Course Details

Course Codes	COMP2041/COMP9044
Course Title	Software Construction
Units of Credit	6
Course Website	https://cse.unsw.edu.au/~cs2041
Handbook	COMP2041 COMP9044
Lecturer/Convener/Admin	Andrew Taylor `andrewt@unsw.edu.au`
Lectures	Tuesday 18:00-20:00 Matthews Theatre A
	Thursday 16:00-18:00 Matthews Theatre A
Tut-lab	3 hour slot selected when you enrol (weeks 1-10)

Lecture Recordings

All lectures will be recorded. Unfortunately UNSW's lecture recording system can only be accessed via Moodle. Moodle will not be used for any other course activities.

Online Stream

An online (WEB) stream is available. Students in this stream are welcome to attend lectures **if** there are spare seats in the lecture theatre. In past sessions there have been spare seats. Students in the WEB stream may need to rely on lecture recordings and the material placed on the web and should consider carefully whether this is sufficient for them to successfully complete the course.

Communication with Course Staff

Sometimes urgent information may be sent to you by email. Make sure you pay careful attention to any email you receive.
All official email will be sent to your UNSW email address. It's essential you read this email address regularly. If you forward your email, please be careful to do so correctly & test the forwarding.

Additional information will be provided in the Course Forum. You should check the course regularly. It is the best place to ask questions about the course.

Consultations times vary through session and are listed on the course home page. The forums is the best place to ask questions about the course.

Course Schedule

Week 1	Course Introduction. Unix Filters
Week 2	Unix Filters continued. Shell programming.
Week 3	Shell programming continued. Introduction to Version Control with Git.
Week 4	Perl programming.
Week 5	Perl programming continued.
Week 6	Programming for the web. Introduction to CGI. JavaScript programming **Assignment 1** due
Week 7	JavaScript programming continued
Week 8	JavaScript programming continued
Week 9	Tools for Performance Analysis & Deployment.

Week 10	Revision Exam **Assignment 2** due
Exam Period	**Final Exam**

Course Aims

This course is designed for students who have mastered the basics of programming. It aims to broaden your knowledge of techniques and tools for software construction.

Learning Outcomes

By the end of the course, you should have these attributes which will be useful to you for the remainder of your studies and after graduation:

- have practical experience in programming with the Unix shell, Perl and JavaScript
- have a broader & deeper knowledge of building software systems
- more appreciation of the use of specific technologies and strategies during software development
- exposure to tools for version control, performance improvement, configuration and debugging
- improvement of your ability to articulate & communicate concepts related to programming & systems

Assumed Knowledge:

COMP2041/9041 assumes that you have a sound understanding of a procedural programming language such as C or Python and can:

- produce a correct procedural program from a spec
- understand fundamental data structures + algorithms (sorting, searching)
- appreciate use of abstraction in computing

For undergraduate (COMP2041) students, the above material will typically have been covered in COMP1511 (or its predecessor COMP1917) and COMP2521 (or its predecessor COMP1927)

Students who have not taken COMP2521 may need to do some reading on sorting and searching.

For postgraduate (COMP9041) students, the above material will have been covered in COMP9021 and COMP9024 or in equivalent courses in their undergraduate degree.

A significant fraction of the class has used the programming language C and a limited amount of knowledge of the C programming language may be assumed during some lecture examples. Students who have not programmed C (or C++ or Java) might need to do some extra reading to understand these examples.

Almost all the class has used the programming language Python previously. Students who have not programmed in Python previously can expect to do extra work for assignment 2 where you may not modify or extend supplied Python code.

Teaching Rationale & Strategies

This course has a heavy practical orientation. Do not take this course unless you wish to do a very large amount of coding. This will place large demands on your time particularly in the second half of session.

Lectures

Lectures will be used to present the theory and practice of the techniques and tools in this course. There will be extensive use of practical demonstrations during lectures. Lecture notes will be available on the course web pages before each lecture.

Tutorials

From week 1 you will also be expected to attend a one-hour tutorial session to clarify ideas from lectures and work through exercises based on the lecture material. You should make sure that you use them effectively by examining in advance the material to be covered in each week's tutorial, by asking questions, by offering suggestions and by generally participating. The tutorial questions will be posted on the Web in the week before each tutorial. There are no marks for tutorial attendance but note students with borderline results will not be offered supplementary assessment unless they have attended tutorials (or are a COMP9041 student who has obtained exemption from the tut-lab component).

