Course Outline

Contents

- Course Details
- Course Summary
- Assumed Knowledge
- Student Learning Outcomes
- Teaching Strategies
- Teaching Rationale
- Student Conduct
- Assessment
- Course Schedule
- Resources for Students
- Course Evaluation and Development

Course Details

Course Code	COMP3141
Course Title	Software System Design and Implementation
Convenor	Liam O'Connor (/users/z3253158)
Admin	Liam O'Connor (/users/z3253158)
Classes	**Lectures** Lectures are online via Blackboard collaborate, or released as asynchronous YouTube videos. Timetable for all classes (/COMP3141/20T2/timetable)
Consultations	By email appointment
Units of Credit	6
Course Website	http://cse.unsw.edu.au/~cs3141/20T2/ (http://cse.unsw.edu.au/~cs3141/20T2/)
Handbook Entry	http://www.handbook.unsw.edu.au/undergraduate/courses/current/COMP3141.html (http://www.handbook.unsw.edu.au/undergraduate/courses/current/COMP3141.html)

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.

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.

Student Learning Outcomes

After completing this course, students will:

1. be comfortable with using Haskell and functional programming to write practical software.
2. be able to specify programs in terms of logical properties and functional correctness.
3. 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.
4. understand the role of types in program design, implementation, validation, and verification.
5. be able to use a variety of tools based on formal specifications of logical properties.

This course contributes to the development of the following graduate capabilities:

Graduate Capability	Acquired in
Scholars capable of independent and collaborative enquiry, rigorous in their analysis, critique and reflection, and able to innovate by applying their knowledge and skills to the solution of novel as well as routine problems	Lectures, Assignments
Entrepreneurial leaders capable of initiating and embracing innovation and change, as well as engaging and enabling others to contribute to change	Interactive Lectures
Professionals capable of ethical, self- directed practice and independent lifelong learning	Assignments
Global citizens who are culturally adept and capable of respecting diversity and acting in a socially just and responsible way	Interactive Lectures

Teaching Strategies

- **Lectures** are released as asynchronous YouTube videos which introduce new concepts.
- **Practice Lectures** are an interactive lecture on Blackboard Collaborate, where students will work with the lecturer to solve problems and develop homework solutions.
- 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 practice lectures lectures each week.

- **Quizzes** act as a forcing function requiring students to digest and comprehend the topics introduced each week. 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** allow students to apply their skills to larger problems. There will be two practical assignments. 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.

Teaching Rationale

This is a course about programming and formal reasoning, and to develop both of these skills requires extensive practice. That is why we devote over half of the course --- practice lectures, quizzes, assignments and exercises --- to give students opportunities to put their skills into practice. This applied learning approach has proven to be very successful in achieving our desired graduate outcomes.

Student Conduct

The **Student Code of Conduct** (Information (https://student.unsw.edu.au/conduct) , Policy (https://www.gs.unsw.edu.au/policy/documents/studentcodepolicy.pdf)) 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 (mailto:ethics-officer@cse.unsw.edu.au) , Grievance Officer (mailto:grievance-officer@cse.unsw.edu.au) , or one of the student representatives.

Plagiarism is defined as (https://student.unsw.edu.au/plagiarism) 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 (https://student.unsw.edu.au/plagiarism)
- UNSW Plagiarism Procedure (https://www.gs.unsw.edu.au/policy/documents/plagiarismprocedure.pdf)

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 (https://student.unsw.edu.au/plagiarism)

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

- Student Code Policy (https://www.gs.unsw.edu.au/policy/documents/studentcodepolicy.pdf)
- Student Misconduct Procedure (https://www.gs.unsw.edu.au/policy/documents/studentmisconductprocedures.pdf)
- Plagiarism Policy Statement (https://www.gs.unsw.edu.au/policy/documents/plagiarismpolicy.pdf)
- Plagiarism Procedure (https://www.gs.unsw.edu.au/policy/documents/plagiarismprocedure.pdf)

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

- Essential Advice for CSE Students (https://www.engineering.unsw.edu.au/computer-science-engineering/about-us/organisational-structure/student-services/policies/essential-advice-for-cse-students)

Assessment

Item	Topics	Due	Marks	Contributes to
Quizzes	All topics	Weeks 2,3,4,5,7,8,9,10	10%	CLOs 2,3,4
Assignment 1	Specifications and Haskell	Week 5	10%	CLOs 1,2,5
Assignment 2	Types and Effects	Week 9	10%	CLOs 1,3,4,5.
Exercises	All topics	All Weeks	20%	CLOs 1,2,3,4,5
Final Exam	All topics	Exam period	50% (with 40/100 hurdle to pass the course)	CLOs 2,3,4

Course Schedule

Week	Lectures
1	Course Overview, Functional Programming

2	Data Types and Type Classes, Reasoning about Haskell
3	Property-based Testing and QuickCheck
4	Devising program properties
5	Controlling Effects, State Spaces, Errors
6	Flexibility Week
7	Applicative Functors and Monads
8	Rich types, GADTS,
9	Polymorphism and Parametricity, Type Theory
10	Guest Lecture or Extension Topics, Revision

Resources for Students

There is no set textbook for the course. We draw on the following books for some materials in the course:

- Thinking Functionally with Haskell, by Richard Bird, Cambridge University Press.
- Data Refinement: Model-Oriented Proof Methods and their Comparison, by Kai Engelhardt and W.P. de Roever, Cambridge University Press

The first book is recommended for students who are struggling with Haskell. The following books are also available, which some students have found helpful:

- 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.

Additionally, the former lecturers of this course wrote the Learning Haskell (http://learn.hfm.io/) tutorial which we also highly recommend.

Course Evaluation and Development

This course is evaluated each session using the myExperience system.

In the previous offering of this courses, students noted that the coverage of Monads was confusing as the motivation for them (effects) was presented afterwards. We are rearranging the order of this presentation this time to address that problem.

Other course feedback has been overwhelmingly positive.

Resource created about a year ago (Sunday 24 May 2020, 11:41:44 PM), last modified 11 months ago (Sunday 02 August 2020, 04:07:42 PM).

Comments

(/COMP3141/20T2/forums/search?forum_choice=resource/47558)

(/COMP3141/20T2/forums/resource/47558)

There are no comments yet.