Course Outline

COMP1511 19T1

Course Details

Course Codes	COMP1511
Course Title	Programming Fundamentals
Units of Credit	6
Course Website	http://cse.unsw.edu.au/~cs1511
Handbook	COMP1511
Lecturer In Charge/Course Convener	Andrew Taylor `andrewt@unsw.edu.au`
Lecturers	Andrew Taylor `andrewt@unsw.edu.au` (Stream A) Marc Chee `marcchee@cse.unsw.edu.au` (Stream B)
Admin	Mei Cheng Whale `meicheng@unsw.edu.au`
Lectures	Tuesday 09:00-11:00: Stream A: Central Lecture Block 7 Stream B: Mathews Theatre A
	Thursday 11:00-13:00: Stream A: Central Lecture Block 7 Stream B: Mathews 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.

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 and test the forwarding.

Additional information will be provided in the Course Forum. You should check the course forum 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.

Prerequisites/Background Knowledge

COMP1511 has no prerequisites, and assumes no background knowledge.

Course Summary

This course introduces students to the basics of programming. Topics covered include:

- fundamental programming concepts
- the C programming language and use of a C compiler
- programming style
- program design and organisation concepts
- program testing and debugging

Course Aims

The course aims for students to become proficient in programming using a high level language, C. By the end of the course, students should be able to construct C programs to solve problems.

Course Learning Outcomes

- understand the core syntax and semantics of the C programming language including types, I/O, arrays, functions, pointers, structs, file manipulation and dynamic memory allocation
- given a problem, solve it by proficiently constructing (designing, testing, debugging) a secure, reliable and correct C program
- understand and employ fundamental data structures including stacks, queues, linked lists and binary trees
- use Linux and Unix-like operating systems to develop and test software

COMP1511 versus COMP1911

COMP1511 is designed for computer science majors, and for any student with a keen interest in computing, regardless of their degree program.

COMP1911 is for students who are not computer science majors. COMP1911 covers a subset of the material in COMP1511 and moves at a more gentle pace.

If you are a computer science major, you must take COMP1511

If you are not a computer science major, but have an interest in computer science and think you may take further COMP courses, you also should enrol in COMP1511

And if you have previous programming experience - **and enjoyed it** - choose COMP1511

Course Schedule

Week 1	Course Introduction, Introduction to Linux & Shell, Compiling and Running C Programs, Data Types, Variables, Simple I/O with print and scanf, Arithmetic expressions, If statements
Week 2	Nested-If statements, Boolean Expressions,
Week 3	Loops, Nested Loops
Week 4	Arrays,
Week 5	Characters and Strings, Command Line Arguments, Multi-file Programs
Week 6	Functions, Variable Scope and lifetime
Week 7	Pointers, Dynamic memory allocation (malloc) **Assignment 1** due
Week 8	Introduction to version control, Structs, Linked Lists
Week 9	Stacks and Queues, Abstract data types & Interfaces,
Week 10	**Assignment 2** due C implementation memory models, Invalid C and Security (buffer overflow and other issues), Professionalism, Codes of Conduct, Ethics for Programmers Revision, Exam preparation
Exam Period	**Final Exam**

Topics including development approaches, programming style, testing, debugging strategies and ethics will be discussed though the course as they arise.

Teaching Rationale

This course has a heavy practical orientation. Lectures will revolve around live demonstrations of programming and use of tools. Labs and assignments form a key part.

Teaching Strategies

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.

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.

Each week, there will be several exercises to work on. These exercises will be released in the week preceding the lab class.

Most lab exercises will be done in pairs, and you and you partner should discuss the exercises before the lab to maximise the usefulness of the class.

Tutors will facilitate you forming pairs in your week 1 lab (which is not assessed). The pairs will change twice during session.

Starting week 3, pairs will also be asked to do code reviews in the tutorials, to explain how they tackled a particular problem and describe interesting features of their solution.

Overview

Lab exercises will be automarked (marked automatically by a computer), so that tutors can spend lab time assisting students rather than marking labs.

Submission

For each of the lab exercises, **both** members of each lab pair need to submit the exercise **separately** using `give` .

You cannot obtain marks by e-mailing lab work to tutors or lecturers.

If you cannot complete the exercise by the end of the lab you may complete it in your own time and submit it using the `give` command **before midnight Sunday** (Sunday 11:59:59pm).

Challenge Exercises

Challenge exercises may be specified for some labs.

Some challenge lab exercises typically will specify that they are individual exercises (not to be done with your partner).

Challenge exercises may be silly, confusing or unreasonably difficult.

Do not worry if you can not complete challenge exercises.

Lab Marking

Lab exercises will be automarked, using test cases that you haven't seen: different to the test cases `autotest` runs for you. (Hint: do your own testing as well as running `autotest`)

There will be partial marks for attempts which fail some of these automated tests.

Automarking

Automarking will be run several days after the submission deadline for the lab. When it is complete you should be able to view it here or by running this command on a CSE machine:

```
$ 1511 classrun -collect exercise_name
```

Lab Marks

When all components of a lab are automarked you should be able to view the the marks via give's web interface or by running this command on a CSE machine:

```
$ 1511 classrun -sturec
```

There will be more lab marks available than necessary to obtain full marks for the 13% lab component. In other words: total lab marks will be capped.