Laboratory Classes

Following the tutorial class each week, there will be a two-hour laboratory class, during which you will work on a variety of small practical problems involving the tools introduced in lectures. Because this course is practical in nature, laboratory classes are a very important component. If you do not put a great deal of effort into the lab classes you risk failing the final exam.
Most lab exercises will be automarked. There will be partial marks for attempts which do not pass autotests.

You will need to submit the lab exercise by Tuesday 17:59 midnight to obtain the mark. Tutors will separately provide feedback on your code.

Submission of any work that it is not your own will result in an automatic mark of zero for the entire lab component.

Several labs will be contain exercises which will be assessed during the lab

COMP9041 students may request exemption from the lab component. This is only recommended if you have a major external commitments such as full-time work and are very confident you can master the course material. Typically only 1 or 2 students each year take this option. If you wish to be exempt from the lab component, you must request exemption by emailing the lecturer before the end of week 2.

Weekly Programming Tests

There will be weekly tests from weeks 3-10 designed to give you timely & realistic feedback of your understanding of the course material.
These will be conducted in your own time under self-enforced exam-like conditions.

Each tests will specify the conditions but typically these will include

1. No assistance from any person.
2. A time limit of 1 hour.
3. No access to materials (written or online) except specified language documentation or man pages.

Each programming test will be automarked.

Your mark for the test component will be the sum of the best 6 of 8 test marks.

Any violation of the test conditions will results in a mark of zero for the entire programming test component.

Assignments

There are two assessable programming assignments. Assignments give you the chance to practice what you have learnt on relatively large problems (compared to the small exercises in the labs). Assignments are a very important part of this course, therefore it is essential that you attempt them yourself.
The first assignment will be a shell application due start week 7, the second assignment will be a JavaScript application and will be due end week 10..

- Assignment 1 (Submission, Week 7) 15%
- Assignment 2 (Submission, Week 10) 15%

The assignment weighting and deadlines may be slightly varied when the assignment designs are complete.

Late assignments submissions will be penalized. The exact penalty will be specified in the assignment specification - often it is 2% reduction in maximum mark for every hour late.

Assessment

Component	**Week**	**Weight**
Labs	Weeks 1-10	9%
Programming Tests	Weeks 3-10	6% (best 6 of 8 tests)
Assignments	Weeks 7 & 10	30%
Final Exam	exam period	55%

There a hurdle requirement for this course that you perform satisfactorily on the final exam to pass the course.

Final Exam

There will be a three-hour primarily practical exam held in the CSE labs during the exam period.
The exam will contain short answer questions which may require you to read code.

It will also contain implementation tasks where you will be required to write code.

During this exam you will be able to execute, debug and test your answers. The implementation tasks will be similar to those encountered in lab exercises

You will not be expected to remember all the details of programming languages used in the course; reference information will be provided along with the exam paper, giving a summary of any language that we expect you to use.

There a hurdle requirement for this course that you score at least 23/55 on the final exam.

Academic Honesty and Plagiarism

What is Plagiarism?
Plagiarism is the presentation of the thoughts or work of another as one's own.* Examples include:

- direct duplication of the thoughts or work of another, including by copying material, ideas or concepts from a book, article, report or other written document (whether published or unpublished), composition, artwork, design, drawing, circuitry, computer program or software, web site, Internet, other electronic resource, or another person's assignment without appropriate acknowledgement;
- paraphrasing another person's work with very minor changes keeping the meaning, form and/or progression of ideas of the original;
- piecing together sections of the work of others into a new whole;
- presenting an assessment item as independent work when it has been produced in whole or part in collusion with other people, for example, another student or a tutor; and
- claiming credit for a proportion a work contributed to a group assessment item that is greater than that actually contributed.

For the purposes of this policy, submitting an assessment item that has already been submitted for academic credit elsewhere may be considered plagiarism. Knowingly permitting your work to be copied by another student may also be considered to be plagiarism.

Note that an assessment item produced in oral, not written, form, or involving live presentation, may similarly contain plagiarised material.

The inclusion of the thoughts or work of another with attribution appropriate to the academic discipline does not amount to plagiarism.

The Learning Centre website is main repository for resources for staff and students on plagiarism and academic honesty. These resources can be located via: www.lc.unsw.edu.au/plagiarism

The Learning Centre also provides substantial educational written materials, workshops, and tutorials to aid students, for example, in:

- correct referencing practices;
- paraphrasing, summarising, essay writing, and time management;
- appropriate use of, and attribution for, a range of materials including text, images, formulae and concepts.

Individual assistance is available on request from The Learning Centre. Students are also reminded that careful time management is an important part of study and one of the identified causes of plagiarism is poor time management. Students should allow sufficient time for research, drafting, and the proper referencing of sources in preparing all assessment items.

