# COMP3161/COMP9161

## Concepts of Programming Languages

**Semester 2 2016**

---

## Content


- Course Details
- Course Summary
- Course Constituents and Aims
- Student Learning Outcomes
- Assumed Knowledge
- Teaching Rationale
- Teaching Strategies
- Assessment
- Academic Honesty and Plagiarism
- Course Schedule
- Resources for Students
- Course Evaluation and Development


## Course Staff

| Staff Name | Role | Email | Phone |
|---|---|---|---|
| Manuel Chakravarty | Course Convenor | chak at cse.unsw.edu.au | 55546 |
| Liam O'Connor-Davis | Lecturer | liamoc at cse.unsw.edu.au | |

## Course Details

| | |
|---|---|
| **Course Code:** | COMP3161 |
| **Course Title:** | Concepts of Programming Languages |
| **Units of Credit:** | 6 |
| **Course WebSite:** | http://www.cse.unsw.edu.au/~cs3161/ |
| **Handbook Entry:** | http://www.handbook.unsw.edu.au/undergraduate/courses/2011/COMP3161.html |

## Course Summary

This course discusses and relates a range of programming language concepts and paradigms, including imperative, object-oriented, functional, concurrent & parallel programming. It covers the theoretical foundations (syntax, operatational, axiomatic and denotational semantics) as well as practical development and implementation aspects (dynamic and strong typing, polymorphism, overloading, automatic memory management, and multicore parallelism). Due to the course topic, a variety of programming languages will be studied and used. Assignments will be in the programming language Haskell, but no previous knowledge of Haskell is assumed.

## Course Constituents and Aims

### Lectures

The lectures will introduce you to new material, which is being re-enforced and practised in tutorials in smaller groups. In addition to the lecture slides, we will provide lecture notes. Students are required to read these notes and other reading material as advised during the lecture and/or on the course web page. Informal exercises that are not graded will occasionally be supplied. There are three hours of lectures each week.

### Tutorials

Tutorials start in Week 2. The last tutorial will be held in Week 13. Tutorials will discuss problems related to the lecture material and the assignments. Tutorial marks will be awarded based on participation. Tutorial problems will be handed out in advance and students will be asked to present the solution. Tutorial locations and times are published through NSS.

### Assignments

There will be two programming assignments. They will be due approximately around Week 6 and Week 10. Students will have about two weeks to understand each assignment and to develop a solution. Unless otherwise stated if you wish to submit an assignment late, you may do so, but a late penalty reducing the maximum available mark applies to every late assignment. The maximum available mark is reduced by 10% if the assignment is one day late, by 25% if it is 2 days late and by 50% if it is 3 days late. Assignments that are late 4 days or more will be awarded zero marks. So if your assignment is worth 88% and you submit it one day late you still get 88%, but if you submit it two days late you get 75%, three days late 50%, and four days late zero. Assignment extensions are only awarded for serious and unforeseeable events. Having the flu for a few days, deleting your assignment by mistake, going on holiday, work commitments, and so on do not qualify. Therefore aim to complete your assignments well before the due date in case of last minute illness, and make regular backups of your work.

## Student Learning Outcomes

After completing this course, you should

- be able to distinguish superficial (i.e., syntactic) from fundamental (i.e., semantic) differences of programming languages.
- understand a variety of programming language paradigms, including imperative, object-oriented, functional, concurrent & parallel programming.
- use the basic tools of language design and analysis, including formal languages and formal descriptions of static and dynamic semantics.
- understand and be able to use modern language features, such as dynamic and strong typing, polymorphism, overloading, automatic memory management, and multicore parallelism.

The courses exposes students to a scientific theory of programming languages based on type theory and operational semantics as a mathematically sound framework which enables an objective analysis of the properties of a large number of programming language features. The tutorial format will give students practice in the presentation of solutions to an audience of peers, and will challenge them to critique peer technical presentations. Furthermore, the whole course encourages critical examination and analysis of existing solutions.

## Assumed Knowledge

You need to have successfully completed the core programming, algorithm, and software development courses. You should be a confident coder and be prepared to study the elementary concepts of a new programming language in directed self-study. You also should know boolean logic and the basics of predicate logic, but there will also be the opportunity in the first few weeks to revise these topics.

### Course Prerequisites:

- COMP3161: COMP2011 or COMP2711
- COMP9161: COMP9024

## Teaching Rationale

The learning focus in this course is primarily on lectures and assignments, in addition to a mid term exam which you can use to test your preparedness for the final exam. While the assignments are graded and contribute to the final mark, their primary purpose is to facilitate learning by hands-on experience. Tutorial classes will give you a chance to discuss the lecture material and deepen your understanding.

## Teaching Strategies

### Lectures

Students are required to attend three hours of lectures. In addition, COMP3161 students are also required to attend the one hour tutorial class. COMP9161 students are also encouraged to attend the tutorials.

We are offering an optional fourth hour of lectures. We will use this, depending on your feedback and requests, to discuss advanced material which will not be tested in the exam, or as a remedial lecture.

### Tutorials

Tutorials aim to clarify ideas from lectures and to get you to explore the concepts in more depth. There will be a number of exercises set for each tutorial class. The aim of the class is not to simply get the tutor to give you the answers; the aim is to focus on just one or two of the exercises and work through them in detail, discussing as many aspects, alternative approaches, fine details, etc. as possible. You must be active and ask questions in tutorials. Ideally, students should run the entire tute themselves, with the tutor being a moderator and providing additional explanations.

