1927 Computing 2 (Summer Session, 2011)

Course Description

Prerequisites

The official pre-requisite for this course is either COMP1917 or COMP1921. We expect that you can program the C language. In particular, you should to be able to design, implement, debug, test and document small C programs (up to several hundred lines of code). You also need to be familiar with the Linux environment on CSE computers.

Aims

During this course we will concentrate on introducing some of the fundamental data structures and algorithms that are essential to understanding and developing modern software. We will focus not only on their design, but methods for testing and analysing them. Our analysis will enable us to compare alternative data structures and algorithms for particular problem cases. We will cover additional aspects of the C language that were not covered in the pre-requisite course, however, COMP1927 is not intended to be simply a second C programming course. The focus is on the ideas behind the data structures and algorithms that we'll look at. The C language is simply a vehicle to express our ideas. If you're not confident in C, we strongly recommend you revise COMP1917 material before the start of semester.

Learning Outcomes

By the end of the course, you should:

- be familiar with fundamental data structures and algorithms
- be able to analyse the performance characteristics of algorithms
- be able to package a set of data structures and algorithms as an abstract data type
- 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

Course Details

- Course Code/Title: COMP1927 Computing 2
- Units of Credit: 6
- Course Website: http://www.cse.unsw.edu.au/~cs1927
- Handbook Entry: http://www.handbook.unsw.edu.au/undergraduate/courses/2010/COMP1927.html

You should become familiar with the structure and content of the course website early on, and consult them frequently. All important announcements will be posted to the Noticeboard on the course website. Urgent announcements will also be emailed to all members of the class.

Permanent Staff

Name	Room	Phone	Email	Role
John Plaice	K17 Room 201E	9385 6012	cs1927	Lecturer

The names and contact details of the tutors are given on the course website.

Material Covered

A list of topics and references to the course textbooks will be given on the course web site. The course textbooks are:

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

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

Teaching Strategies

COMP1927 includes lectures, tutorials, labs, quiz, assignment and the textbooks.

Lectures aim to convey basic information about the course content and to model the practices and techniques involved in software development. The most important components of the course, however, are the tutorials, labs and assignment. Tutorials aim to clarify and refine the knowledge that you got from lectures and from reading the textbook. Labs and the assignment are where you get to put together and practise all of the ideas from the lectures, tutorials and text. The only way to develop the skills to do effective software development is by practising them, and to test them using a quiz. The labs and assignment are designed specifically to help you learn, practice understand the most important material for the course. If you do not make enough effort with the assignment and lab exercises, you're wasting the course's most valuable learning opportunities.

Lectures

Each week there will be up to 6 hours of lectures during which theory, practical demonstrations and case-studies will be presented. The lectures will be held in Weeks 1-3, 5-7. There will be no lectures in Week 4.

Day	Time	Location
Wednesday	09:00 - 12:00	Civil Engineering G1
Friday	09:00 - 12:00	Civil Engineering G1

Lectures convey a part of the course content, and are there to stimulate thought and support actual practical behind-the-computer learning. Passive learning (i.e. just listening) will not result in improved programming skills. You must be actively learning throughout the session by programming. Feel free to ask questions at any stage, but otherwise please respect the right of other people around you who are trying to listen and keep quiet.

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. You will gain more from these classes if you read through and think about the questions in advance. The aim of the class is for you to focus on material that is essential for understanding. You need to be active and ask questions in tutorials to get the most out of them.

There may not be enough time in each tutorial to finish every question. If you do not finish, you should complete the rest on your own.

Consultation

Consultation is right after the Wednesday lecture: John Plaice, 12:00--13:00.

Lab Classes (assessed)

Lab classes aim to give you practice in problem-solving, program development and analysis. Each week, there will be a number of small exercises to work on. The exercises should be submitted by the deadline to be assessed. They must compile without warning or error and run successfully against test data to be considered correct. There will be no lab in Week 1. The lab in Week 7 will be for the quiz. The lab in Week 8 will be for review.

Assignment (assessed)

In the assignment, you will work on more substantial programming exercises or analytical tasks. The assignment should be completed individually; this means that you should do them yourself without assistance from others, except for asking advice from the teaching staff (lecturers and tutors) in this course. As noted above, they 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, and you take the risk of being caught for plagiarism. If you are caught, you will be awarded an overall mark of zero for the assignment, and possibly the course. This does not look good on your academic record.

