Course Outline

This is more or less a copy of the Official course outline on webCMS, with more details added. If there is a discrepancy, prefer the webCMS version.

Course Staff

Staff Name	Role	Email
Christine Rizkallah	Lecturer in Charge	c.rizkallah at unsw.edu.au
Liam O'Connor	Lecturer	liamoc at cse.unsw.edu.au
Vivian Dang	Tutor	vivian.dang at unsw.edu.au
Matthew Di Meglio	Tutor	m.dimeglio at unsw.edu.au

Course Details

Course Code	COMP3161/COMP9164
Course Title	Concepts of Programming Languages
Units of Credit	6
Course Website	https://cse.unsw.edu.au/~cs3161
Handbook Entry	COMP3161; COMP9164

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 Timetable

Lectures

For weeks 1 through 5, lectures will be pre-recorded lectures released on Mondays and Thursdays on YouTube. The remaining weeks will be conducted Synchronously on Blackboard Collaborate, at the below listed times:

Day	Time	Location
Monday	16:00-18:00	Blackboard Collaborate
Thursday	12:00-14:00	Blackboard Collaborate

Tutorials

The following tutorials are on offer. Undergraduate students must attend at least one tutorial per week.

Day	Time	Location	Tutor
Thursday	14:00-15:00	Blackboard Collaborate	Vivian Dang
Thursday	15:00-16:00	Blackboard Collaborate	Vivian Dang
Thursday	16:00-17:00	Blackboard Collaborate	Matthew di Meglio
Thursday	17:00-18:00	Blackboard Collaborate	Vivian Dang
Friday	12:00-13:00	Blackboard Collaborate	Matthew di Meglio

Friday	13:00-14:00	Blackboard Collaborate	Matthew di Meglio

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 four hours of lectures each week, however, time permitting, we will also cover extension topics on occasion in the fourth hour of the week's lectures.

Tutorials

Tutorials start in Week 1. The last Friday tutorial will be held in Week 10. Tutorials will discuss problems related to the lecture material and the assignments. Tutorial problems will be handed out in advance and students may be asked to present the solution.

Assignments

There will be two programming assignments. They will be due approximately around Week 7 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, a student should be able to:

- distinguish superficial (i.e., syntactic) from fundamental (i.e., semantic) differences of programming languages.
- read, write and reason about programs in various programming language paradigms, including imperative, object-oriented, functional and concurrent programming.
- use the basic tools of language design and analysis, including formal languages and formal descriptions of static and dynamic semantics.
- 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 concepts of a new programming language in directed self-study.

The prerequisites of COMP3161 are COMP2521 (or COMP1927). The prerequisite of COMP9164 is COMP9024.

The course makes use of a number of discrete mathematics concepts. While it is possible to do well in the course without having done MATH1081 or COMP2111, familiarity with logic and set theory is valuable when studying this course.

Teaching Rationale

The learning focus in this course is primarily on lectures and assignments including a theoretical assignment 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

The lectures will introduce you to new material, which is being reinforced and practised in tutorials and assignments. The course follows no particular textbook, but reading material covering specific topics will be identified throughout the course. Students are required to study reading material as advised during the lecture and/or on the course web page.

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 marks breakdown is as follows:

Assignment 0	15%
Assignment 1	17.5%
Assignment 2	17.5%
Final Exam	50%

Assignments

Assignment Zero is a smaller assignment that focuses on theoretical topics and proofs. You will be given a series of proof exercises to complete and submit. Some of the questions given will be similar to exam questions. Assignments One and Two are programming assignments involving the implementation of a programming language.

All assignments are individual; i.e., no team work of any kind is permitted. 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 may lead to serious consequences. Read the plagiarism warning below for more details. For assignment 0, you will have approximately one week to complete it, but for assignments 1 and 2, 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 coding assignments 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.

Final Exam

The final exam details have yet to be determined in light of the COVID crisis. 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 Student Lifecycle within three working days of the final exam.

Student Conduct

The **Student Code of Conduct** Information, Policy) sets out what the University expects from students as members of the UNSW community. As well as the learning, teaching and research environment, the University aims to provide an environment that enables students to achieve their full potential and to provide an experience consistent with the University's values and guiding principles. A condition of enrolment is that students **inform themselves** of the University's rules and policies affecting them, and conduct themselves accordingly.

In particular, students have the responsibility to observe standards of equity and respect in dealing with every member of the University community. This applies to all activities on UNSW premises and all external activities related to study and research. This includes behaviour in person as well as behaviour on social media, for example Facebook groups set up for the purpose of discussing UNSW courses or course work. Behaviour that is considered in breach of the Student Code Policy as discriminatory, sexually inappropriate, bullying, harassing, invading another's privacy or causing any person to fear for their personal safety is serious misconduct and can lead to severe penalties, including suspension or exclusion from UNSW.

If you have any concerns, you may raise them with your lecturer, or approach the School Ethics Officer, Grievance Officer, or one of the student representatives.

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

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. Note also that plagiarism includes paying or asking another person to do a piece of work for you and then submitting it as your own work.

UNSW has an ongoing commitment to fostering a culture of learning informed by academic integrity. All UNSW staff and students have a responsibility to adhere to this principle of academic integrity. Plagiarism undermines academic integrity and is not tolerated at UNSW. Plagiarism at UNSW is defined as using the words or ideas of others and passing them off as your own.

If you haven't done so yet, please take the time to read the full text of

- UNSW's policy regarding academic honesty and plagiarism

The pages below describe the policies and procedures in more detail:

- Student Code Policy
- Student Misconduct Procedure
- Plagiarism Policy Statement
- Plagiarism Procedure

You should also read the following page which describes your rights and responsibilities in the CSE context:

- Essential Advice for CSE Students

Course Schedule

Please see the separate course schedule page for a detailed breakdown of course content over the 10 weeks of session.

Resources for Students

There is no textbook for the course. As tutorial and reference for Haskell, you can use any of these books:

- Thinking Functionally with Haskell, by Richard Bird, Cambridge University Press.
- Haskell Programming From First Principles by Christopher Allen and Julie Moronuki, Gumroad.
- Programming in Haskell, 2nd Edition by Graham Hutton, Cambridge University Press.

- Real World Haskell by Bryan O'Sullivan, Don Stewart, and John Goerzen, O'Reilly Media.
- Learn You a Haskell for Great Good! by Miran Lipovača, No Starch Press.

The last two books in this list are available as online ebooks for free.

Further reading material will be announced in the lecture and/or on the course web page.

In addition, the Learning Haskell tutorial is written by former lecturers of this course and covers the basics of Haskell programming necessary to get started in COMP3161.

Getting Help

Questions regarding the course material, assignments, exercises, and general administrative questions should be asked on the course forum (accessible from the course web page), where answers benefit the whole class. Alternatively, approach the lecturer after class. To discuss matters concerning your personal performance, please send an email to the course account `cs3161@cse.unsw.edu.au`. For identification purposes, if you wish to send email concerning the course, you must send the mail from your CSE or UNSW student account (not from GMail, Yahoo, Bigpond or similar), and include your student id and your full name.

Course Evaluation and Development

This course is being continuously 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.

Student feedback over the last years has generally been positive with a high student satisfaction rate.

We also do appreciate constructive criticism during the semester, as (in contrast to course surveys at the end), this allows us to address potential problems straight away.