COMP(2041|9044) - Software Construction 20T2

- Convenor/Lecturer: Andrew Taylor
- Course Admin: Jashank Jeremy
- Tutors:

  Samuel O'Brien
  Anson Lee
  Matthew Turner
  Sage Barreda-Pitcairn
  Tom Nguyen
  Nathan Ellis
  Vivian Dang
  Maxwell Goodbury
  Luka Kerr
  Dylan Brotherston
  Asher Silvers
  Coen Townson
  Sabrina Yan
  James Treloar
  Sabine Lim
  Esther Wong
  Connor O'Shea
  Bridget McCarthy

Course Goals

Overview: to expand your knowledge of programming.
First programming courses deals with ...

- one language (C or Python at CSE)
- some aspects of programming (e.g. basics, correctness)
- on small tightly-specified examples

COMP(2041|9044) deals with ...

- other languages (Shell, Perl)
- other aspects of programming (e.g. testing, performance)
- on larger (less-small) less-specified examples

Course Goals

Introduce you to:

- building software systems from components
- treating software as an object of experimental study

Develop skills in:

- using software development tools (e.g. git)
- building reliable, efficient, maintainable, portable software

Ultimately: get you to the point where you could build some software, put it on github, have people use it and have it rated well.

Inputs

At the start of this course you should be able to:

- produce a correct procedural program from a spec
- understand fundamental data structures + algorithms
  (char, int, float, array, struct, pointers, sorting, searching)
- appreciate the use of abstraction in computing

Outputs

At the end of this course you should be able to:

- understand the capabilities of many programming tools
- choose an appropriate tool to solve a given problem
- apply that tool to develop a software solution
- use appropriate tools to assist in the development
- show that your solution is reliable, efficient, portable

Syllabus Overview

1. Qualities of software systems
   - Correctness, clarity, reliability, efficiency, portability, ...
2. Techniques for software construction
   - Analysis, design, coding, testing, debugging, tuning
   - Interface design, documentation, configuration
3. Tools for software construction
   - Filters (*grep, cut, sort, uniq, tr, sed*...)
   - Scripting languages (*shell*, *Perl*, *Python*)
   - Intro to Programming for the web
   - Software tools (git, ...)

Lectures

- Tuesday 14:00 - 16:00
- Friday 10:00 - 12:00
- lectures will be delivered via Teams Live Event
- you will have email about how to access Teams Teams Live Event
- feel free to ask questions via chat
- present a brief overview of theory
- present a brief overview of theory
- focus on practical demonstrations of coding
- demonstrate problem-solving (testing, debugging)
- Lecture slides available on the web before lecture.
- Lectures recorded and linked to home page.

Tutorials

Tutorials aim to:

- tutorials start in week 1.
- tut/lab online (Blackboard Collaborate)
- you will have email about how to access Collaborate
- tutes clarify lecture material
- work through problems related to lecture topics
- give practice with design *(think before coding)*
- answers available on the web after the week's last tutorial.

To get the best out of tutorials

- attempt the problems yourself beforehand
- ask if you don't understand a question or how to solve it
- Do *not* keep quiet in tutorials ... talk, discuss, ...
- Your tutor may ask for your attempt to start a discussion.

Lab Classes

Each tutorial is followed by a two-hour lab class.
Lab exercises aim to build skills that will help you to

- complete the assignments
- pass the final exam

Lab classes give you experience applying tools/techniques.
Each lab exercise is a small implementation/analysis task.
Labs often includes challenge exercise(s)
Do them yourself!

Lab Classes

Each tutorial is followed by a two-hour lab class.

- Lab exercises mostly small coding tasks.
- Lab exercise build skills needed for assignments & exam.
- Lab exercises done individually
- submitted via `give` before Tuesday 12:00
- Automarked (with partial marks) 15% of final mark.
- Labs may include challenge exercises.
- Challenge exercises may be silly, confusing, or impossibly difficult.
- Full marks possible without completing any challenge exercises

Weekly Tests

- programming tests
- immediate reality-check on your progress.
- done in your own time under self-enforced exam conditions.
- time limit of 1 hour
- automarked (with partial marks)
- contribute 10% of final mark.
- best 6 of 8 tests used to calculate the 10%
- any violation of the test conditions $\Rightarrow$ zero for whole component

Assignments

Assignments give you experience applying tools/techniques
(but to larger programming problems than the lab exercises)
Assignments will be carried out individually.
They always take longer than you expect.
Don't leave them to the last minute.
There are late penalties applied to maximum assignment marks, typically:

- 2%/hour

Organising your time $\Rightarrow$ no penalty.

