COMP1911 - Computing 1A - Course Outline

Course Details

Course Code	COMP1911
Course Title	Computing 1A
Units of Credit	6
Course Website	http://cse.unsw.edu.au/~cs1911 (http://cse.unsw.edu.au/~cs1911)
Handbook Entry	http://www.handbook.unsw.edu.au/undergraduate/courses/current/COMP1911.html (http://www.handbook.unsw.edu.au/undergraduate/courses/current/COMP1911.html)
Lecturer/Convener/Admin	Hayden Smith `hsmith@cse.unsw.edu.au`
Lectures	Tuesday (weeks 1-10) 9:00-11:00 Science Theatre (K-F13-G09) Tuesday (weeks 11) 11:00-13:00 Science Theatre (K-F13-G09) Thursday (weeks 1-9) 11:00-13:00 Science Theatre (K-F13-G09)
Complete Timetable	Course Timetable with Lectures, Tutorials,Labs (https://webcms3.cse.unsw.edu.au/COMP1911/19T1/timetable) and Consultations and Help Labs (https://webcms3.cse.unsw.edu.au/COMP1911/19T1/resources/22145)

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. Its essential your 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 Forums (https://webcms3.cse.unsw.edu.au/COMP1911/19T1/forums/) You should read these regularly. These forums are the best place to ask questions about the course.

Prerequisites/Background Knowledge

COMP1911 has no prerequisites, and assumes no background knowledge.
No previous programming experience is assumed. If you have previous programming experience you should consider taking COMP1511 instead.

COMP1911 versus COMP1511

COMP1911 is for students who are not computer science majors.

- If you are a computer science major, you MUST be enrolled in COMP1511 instead.
- If you not a computer science major, have a interest in computer science and think you may wish to take more COMP courses, you also should enrol in COMP1511 instead.
- And if you have previous programming experience - **and enjoyed it** - choose COMP1511

COMP1911 and COMP1511 are both pre-requisites for COMP1521. However if you do COMP1911 instead of COMP1511 and then you decide you want to do even more COMP subjects as well you will need to do a bridging course.

A bridging course will be offered to 19T1 COMP1911 students in the midyear break. It will cover the material in COMP1511 which is not covered in COMP1911. Satisfactory performance in the exam at the finish of this bridging course will allow students to enrol in courses which have have COMP1511 as a prerequisite.

Course Summary

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

- fundamental programming concepts
- the C programming language & 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 its completion that should be able to construct C programs to solve problems.

Student Learning Outcomes

- LO1: Understand fundamental procedural programming concepts including basic data representation.
- LO2: Design, implement, test and debug complete C programs to solve specified problems.
- LO3: Understand good practice for program organsation and programming style.

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	tutorials, labs and assignments
Professionals capable of ethical, self- directed practice and independent lifelong learning	labs,assignments
Global citizens who are culturally adept and capable of respecting diversity and acting in a socially just and responsible way	lectures, tutorials, labs

Teaching Rationale

This course has a heavy practical orientation. Lectures will revolve around live demonstrations of programming and use of tools. Labs & 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. Lecture recordings will be available online a short time after the actual lectures.

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 tute, by asking questions, by offering suggestions and by generally participating. The tutorial questions will be posted on the Web in the week before each tute. There are no marks for tutorial attendance.

Laboratory Classes

Following the tute class each week, there will be a two-hour lab 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, lab class are a very important component, and you should make every effort to attend the labs and complete the exercises diligently. In particular, **keep up-to-date** with the Lab work; if you fall behind it affects your ability to understand later material in the course.
To obtain a mark for a lab exercise you must both demonstrate the completed lab exercise to your tutor during a lab class and submit it using give.

If you don't complete a lab exercise during the scheduled class, you can still obtain the mark if you both submit the completed exercises before midnight Sunday and you demonstrate it to you tutor in the follow week's lab.

Lab exercises will be assessed using the following grade system:

Grade	Mark	Criteria
A+	1.25	Complete, correct solution with perfect style
A	1	Competent solution to core part of lab - any problems or bugs must be minor
B	0.5	Incomplete solution, or complete with significant defects or obvious bugs
C	0.25	Partial solution only, minimal achievement; or inability to explain solution
D or .	0	Not attempted

Challenge exercises are optional to attempt/complete. Completing challenge exercises can help boost your lab mark. For example, if you get an A for all non-challenge exercises (because you made a couple of mistakes), then if you also complete the challenge exercise that can be your path to an A+ instead of further improving the other non-challenge exercises.

Your lab marks (there will be 9 labs in total) will be summed and capped at 8 marks. These 8 marks will contribute 10% of your overall course mark as per the assessment section below.

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.

- Assignment 1 (Submission, Week 5) 15%
- Assignment 2 (Submission, Week 9) 15%

The deadlines may be slightly varied when the assignment designs are complete and to fit best with the lecture schedule.

Late programming assignments 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.

Exam

There will be a three-hour exam, to be held in the CSE labs during the exam period. It consists of small implementation tasks which will require you to write C programs, and some theory questions. All questions will be answered and submitted on a computer. There is no handwritten component.
During this exam you will be able to execute, debug and test your programs. The implementation tasks will be similar to those encountered in lab exercises

It is a hurdle requirement for this course that you perform satisfactorily on the implementation tasks in the exam. Satisfactory performance is defined as solving two of the small implementation tasks.

The exact exam format (exam skeleton) will be released in week 13 but it is likely there will be 4 implementation tasks.

Assessment

Component	Value	Contributes to Learning Outcomes
Lab Work	10%	LO1,LO2,LO3
Assignments	30%	LO1,LO2,LO3
Exam	60%	LO1

As mentioned above, your performance on the practical component of the final exam must also be satisfactory to pass the course.

The lecturer may scale overall marks, or individual components, up or down to obtain a desired mark distribution (don't panic - scaling marks down is very rare).

Supplementary Exams

The document "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 held on the date specified in the above document. 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 School Office for details of Supplementary examinations.

Content chedule

The following content schedule is a draft only and is subject to change throughout the session.
Week 1 Introduction, C Basics
Week 2 Numeric Types and C conditions
Week 3 Loops & Functions
Week 4 Arrays
Week 5 Characters and Strings
Week 6 Pointers and memory
Week 7 Structs
Week 8 Stacks and queues
Week 10 Bonus / Extension content
Week 11 Exam Revision

Course evaluation and development

Every semester, COMP1911 Student feedback is requested in a survey at the end of this course using UNSW's CATEI online survey system.
This feedback is used to improve the course materials & their delivery.

Feedback from the last year was very positive. However it was suggested to provide more and better access to sample code and better practice material for the exam.

Resources for Students

Information for students with disabilities (http://www.studentequity.unsw.edu.au/) Talk to the lecturer if you have any disabilities that may affect this course.

The optional textbook for the course is: Alistair Moffat, Programming, Problem Solving, and Abstraction with C (http://www.bookshop.unsw.edu.au/details.cgi?ITEMNO=9781486010974),
Pearson Education, 2012.

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

Academic Honesty and Plagiarism

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)
- 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)
- Student Misconduct Procedure (https://www.gs.unsw.edu.au/policy/documents/studentmisconductprocedures.pdf)

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.