareTo(BigDecimal.valueOf(20)) > 0)
        .map(price -> price.multiply(BigDecimal.valueOf(0.9)))
        .reduce(BigDecimal.ZERO, BigDecimal::add);

System.out.println("Total of discounted prices: " + totalOfDiscountedPrices);
```

Functional Programming in Java

A lambda function

```
final BigDecimal totalOfDiscountedPrices =
  prices.stream()
        .filter(price -> price.compareTo(BigDecimal.valueOf(20)) > 0)
        .map(price -> price.multiply(BigDecimal.valueOf(0.9)))
        .reduce(BigDecimal.ZERO, BigDecimal::add);

System.out.println("Total of discounted prices: " + totalOfDiscountedPrices);
```

Functional Programming in Java

```
final BigDecimal totalOfDiscountedPrices =
  prices.stream()
        .filter(price -> price.compareTo(BigDecimal.valueOf(20)) > 0)
        .map(price -> price.multiply(BigDecimal.valueOf(0.9)))
        .reduce(BigDecimal.ZERO, BigDecimal::add);
```

A higher-order function

Functional Programming in Java

```
final BigDecimal totalOfDiscountedPrices =
  prices.stream()
        .filter(price -> price.compareTo(BigDecimal.valueOf(20)) > 0)
        .map(price -> price.multiply(BigDecimal.valueOf(0.9)))
        .reduce(BigDecimal.ZERO, BigDecimal::add);
```

A function as an argument

Functional Programming in Java

Lazy streams

```
prices.stream()
      .filter(price -> price.compareTo(BigDecimal.valueOf(20)) > 0)
      .map(price -> price.multiply(BigDecimal.valueOf(0.9)))
      .reduce(BigDecimal.ZERO, BigDecimal::add);
```

Requirements

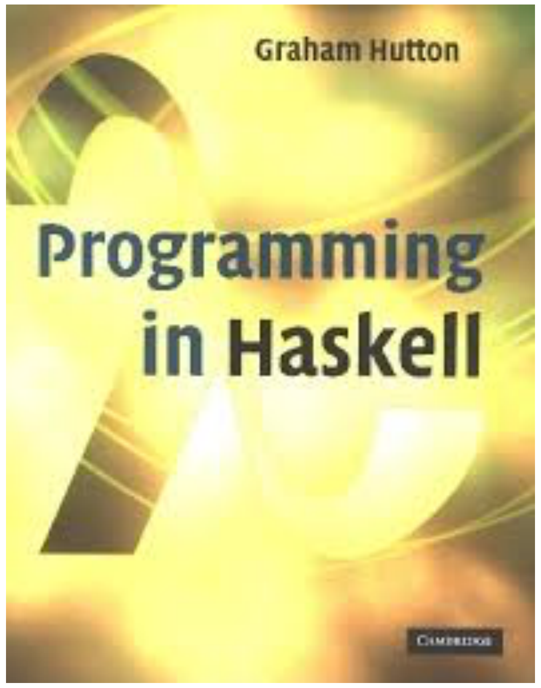
- This is (almost) a beginners programming course
    - Students are not required to have taken a previous programming course
    - However having a previous programming course will definitely help
- Some basic knowledge about discrete maths is recommended, but not necessary

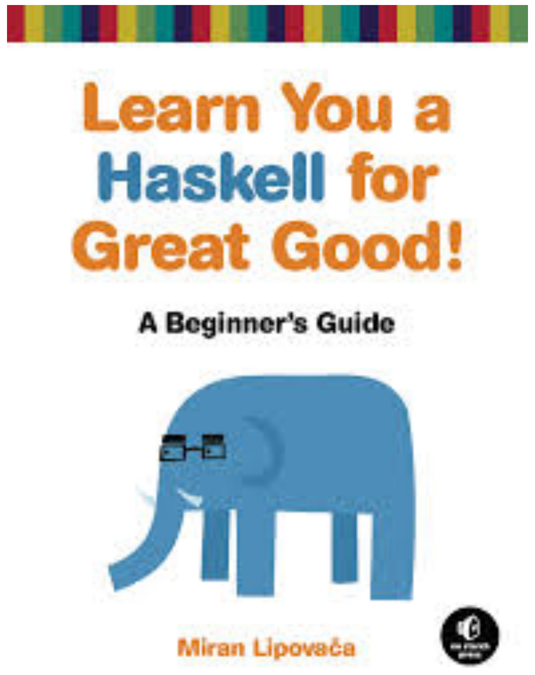
Course Learning Outcomes

- [Implementation] Implement programs correctly using Functional Programming techniques and Haskell.
- [Technologies] To use the GHC compiler and the GHCI command line interpreter.
- [Problem solving] To analyze and design solutions for problems using common functional programming modeling techniques.
- [Programming Techniques] To understand and explain the principles of advanced functional programming techniques including recursion, datatypes, higher-order functions, functional data structures and algorithms.

Basic Information

- Reference Books and Materials
  - Programming in Haskell (Graham Hutton)
    - This is the textbook for the course!
  - Learn you a Haskell for Great Good! (Miran Lipovača)
    - Free and Fun book available online: http://learnyouahaskell.com





Schedule

Week	Topic
1	*Introduction/First steps*
2	*Types and Classes/Defining Functions*
3	*Recursive Functions*
4	*List Comprehensions*
5	*Higher-Order Functions*
6	*Functional Parsers*
7	*Declaring Types and Classes*
8	*Interactive Programs*
9	*The Countdown Problem*
10	*Lazy Evaluation*
11	*Reasoning About Programs*
12	*Revision*

Warning: Content may change according to progress!

Lectures and Tutorials

- Tuesday (Mostly Tutorials)
  - Time: 9:30 ~ 10:20
  - Venue: KKLG102
- Friday (Lectures)
  - Time: 9:30 ~ 11:20
  - Venue: KKLG102
  - First Tutorial: 28th of January

Tutorials

- Tutorial Participation
  - Please attend the tutorials!
  - Attendance will not be recorded, but it is highly recommended.
  - Please answer/raise questions during tutorial.

Late Assignments Policy

- Late assignments
  - upto 1 day late submissions (15% of marks removed)
  - upto 3 days (30% of marks removed)
  - more than 3 days (not accepted)
- Collaboration in study groups is encouraged, but you should write your own program for the assignments.
- Plagiarism will be taken seriously! (Can be reported to the university).

Assessments

- Assignments (40%) (3 assignments)
- Final examination (60%)

Communication Channels

- Please come to us if you have any difficulties in the course
- There are several ways to contact and get in touch with us:
  - email
  - newsgroup
  - consultation hours