Code of Conduct

CSE offers an inclusive learning environment for all students.
In anything connected to UNSW, including social media,
these things are student misconduct and will not be tolerated:

- racist/sexist/offensive language or images
- sexually inappropriate behaviour
- bullying, harassing or aggressive behaviour
- invasion of privacy

Show respect to your fellow students and the course staff

Plagiarism

What is plagiarism?

Presenting the (thoughts or) work of another as your own.

Cheating of any kind constitutes academic misconduct and carries a range of penalties. Please read course intro for details.

Examples of inappropriate conduct:

- groupwork on individual assignments (discussion OK)
- allowing another student to copy your work
- getting your hacker cousin to code for you
- purchasing a solution to the assignment

Plagiarism

What is plagiarism?

Presenting the (thoughts or) work of another as your own.

Cheating of any kind constitutes academic misconduct and carries a range of penalties. Please read course intro for details.

Examples of inappropriate conduct:

- groupwork on individual assignments (discussion OK)
- allowing another student to copy your work
- getting your hacker cousin to code for you
- purchasing a solution to the assignment

Remember

You are only cheating yourself and chances are you will get caught!

Plagiarism

- Labs, tests, assignments must be entirely your own work.
- You can not work on assignment as a pair (or group).
- Plagiarism will be checked for and *penalized*.
- Plagiarism may result in suspension from UNSW .
- Scholarship students may lose scholarship.
- International students may lose visa.
- Supplying your work to any another person may result in loss of all your marks for the lab/assignment.

Final Exam

- Closed-book practival exam during the exam period.
- The exam contributes 45% to total mark for the course.
- Some multiple-choice/short answer questions - similar to tut questions.
- Six (probably) implementation questions - similar to lab exercises.
- COMP(2041|9044) hurdle requirement:
- you must score 18+/45 on the final exam.
- You can not pass unless you meet the hurdle.
- If you fail to meet hurdle and get 50+ overall, you will receive a grade of UF.

Final Exam

- online practical exam (you completefrom home)
- limited on-line language documentation available
- some multiple-choice/short answer questions - similar to tut questions.
- some questions will ask you to read shell, Perl, regex, ...
- most marks for questions which ask you to write shell or Perl
- also may ask you to answer written questions
- you must score 18+/45 on the final exam to pass course

How to Pass this Course

- coding is a *skill* that improves with practice
- the more you practise, the easier you will find assignments/exams
- do the lab exercises
- do the assignments *yourself*
- practise programming outside classes
- treat extra tutorial questions like a mini prac exam

Assessment

- 15% Labs
- 10% Weekly Programming Tests
- 15% Assignment 1 — due week 7
- 15% Assignment 2 — due week 10
- 45% Final Exam

Above marks may be scaled to ensure an appropriate distribution
To pass you must:

- **score 50/100 overall**
- **score 18/45 on final exam**

For example:
55/100 overall and 20/55 on final exam $\Rightarrow$ **55 UF** not 55 PS

Reading Material

General References:

- *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)

Reading Material

Perl Reference Books:

- *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)

Reading Material

Shell Programming:

- *Kochgan & Wood 2003*, UnixÂ® 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)

Reading Material

Unix Tools Reference Books:

- *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)

Reading Material

All tools in the course have extensive on-line documentation.
Links to this material are available in the course Web pages.
You are expected to master these systems largely by reading the manuals.
However ...

- we will also give introductory lectures on them
- the lab exercises will give practice in using them

Note: "The ability to read software manuals is an invaluable skill" (jas,1999)

Home Computing

All of the tools in the course are available under Unix and Linux.
Most have also been ported to MS Windows.
(generally via the CygWin project)
All tools should be available on Mac.
(given that Mac OS X is based on FreeBSD Unix)
Links to downloads will be placed on course Web site.

Home Computing

There may be minor incompatibilities between Unix, Windows and Mac versions of tools.

Therefore ... *test your assignments at CSE* before you submit them.

Note: we expect that any software that *you* produce will be portable to all platforms.

This is accomplished by adhering to *standards*.

Conclusion

The goal is for you to become a better programmer

- more confident in your own ability
- producing a better end-product
- ultimately, enjoying the programming process