Resources / Course Outline

Course Outline

Contents

- 1. Course Details
- 2. Course Summary
- 3. Teaching Strategies
- 4. Assessment
- 5. Schedule / Timetable
- 6. Student Conduct
- 7. Resources for Students
- 8. Course Evaluation and Development

1. Course Details

Find information relating COVID-19 and this offering here (https://www.covid-19.unsw.edu.au/safe-return-campus-faqs) and here (https://edtech.eng.unsw.edu.au/c19mess/comms.html) .

Course Code	COMP2521
Course Title	Data Structures and Algorithms
Contact Email	cs2521@cse.unsw.edu.au
Convenor	Hayden Smith (https://webcms3.cse.unsw.edu.au/users/z3418003)
Admin	Kevin Luxa (https://webcms3.cse.unsw.edu.au/users/z5074984)
Units of Credit	6
Timetable	https://webcms3.cse.unsw.edu.au/COMP2521/21T1/time... (https://webcms3.cse.unsw.edu.au/COMP2521/21T1/timetable)
Course Website	http://cse.unsw.edu.au/~cs2521/21T2/ (http://cse.unsw.edu.au/~cs2521/21T2/)
Handbook Entry	http://www.handbook.unsw.edu.au/undergraduate/courses/current/COMP2521.html (http://www.handbook.unsw.edu.au/undergraduate/courses/current/COMP2521.html)

2. Course Summary

The goal of this course is to deepen your understanding of data structures and algorithms and how these can be employed effectively in the design of software systems. It is an important course in covering a range of core data structures and algorithms that will be used in context in later courses. You explore these ideas in lectures, tutorials, lab classes, quizzes and assignments.

Assessment involves labs, quizzes, assignments and a final exam involving both practice and theory. At the end of the course, we want you to be a solid programmer, with knowledge of a range of useful data structures and programming techniques, capable of building significant software systems in a team environment, and ready to continue with further specialised studies in computing.

The aim of this course is to get you to **think like a computer scientist** . This certainly sounds like a noble goal... but what does it really mean? How does a *scientist* , let alone a computer scientist, actually think?

What many types of scientists try to do is understand natural systems and processes: a geologist, for example, tries to understand the structure of the earth; a biologist tries to understand living organisms; a chemist tries to understand materials and reactions, and so on.

Computer scientists don't, as the name might suggest, simply try to understand the structure and behaviour of computers, but are more concerned with understanding software systems (and the interaction between the software and the hardware on which it runs). Also, unlike other scientists, computer scientists frequently build the objects that they study.

During this course, we'll be looking at ways of creating, analysing and understanding software. Ultimately, you should be able to answer the question, **"is this piece of software any good?"** and be able to provide sound reasons to justify your answer.

This course follows on from introductory C programming courses: COMP1511, COMP1917, or COMP1921. We cover additional aspects of the C programming language that were not covered in those courses, and also look at some programming tools which were not covered (in detail) earlier. However, this course is not simply a second C programming course: the focus is on the ideas and abstractions behind the data structures and algorithms that are used.

COMP2521 is a critical course in the study of computing at UNSW, since it deals with many concepts that are central to future studies in the area. Whether you are studying Computer Science, Software Engineering, Bioinformatics, Computer Engineering, or even a discipline outside the realm of computing, understanding a range of algorithms and data structures and how to use them will make you a much more effective computing problem solver in the future.

Computer science is, to a large extent, a practical discipline, and so COMP2521 has an emphasis on practice. Lectures will include exercises where we examine the practice of developing and analysing programs. The aim of tutorials is to develop analysis and understanding via practical case studies. Lab classes also provide practice in program development and analysis. Assignments provide large case studies of software development.

2.1. Assumed Knowledge

The official pre-requisite for this course is either COMP1511 or COMP1917 or COMP1921.

Whether or not you satisfy the pre-requisite, we assume that:

- you can program in the C programming language, and are familiar with arrays, strings, pointers, and dynamic memory allocation
- you are able to design, implement, debug, test and document small C programs (up to several hundred lines of code)
- you are familiar with the Linux environment on the CSE computers

2.2. Student Learning Outcomes

After completing this course, students will:

- be familiar with fundamental data structures and algorithms
- be able to analyse the performance characteristics of algorithms
- be able to measure the performance behaviour of programs
- be able to choose/develop an appropriate data structure for a given problem
- be able to choose/develop appropriate algorithms to manipulate this data structure
- be able to reason about the effectiveness of data structures and algorithms for solving a given problem
- be able to package a set of data structures and algorithms as an abstract data type
- be able to develop and maintain software systems in C that contain thousands of lines of code