The lab exercises for week are worth in total 2 marks.

Except there are no marks for the week 1 lab.

Usually each lab exercise will be worth the same - for example if there are 5 lab exercises each will be worth 0.4 marks.

Except challenge exercises (see below) will never total more than 20% of each week's lab mark.

The best 8 of your 9 lab marks for weeks 2-10, will be summed to give you a mark out of 13.

If their sum exceeds 13 - your total mark will be capped at 13.

Hence:

- You can miss 1 lab without affecting your mark.
- You can obtain full marks for the lab component without doing any challenge exercises.

Weekly Coding Tests

There will be 8 weekly coding 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 (probably 1 hour).
3. No access to materials (written or online) except specified language documentation or man pages.

Each coding test will be automatically marked. There will be partial marks for attempts which do not pass automatic tests. Here is an indicative guide:

Passes all automatic tests.	1/1
Fails several automatic tests.	0.75/1
otherwise	0.5/1

Your mark for the coding test component will be the sum of your best 7 of 8 test marks.

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

The weekly programming test must be completed by Sunday Midnight.

Assignments

There are two assessable programming assignments. Assignments give you the chance to practice what you have learned 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.

- Assignment 1 (Submission, Week 7) 13%
- Assignment 2 (Submission, Week 10) 13%

The assignment weighting and deadlines may change a little when the assignment designs are complete.

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

Final Exam

There will be a three-hour primarily practical exam, to be held in the CSE labs during the exam period. This will be centrally timetabled and appear in your UNSW exam timetable.

It will contain implementation tasks which will require you to write C programs. It will also contain sections which require you to read code or answer questions.

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

Exam Hurdle Requirements

COMP1511 has two hurdle requirements on the final exam.

Hurdle Requirement #1: on the final exam you must solve a task by writing a program that uses an **array**. The final exam will contain multiple questions (clearly marked) which if answered successfully will meet this hurdle requirement. Answering any one of these questions will meet this hurdle requirements.

Hurdle Requirement #2: on the final exam you must solve a task by writing a program that uses a **linked list**. The final exam will contain multiple questions (clearly marked) which if answered successfully meet this hurdle requirement. Answering any one of these questions will meet this hurdle requirements.

You can not pass COMP1511 unless you achieve both the above hurdles. However you will be offered an additional chance to pass the hurdles in the supplementary exam, if you achieve a mark of 50+ but do not pass both hurdles.

Assessment

Component	Weight
Lab Work	13%
Weekly Tests	7%
Assignments	26%
Final Exam (exam period)	54%

Supplementary Assessment

Students will be offered a supplementary exam if they miss the original exam due to (documented) illness or misadventure.

Students who will be automatically offered supplementary assessment if they achieve a final mark of 50+ but fail to meet the hurdle requirement, if they have attended 9+ tut-labs, 8+ programming tests and have made reasonable attempts on all assignments (achieving > 45%)

Students with final marks in the range 40-49 (whether they have met the hurdle requirement or not) will also be offered supplementary assessment if they have attended 9+ labs, 8+ programming tests and have made reasonable attempts on all assignments (achieving > 45%)

The supplementary exam is I scheduled for the week 27/05-01/06. It is your responsibility to be in Sydney and available for the supplementary exam. No alternative will be offered.

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.

All work submitted for assessment must be your own work.

Lab exercises must be completed by you and your partner.

Assignments must be completed *individually*.

Submission of other people's work as your own (plagiarism) has a major impact on learning so we use plagiarism detection software to search for multiply-submitted work.

Please note:

- Submitting part or all of other students' work, with or without acknowledgement, is not acceptable.
- Submitting work written even partly by another person is also not acceptable.
- Building on ideas and code snippets obtained from public sources, e.g. Stack Overflow, is acceptable (unless the assignment/lab forbids this) provided full acknowledgement of their resources is made.
- Discussing approaches to exercises and assignments 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.
- 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 result in loss of marks and your name being entered on UNSW's plagiarism register. This loss of marks often results in the student failing the course.
- Providing your work to another person will also result in the loss of marks and your name being entered on UNSW's plagiarism register. Do not provide your work to any other person, even people who are not UNSW students. You will be held responsible for the actions of anyone you provide your work to.
- Severe or second offences may result in automatic failure or exclusion from UNSW.

Make sure you read:

- the Learning Centre's explanation of plagiarism.
- UNSW's policy regarding academic honesty and plagiarism

Other matters

- Occupational Health and Safety policies
- Information for students with disabilities

  Contact the lecturer ASAP if you have any disabilities that may affect this course.

Course Evaluation and Development

Every semester, COMP1511 student feedback is requested in a survey at the end of this course using UNSW's myExperience online survey system.
This feedback is used to improve the course materials and their delivery.

In the most recent session feedback was very favourable probably as a result of changes based on the previous session's feedback. Feedback from surveys for COMP1511's previous offerings has resulted in changes to COMP1511 delivery, including introduction of weekly programming tests. Some lab exercises and lecture topics will be updated to better reflect current practice.

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.

Resources for Students

The optional textbook for the course is: Programming, Problem Solving, and Abstraction with C by Alistair Moffat,

COMP1511 19T1: Programming Fundamentals 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 cs1511@cse.unsw.edu.au

CRICOS Provider 00098G