Course Outline

Course Details

Course Code	COMP3141
Course Title	Software System Design and Implementation
Units of Credit	6
Course Website	https://www.cse.unsw.edu.au/~cs3141
Handbook Entry	COMP3141 UNSW Handbook Entry

Staff

Name	Role	Contact
Liam O'Connor	Lecturer in Charge	liamoc@cse.unsw.edu.au
Christine Rizkallah	Lecturer	

Course Summary

This course presents a variety of programming techniques that make use of semi-formal and formal methods for the design and implementation phases of real-world software system development. It introduces approaches to testing informed by formal designs, and it discusses trade-offs between static and dynamic approaches to improving software correctness. Throughout the course, the discussed methods are supported by software tools that assist in managing design, implementation, and testing. The course content is illustrated by weekly programming case studies and regular practical exercises. Central topics are the use of functional programming, logical properties and types to inform program design, implementation, validation,

and verification. The course will introduces students to the strongly-typed Haskell programming language. No previous knowledge of Haskell is assumed.

As with any course which teaches a potentially unfamiliar language, this course involves a significant amount of programming, both during and outside of lectures.

Course Timetable

Day	Time	Location	Lecturer
Tuesday	14:00-16:00	Law Theatre G04	Liam O'Connor
Wednesday	09:00-11:00	Law Theatre G04	Christine Rizkallah

Outcomes

After completing this course, you should

- be comfortable with using Haskell and functional programming to write practical software.
- be able to specify programs in terms of logical properties and functional correctness.
- distinguish and balance the trade-offs between static methods (such as formal methods and type systems) and dynamic methods (such as testing) in assisting software design and implementation.
- understand the role of types in program design, implementation, validation, and verification.
- be able to use a variety of tools based on formal specifications of logical properties.

The course gradually exposes students to a mathematically founded approach to specifying and implementing software systems. It develops basic skills required to engineer software with high confidence in the correctness of the final product. 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 COMP3141 are COMP1927 or COMP2521, or a mark of at least 65 in COMP1921.

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 COMP3141.

Teaching Strategies

Lectures

The lectures will introduce you to new material, which is being reinforced and practised in weekly exercises and larger 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.

There are three to four hours of lectures each week. The first two hour slot will be an *introduction* lecture, generally delivered by Liam. This will introduce you to the various concepts in the course. The second two hour slot will be a *reinforcement* lecture, generally delivered by Christine. This will apply the concepts introduced by Liam in the previous lecture to various examples.

Although there are no labs for this course, it does involve a significant amount of Haskell programming. For this reason, it's important to stay up to date with the Haskell level of the rest of the cohort. If you find yourself falling behind, please arrange a consultation with a lecturer to get caught up.

Exercises

Weekly marked programming exercises start in Week 2. There will be 7 exercises, including one art peer review assessment run from weeks 4 to 6. These exercises will reinforce the material discussed in the week's lecture.

Students have approximately one week to submit a solution. Submitting solutions to the exercises is compulsory, and solutions will be automatically marked. Solutions to

exercises will not be accepted late. Automarking results are binding. Exercises will generally not be manually remarked. There are no extensions to exercises, excepting special considerations.

Solutions to the exercises will be discussed in the second half of lectures each week.

Quizzes

A total of 8 weekly marked quizzes will be made available, starting in Week 1. They will assess the theoretical lecture content for that week in a series of multiple choice questions, and will be due on the Friday of the following week. For example, the quiz for the Week 1 lectures will be due on the Friday of week 2. Students are encouraged to submit the quiz soon after the lectures for the week have concluded, as part of a weekly review.

The quizzes are available on the course website. Students can submit as many times as they like up until the due date, however solutions are shown after the due date, so late submission of the quizzes will not be accepted.

These quizzes are all automatically marked.

In general, the difficulty of the quizzes is not indicative of any other component such as the exam. The quizzes help to consolidate lecture content, but sometimes the quiz questions will require some lateral thinking or referring to the lecture code, slides, or notes.

Assignments

There will be two practical assignments. They will be released approximately around Week 4 and Week 8. Students will have between one and two weeks to understand each individual 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.

Assignments are being marked automatically. You need to make sure to follow the instructions closely. Failure to follow the details of the assignment specification is no reason for re-marking, even if small mistakes lead to a substantial loss of marks.

Assessment

Assessment consists of a practical component and a final examination. The break down of the practical component is as follows:

- Assignment 1: 20 marks
- Assignment 2: 20 marks
- Weekly assessments, consisting of:
  - Weekly Exercises: 40 marks
  - Weekly Quizzes: 20 marks

Both the practical component and the final examination are out of 100. The final grade for the course is determined by the mean between the practical component and the final examination.

Practical Work

Assignments and exercises are an important part of the course. They are an essential way of learning the practical skills you need to acquire. Any plagiarism in assignments or exercises 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 up to 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 properly run on the computers at UNSW; so, test them at UNSW 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 assignment 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 completed the core component of each assignment.

Assignments

Each assignment is individual; i.e., no team work of any kind is permitted. Completing and submitting all 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.

Exercises

There will be a number of programming exercises distributed throughout the term. Each is marked with equal weighting. Each exercise is individual; i.e., no team work of any kind is permitted.

Quizzes

There will be 8 weekly quizzes. Each quiz is individual; i.e., no team work of any kind is permitted. Each quiz is given varying weighting, for a total of 20 marks. The marks for quizzes are capped such that a student technically only must complete seven of the eight quizzes to get full marks for the quiz component (although completing all quizzes will enable students to regain marks they lost previously).

Final Examination

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 student lifecycle 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 40% 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.

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

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 COMP3141.

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 `cs3141@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.