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

Graduate Capability	**Acquired in**
scholarship: understanding of their discipline in its interdisciplinary context	lectures, assignments
scholarship: capable of independent and collaborative enquiry	lab work, assignments
scholarship: rigorous in their analysis, critique, and reflection	tutorials
scholarship: able to apply their knowledge and skills to solving problems	tutorials, lab work, assignments
scholarship: ethical practitioners	all course-work, by doing it yourself
scholarship: capable of effective communication	tutorials
scholarship: digitally literate	everywhere in CSE
leadership: enterprising, innovative and creative	assignments
leadership: collaborative team workers	lab work, assignments
professionalism: capable of operating within an agreed Code of Practice	all prac work

3. Teaching Strategies

This course uses the standard set of practice-focused teaching strategies employed by most CSE foundational courses:

- Lectures
- Tutorials
- Laboratories
- Help Sessions
- Assignments
- Final Exam

This course aims to provide the students with a strong foundation in the fundamental principles and practices of software engineering that will prepare them for the advanced software engineering workshops. As such, a broad range of key software engineering topics will be taught and reinforced through a group project, that will enable students to apply the theoretical concepts acquired to solve a practical software engineering problem. An agile software delivery style has been chosen for the implementation of the group project, to make students familiar with modern agile development methodologies.

For instructions about how to use Blackboard Collaborate, see here (https://webcms3.cse.unsw.edu.au/COMP2521/21T2/resources/60440) .

3.1. Lectures

Each week, there will be four hours of lectures during which theory, practical demonstrations and case studies will be presented. Lectures convey a small amount of information about the course content, but their main aim is to try to stimulate you to think about concepts and techniques. Lectures will be delivered online via **YouTube Live** . Lecture slides will be available on the course web page.

Lectures aim to convey basic information about the course content and to model the practices and techniques involved in software development (*i.e.* , we do demos). The most important components of the course, however, are the tutorials, labs and assignments. Tutorials aim to clarify and refine the knowledge that you got from lectures, and from reading the textbook and notes. Labs and assignments are where you get to put together and practise all of the ideas from the lectures, tutes and text. The only way to develop the skills to do effective software development is by practising them. If you slack off on the assignments and lab exercises (or, worse, rely on someone else to do them for you), you're wasting the course's most valuable learning opportunities.

The university requires us to assess how well you have learned the course content, and the primary approach to achieving this is via a final exam. A final exam is the ultimate summative assessment tool; it gives you a chance, at the end of the course, to demonstrate everything that you've learned. Labs and assignments are a *learning* tool, not an assessment tool, so, in an ideal world, we would have them as pure learning exercises and award no marks for them. However, to give a more concrete incentive to do them (in a timely fashion), there are marks tied to them.

3.2 Tutorials

Tutorials aim to clarify ideas from lectures and to get you to think about design/analysis issues. 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.

Tutorials will be run via Blackboard Collaborate.

3.3. Laboratories

In terms of the 2 hour period each week allocated for labs, these sessions run immediately after your tutorial.

Lab classes aim to give you practice in problem-solving and program development. Each week, there will be one or two small exercises to work on. These exercises will be released in the week preceding the lab class. Labs will be done individually or in pairs, and you and your partner should discuss the exercises *before* going to the lab, to maximise the usefulness of the class. The exercises will need to be submitted (for our records) and will be assessed by your tutor. During the lab, your tutor will provide feedback on your approach to the problem and on the style of your solution.

Important: Although you are required to submit your lab exercises, marks for lab exercises can **only** be obtained by demonstrating your solution to a tutor during your lab session in either the week of your lab exercise or the following week. Simply submitting the lab without demonstrating your solution to a tutor is not worth any marks.

3.4. Help Sessions

Help sessions are unprepared drop-in "clinics" where students and groups can go to seek help about course related matters, whether that be the project, tutorials, or labs. Current tutors or lab assistants will supervise each help session.

Help sessions will be run via Blackboard Collaborate.

3.5. Assignments

In the assignments, you will work on more substantial (hundreds of lines of code) programming exercises. All assignments will be completed individually. As noted above, assignments are the primary 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.

3.6. Final Exam

There will be a centrally timetabled final exam which will in your UNSW exam timetable. The exam may contain a mixture of multiple choice questions, short answer questions, and programming exercises. More specific details of the exam will be provided through the course.

If you cannot attend the final exam because of illness or misadventure, then you must submit a Special Consideration request, with documentation, through MyUNSW within 72 hours of the start of the exam exam. If your request is reasonable, then you will be awarded a Supplementary Exam. No supplementary exams will be provided for students who score marks 49 or below on grounds of being "close" to a pass.

