Welcome to Concepts of Programming Languages!

School of Computer Science & Engineering, UNSW
Formal Methods, NICTA

2006/S1

— Staffing —

Gerwin Klein (Lecturer)
Simon Winwood (Tutor)
COURSE INTRODUCTION

Check course web page:

http://www.cse.unsw.edu.au/~cs3161/

for information about the course.

Today:

- Admin Issues
- Course Content
Lecture and Tutes

➡ Lecture
  - Monday, 10:00 - 11:00: CivEng G8
  - Tuesday, 10:00 - 12:00: CivEng G8

➡ Tutes
  - Mon and/or Tue afternoon
  - will be announced shortly

➡ Feedback and Questions
  - Messageboard will soon be up
  - Or, send an email to your tutor or lecturer
Course Material

- Text book: Programming Languages: Theory and Practice, Robert Harper (Draft)
- Supplementary reading: The Implementation of Functional Programming Languages, Simon Peyton Jones, Full text online (as jpg page images). The compiler used in this course is based to some extent on this book.
- Supplementary reading: Types and Programming Languages, Benjamin C. Pierce
- Additional material like lecture slides and lecture notes will be available online
ASSESSMENT

➜ Class mark: (50%)
  - Tutorial participation mark (UG 10%, PG 0%)
    Exercises related to lecture of the previous week and forthcoming assignments
    Please read through exercises before the tute!
  - Midsession exam: in Week 5 (UG 10%, PG 12.5%)
  - Assignments: two assignments (30%, PG 37.5%)

➜ Exam mark (50%)
  - supplementary exam: oral exam
Overall mark:

- if at least 40% of the marks in each component are achieved:
  
  \[
  \frac{(\text{class mark} + \text{exam mark})}{2}
  \]

- otherwise

  \[
  \min \left( \frac{(\text{class mark} + \text{exam mark})}{2}, 44 \right)
  \]
ASSIGNMENTS

- Approx. in Week 5 and Week 9, four weeks
- Programming language: Haskell
- Late penalty:
  - 4% of assignment worth per day the assignment is submitted after the deadline
  - submissions accepted until one week after deadline
- Plagiarism: please check School Policy on Yellow Form
**Course Content**

Programming Language Stream:

- Advanced Compiler Construction
- Algorithmic Verification
- Concurrency
- Programming Languages and Compilers
- Advanced Topics in Software Verification
- Advanced Functional Programming
- Advanced Topics in Software Verification

Concepts of Programming Languages

Programming Languages and Compilers

Advanced Compiler Construction

Algorithmic Verification

Concurrency
COURSE CONTENT

We will look at different programming paradigms
We will look at different programming paradigms

- imperative
- object-oriented
- functional
- concurrent
We will look at different programming paradigms

- imperative
- object-oriented
- functional
- concurrent

and language features: data abstraction, inheritance, overloading, . . .

Course content and material similar to course taught at CMU by Robert Harper, Frank Pfenning, and Princeton by Andrew Appel
MAIN TOPICS

→ An overview of the implementation of programming languages:
  - How do compilers/interpreters work?
  - What are their advantages/disadvantages?

Introduction to formal reasoning:
- judgements,
- induction, and
- transition systems

Semantics of programming languages
- static semantics
- dynamic semantics

Case studies using simplified variants of various programming languages
- TinyC
- MinHS
- Featherweight Java
An overview of the implementation of programming languages:
- How do compilers/interpreters work?
- What are their advantages/disadvantages?

Introduction to formal reasoning:
- judgements,
- induction, and
- transition systems
MAIN TOPICS

- An overview of the implementation of programming languages:
  - How do compilers/interpreters work?
  - What are their advantages/disadvantages?
- Introduction to formal reasoning:
  - judgements,
  - induction, and
  - transition systems
- Semantics of programming languages
  - static semantics
  - dynamic semantics
An overview of the implementation of programming languages:
  - How do compilers/interpreters work?
  - What are their advantages/disadvantages?

Introduction to formal reasoning:
  - judgements,
  - induction, and
  - transition systems

Semantics of programming languages
  - static semantics
  - dynamic semantics

Case studies using simplified variants of various programming languages
  - TinyC
  - MinHS
  - Featherweight Java
Data structures

- Products
- Sum-types
- Recursive types
- Data structures
  - Products
  - Sum-types
  - Recursive types
- Type Systems
  - type checking
  - type inference
  - subtyping and inheritance
  - Polymorphism and overloading
  - statically typed and dynamically typed languages
  - weakly and strongly typed languages
Data structures
- Products
- Sum-types
- Recursive types

Type Systems
- type checking
- type inference
- subtyping and inheritance
- Polymorphism and overloading
- statically typed and dynamically typed languages
- weakly and strongly typed languages

Abstract machines