### Assignments

The purpose of the assignments is to apply the theoretical concepts we discuss in the lecture to a practical problem. All assignments will be completed individually; this means that you should do them yourself without assistance from others, except for asking advice from the teaching staff (lecturers and tutors) in this course. As noted above, they are an important vehicle for learning the material in this

course. If you don't do them, or simply copy and submit someone else's work, you have wasted a valuable learning opportunity.

## Assessment

The assessment break down is as follows:

| | Comp3161 | COMP9161 |
|---|---|---|
| **Tutorial participation mark** | 20 marks | n/a |
| **Assignment 1** | 30 marks | 30 marks |
| **Assignment 2** | 30 marks | 30 marks |
| **Midsession exam** | 40 marks | 40 marks |

Both the practical component and the final examination are out of 100 (that is, for COMP3161 students, the sum of the class marks are divided by 1.2 to scale to 100). The final grade for the course is determined by the harmonic mean between the practical component and the final examination.

```
final = (2 * exam mark * class mark)/(exam mark + class mark)
```

### Tutorial participation mark

Attends of tutorials is compulsory for undergraduate students (COMP3161). Details of the tutorial participation mark will be discussed in the tutorials.

### Assignment 1 & 2

You can earn an equal amount of marks in the two assignments. Both assignments are individual; i.e., no team work of any kind is permitted. Completing and submitting both assignments is compulsory; i.e., each assignment has a core component and you will not be permitted to pass the course unless you have made a reasonable effort to solve the core component. A "reasonable effort" means that there may be bugs in your solution, but you must submit an at least partially working piece of adequately structured code. Midsession exam A 45 min midsession exam will be held in Week 5.

### Final exam

The final exam is a two hour written exam. Requests for a supplementary exam will only be considered where students (a) have completed all other course components to a satisfactory standard, (b) have been absent from the final exam, (c) and have submitted a fully documented request for special consideration to NSQ within three working days of the final exam.

### Examination hurdle

To achieve a passing final mark, a student needs to pass the exam. To pass the exam, a student needs to achieve 50% of overall marks awarded in the exam. Any student who fails the exam will automatically fail the entire course. No re-assessment will be awarded in this case.

### Assignment work

The assignments are an extremely important part of the course. They are an essential way of learning the practical skills you need to acquire. Any plagiarism in assignments will be severely punished and may result in an automatic Fail for the whole course. Read the plagiarism warning below for more details.

For each assignment, you will have approximately two weeks from release of the specification until the submission deadline. The specifications will be posted on the course web page.

Assignment work can be completed on the workstations at UNSW or on a computer at home. Your assignment must be able to run on the computers at UNSW so test them here if you develop them at home. Unless otherwise stated assignments must be submitted on-line from a school terminal using the give command. It is in your best interest to make regular backup copies of your work and (because of machine loads on deadline days, for example) to complete assignments well before their deadlines. Moreover, the electronic submission system "give" allows you to submit an assigment multiple times; only the last submission will be marked. We suggest that you submit a version once you have a partially complete solution and repeatedly submit whenever you improved your solution significantly. In particular, make sure that you submit your solution once you have complete the core component of each assignment. The core component for each assignment must be submitted to be able to pass the course.

## Academic Honesty and Plagiarism

**Plagiarism** is defined as "using the words or ideas of others and presenting them as your own". UNSW and CSE treat plagiarism as academic misconduct, which means that it carries penalties as severe as being excluded from further study at UNSW. There are several on-line sources to help you understand what plagiarism is and how it is dealt with at UNSW:

- Plagiarism and Academic Integrity
- UNSW Plagiarism Procedure
- CSE: Yellow Form (whose terms you have agreed to)

Make sure that you read and understand these. Ignorance is not accepted as an excuse for plagiarism. In particular, you are also responsible that your assignment files are not accessible by anyone but you by setting the correct permissions in your CSE directory and code repository, if using.

## Preliminary Course Schedule

| Week | Topic |
|---|---|
| 1 | Intro, Haskell |
| 2 | Preliminaries |
| 3 | Syntax |
| 4 | Semantics |
| 5 | MinHs |
| 6 | TinyC |
| 7 | Abstract Machines |
| 8 | Exceptions, Data Types |
| 9 | Polymorphism, Type Inference |
| 10 | Overloading and Subtyping |
| 11 | Concurrency |
| 12 | Parallelism |

## Resources for Students

- Lecture slides (available after the lecture), as well as additional lecture notes for some weeks, relevant papers and code.
- Haskell resources

## Course Evaluation and Development

This course is being continously improved and we will conduct a survey at the end of session to obtain feedback on the quality of the various course components. Your participation in the survey will be greatly appreciated.

The CATEI evaluation from the last time offering of this course showed that students were generally satisfied with all aspects of the course. Thus we intend to maintain the same style and structure for the up-coming offering.

The course content changed last year to include a stronger focus on concurrent and parallel programming, as we recognise the importance of current and emerging multicore architectures. We will continue to elaborate and polish this new material this semester.

*Gabriele Keller*
Last modified: Fri Jul 17 16:46:27 AEST 2016