---

4. Assessment

Item	Topics	Due	Marks	Contributes to
Quizzes	All topics	Weeks 2,3,4,5,7,8,9	10%	1, 2, 3, 4, 5, 6, 7
Assignment 1	Trees	Week 7	15%	4, 5, 7, 8
Assignment 2	Graphs	Week 10	20%	4, 5, 7, 8
Labs	All topics	Weeks 1,2,3,4,5,7,8,9	15%	1, 3, 4, 5
Final Exam	All topics	Exam period	40%	1, 2, 3, 4, 5, 6, 7, 8

Each **quiz** contributes 1.6 marks, allowing for a total quiz result of 11.2 marks. We will cap your quiz mark at 10, though, allowing you to get some questions wrong throughout the term.

Each **lab** contributes 2 marks, allowing for a total lab result of 16 marks. We cap your lab mark at 15, though, allowing you to get some lab questions wrong throughout term. **Labs have no late penalty** , **because late submissions are not accepted** .

For the **exam** , the marking criteria will be specified in the exam specification which can be (once released) found here (https://webcms3.cse.unsw.edu.au/COMP2521/21T2/resources/61763) . The document " (https://www.engineering.unsw.edu.au/computer-science-engineering/about-us/organisational-structure/student-services/policies/essential-advice-for-cse-students) 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) states the supplementary assessment policy for the School of CSE. Please take the time to read it carefully. If you are granted a supplementary examination, then it will be centrally timetabled. If you think that you may be eligible for a supplementary exam, then make sure you are available on that day. It is your responsibility to check at the student office for details of supplementary examinations.

```
quizzes = mark for quizzes (out of 10)
labs = mark for lab exercises (out of 15)
ass1 = mark for assignment 1 (out of 15)
ass2 = mark for assignment 2 (out of 20)
finalExam = mark for final exam (out of 40)

mark = quizzes + labs + ass1 + ass2 + finalExam
grade = HD|DN|CR|PS|FL
```

---

5. Course Schedule / Timetable

Week	Topics
1	Analysis of algorithms
2	Recursion, Analysis of ADT (multiple) implementations, Trees
3	Binary search trees (BST), Balanced search trees
4	Search tree algorithms
5	Graph ADT, Graph algorithms (1)
6	Flex week
7	Graph algorithms (2)
8	Sorting
9	Hashing, Heaps
10	Tries, Course review and review exercises

Tutorial/Laboratories: Each topic will be dealt with in tutes/labs in the week after it is covered in lectures.

The schedule for this course is outlined clearly in the timetable for lectures (https://webcms3.cse.unsw.edu.au/COMP1531/21T1/resources/56483) , tutorials/labs (https://webcms3.cse.unsw.edu.au/COMP1531/21T1/resources/56484) , and help sessions (https://webcms3.cse.unsw.edu.au/COMP1531/21T1/resources/56485) .

---

6. 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)

At any time in this course, when you push code to gitlab from your own machine (locally or on CSE account), you are acknowledging that this code you push is your own work, except here permitted by the originality rules for this course and this assessment. You acknowledge that this pushed code has not been submitted for academic credit elsewhere, and you acknowledge that you have read an understood the University Rules in respect of Student Academic Misconduct.

---

7. Resources for Students

COMP2521 follows the contents of the pair of books:

- *Algorithms in C, Parts 1-4: Fundamentals, Data Structures, Sorting, Searching* (3rd Edition) by Robert Sedgewick, published by Addison-Wesley
- *Algorithms in C, Part 5: Graph Algorithms* (3rd Edition) by Robert Sedgewick, published by Addison Wesley

These two books are available as a bundle from the UNSW bookshop. They are expensive, but are useful well beyond this course, and will serve as a useful reference on the bookshelf of any serious programmer.

You may also be able to find on-line resources related to the textbooks. Robert Sedgewick has a series of videos on the topics in this course, but unfortunately they all seem to be in Java (which he has used for the new edition of his book). If you find any useful on-line resources, please let me know and we will add them to the *Resources* section of the course web site (with credit to the finder).

This website also has links to the auxiliary material/documentation that you will need for the course. Solutions for all tutorial questions and lab exercises will also be made available. We will review quiz and assignment solutions in the lectures.

---

8. Course Evaluation and Development

Every term, student feedback is requested in a survey using UNSW's myExperience online survey system where the feedback will be used to make improvements to the course. Students are also encouraged to provide informal feedback during the session, and to let course staff know of any problems as soon as they arise. Suggestions will be listened to openly, positively, constructively, and thankfully, and every reasonable effort will be made to address them.

This term, our focuses will include:

- Trying to speed up the time required for tutors to mark labs
- Releasing lab solutions to students
- Simplifying and modernising some lecture content
- Trimming down the number of topics taught in the course

Resource created 6 months ago (Friday 14 May 2021, 01:15:23 AM), last modified 5 months ago (Saturday 05 June 2021, 08:13:34 PM).

Comments

(/COMP2521/21T2/forums/search?forum_choice=resource/60367)

(/COMP2521/21T2/forums/resource/60367)

There are no comments yet.