All work submitted for assessment must be your own work. Lab exercises and assignments must be completed *individually*. We regard copying of assignments or lab exercises, in whole or part, as a very serious offence. We use plagiarism detection software to search for multiply-submitted work.

1. Submitting part or all of other students' work, with or without acknowledgement, is not acceptable.
2. Submitting solutions written by other persons is also not acceptable.
3. Building on ideas and partial solutions obtained from public sources, such as web resources, may be acceptable, provided full acknowledgement is made. However, the final mark will take into account the starting point and how much development work would have been required. Failing to acknowledge web or other resources is unacceptable.
4. Discussing approaches to solutions with other students is quite appropriate, but any discussions should remain at the design level, and must not include program text. Comparison tools will detect any common code across the student body.
5. The safest approach is to work diligently on your own, seeking help from the forum or course staff.

- Submission of work derived from another person, or jointly written with someone else will, may result in automatic failure for COMP2041/COMP9041 with a mark of zero.
- Allowing another student to copy from you will may result in a mark of zero for your own assignment or lab exercises. Do not provide your work to any other person, even people who not UNSW students. You will be held responsible for the actions of anyone you provide your work to.
- Severe or second offences will result in automatic failure, exclusion from the university, and possibly other academic discipline.

Resources for Students

Information for students with disabilities Talk to the lecturer if you have any disabilities that may affect this course.
There is no required textbook for the course. Useful reference books include the following:

- Kernighan & Pike,
  The Practice of Programming,
  Addison-Wesley, 1998.
  (Inspiration for 2041 - philosophy and some tool details)
- McConnell,
  Code Complete (2ed),
  Microsoft Press, 2004.
  (Many interesting case studies and practical ideas)
- Wall, Christiansen & Orwant,
  Programming Perl (3ed),
  O'Reilly, 2000. (Original & best Perl reference manual)
- Schwartz, Phoenix & Foy,

Learning Perl (5ed),
O'Reilly, 2008. (gentle & careful introduction to Perl)

- Christiansen & Torkington,
  Perl Cookbook (2ed),
  O'Reilly, 2003. (Lots and lots of interesting Perl examples)
- Schwartz & Phoenix,
  Learning Perl Objects, References, and Modules (2ed),
  O'Reilly, 2003. (gentle & careful introduction to parts of Perl mostly not covered in this course)
- Schwartz, Phoenix & Foy,
  Intermediate Perl (2ed),
  O'Reilly, 2008. (good book to read after 2041 - starts where this course finishes)
- Sebesta,
  A Little Book on Perl,
  Prentice Hall, 1999. (Modern, concise introduction to Perl)
- Orwant, Hietaniemi, MacDonald,
  Mastering Algorithms with Perl,
  O'Reilly, 1999. (Algorithms and data structures via Perl)
- Kochgan & Wood 2003,
  Unix® Shell Programming,
  Sams Publishing 2003 (Careful intoduction to Shell Programming)
- Peek, O'Reilly, Loukides,
  Bash Cookbook,
  O'Reilly, 2007. (Recipe(example) based intro to Shell programming)
- Powers, Peek, O'Reilly, Loukides,
  Unix Power Tools (3ed),
  O'Reilly, 2003. (Comprehensive guide to common Unix tools)
- Loukides & Oram,
  Programming with GNU Software,
  O'Reilly, 1997. (Tutorial on the GNU programming tools (gcc,gdb,...))
- Robbins,
  Unix in a Nutshell (4ed),
  O'Reilly, 2006. (Concise guide to Unix and its toolset)
- Kernighan & Pike,
  The Unix Programming Environment,
  Prentice Hall, 1984. (Pre-cursor to the textbook, intro to Unix tools)

For pointers to other useful reading material, including documentation for all of the tools used in the practical work, see the course Web pages.

Course evaluation and development

Student feedback on this course will be obtained via electronic survey at the end of session, and will be used to make continual improvements to the course. Students are also encouraged to provide informal feedback during the session, and to let the lecturer in charge know of any problems, as soon as they arise. Suggestions will be listened to very openly, positively, constructively and thankfully, and every reasonable effort will be made to address them.
This feedback is used to improve the course materials & their delivery. In the most recent session feedback was very favourable probably as results of changes based on previous session's feedback. Some lab exercises and lecture topics will be updated to better reflect current practice.

COMP(2041|9044) 19T2: Software Construction is brought to you by
the School of Computer Science and Engineering at the University of New South Wales, Sydney.
For all enquiries, please email the class account at cs2041@cse.unsw.edu.au
CRICOS Provider 00098G