However, you should be aware that re-use of algorithms and data structures developed outside this course is allowed. You should not re-use code however, but if you feel it is justified, you should give credit to the original author or source.

Plagiarism is defined as "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:

- Learning Centre: Plagiarism and Academic Integrity
- MyUNSW: Plagiarism and Academic Misconduct
- CSE: Addendum to UNSW Plagiarism Guidelines
- CSE: Yellow Form (whose terms you have agreed to)

Make sure that you read and understand these. Ignorance is not accepted as an excuse for plagiarism, especially after completing at least your first semester of study.

Late Submission: An assignment that is submitted late will attract a penalty of approximately 10% off the ceiling mark for each day (or part thereof) that it is late (e.g. after 2 days, the maximum mark you can score for the assignment is 80%). If you have Special Consideration-type grounds for submitting an assignment late, provide me with the evidence before the assignment deadline. If the evidence is convincing, you will be allowed to submit late without penalty. Work commitments will not be accepted as an excuse for handing an assignment in late.

Assessment

Details of the assessment scheme are still being worked on, so some things below may change.

Lab mark

Submitting your lab solutions for assessment requires you to use the give command shown at the end of every exercise. Each lab solution that you submit will be awarded a number of labmarks between 0 and 2, as follows:

- If you give code that does not compile, that receives 0 labmarks.
- If code compiles without warnings, runs successfully, and does not violate the style guide, then it will receive 2 labmarks.
- Otherwise it will receive a mark between these two extremes that depends on the severity of the problem.

There are 2 conditions:

- There is a strict time limit: you must give your code by midnight on the Sunday after the lab.
- There will be no appeal to labmarks awarded (because exercises considered individually are worth only a very small fraction of your final mark). So if you make a silly programming mistake, learn from your mistake and move on.

The student(s) that receive the maximum number of labmarks over the whole course will have their labmark scaled to 10, which forms part of their final mark. All other values of labmark will be scaled proportionally (so if you have received half that number of labmarks, you get 5 marks).

Assignment mark

The assignment will be worth 15 marks.

Quiz

There is one quiz in week 7. It is purely practical (i.e. there is no written component) and closed book, and held during your laboratory. You will be asked to program (from scratch) and possibly test some basic algorithm. The rules for attendance are the same as above. You will not be expected to remember all of C's syntax: concise reference information will be provided.

The rules for non-attendance of this quiz is the same as for the final examination (see below).

Final Examination

There will be a 3-hour closed-book final examination for this course. This examination will consist of a mixture of multiple-choice questions, short answer questions and small programming tasks and analysis exercises.

Attendance at the examination will be treated as an indication by you that you are feeling well enough to sit that examination. If you are feeling so unwell on the day of the examination that you cannot attend, go to the doctor and get a medical certificate. If you put in a request for supplementary assessment after already attending the original examination, your request will not be granted. Also, if your special consideration request does not provide documented evidence that you personally have been severely affected by some unforeseen situation, then your request will not be granted.

The supplementary examination may be a written or an oral examination. Note that a supplementary examination is typically more difficult than the original examination to reflect the fact that you have had more preparation time.

Final Assessment

Your final mark in this course will be based on components from the labs, the assignment, the quiz and the exam. The final examination has a 45%-hurdle, which means that if you score less than 45% in the final examination, then that is your final mark, and you will fail the course. Final marks may be scaled to ensure that the course Pass/Fail boundary and the Distinction/High Distinction boundary reflect a consistent standard from session to session.



Labs	10 marks
Assignment	15 marks
Quiz	20 marks
Final exam	60 marks

The total number of marks is hence 105, but final marks will be capped to 100.

Course Evaluation

This course is being continuously improved and we will conduct a survey at the end of session to obtain feedback on the quality of the various course components. Your participation in the survey will be greatly appreciated.

The course will focus more on testing programming skills (hence the practical quiz as well as a final exam) than in previous sessions. Every effort will also be made to get out the lecture notes, and tutorial and lab exercises very early. Assessment load has been reduced with just one assignment, and making labs optional.

Acknowledgments

Slides, notes, and other material are partially from previous lecturers of this course, including Angela Margaret Finlayson, John Shepherd, Manuel Chakravarty, Richard Buckland, Maurice Pagnucco and Albert Nijmeyer. Some slides and programs are adapted from the teaching material provided with Robert Sedgewick's books.