Course Outline

COMP1927
Computing 2

Teaching Period 1 2011

Contents

- Course Staff
- Course Details
- Course Aims
- Student Learning Outcomes
- Prerequisites
- Teaching Strategies
- Assessment
- Academic Honesty and Plagiarism
- Course Schedule
- Material Covered
- Course Evaluation and Development
- Credits for Material

Course Staff

Contact Details

Staff Name	Role	Email	Phone
Gabi Keller	Course Convenor	keller at cse.unsw.edu.au	56032

Consultations

Staff Name	Day/Time	Location
Gabi Keller	Tuesday 1400-1600	K17-501 (please contact me via email before dropping by))

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.

Course 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.

Student 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

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.

Teaching Strategies

COMP1927 includes lectures, tutorials, labs, quizes, 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 quizes. 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.

Assessment

- **Exam:** There will be a three-hour closed-book exam for this course. This exam will consist of a mixture of multiple choice questions, short answer questions and small programming tasks and analysis exercises, all in the C language. The final exam contributes 60% of the final mark for this course.

  There will be no second-chances to pass the final exam. Attendance at the exam will be treated as an indication by you that you are feeling well enough to sit that exam. If you are feeling so unwell on the day of the exam 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 exam, 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 exam may be a written or an oral exam. Note that supplementary exams are typically more difficult than the original exam to reflect the fact that you have had extra preparation time.

- **Assignments:** the assignments are each worth 10% of the final mark.

  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, and receive an entry in the school plagiarism register. 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.

  Assignment related questions may be part of the exam.

- **Lab marks and Tutorial participation marks:** Attendance of the tutorials and labs is compulsory! You need a minimum of 70% of the combined lab and tute marks to pass the course. You will get a weekly tutorial participation mark, which adds up to 10% of the overall mark, and a weekly lab mark (also 10%).

  Submitting your lab solutions for assessment requires you to use the *submit* 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 submit code that does not compile, you receive 0 labmarks.
  - If code compiles without warnings, runs successfully you 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 submit your code before the end of your 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).

- Relative weight of each component for the final mark:

Assignment 1	10 %
Assignment 2	10 %
Lab & Tute Mark	20 %
Final Exam	60 %

Academic Honesty and Plagiarism

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.

Course Schedule

This schedule is only preliminary

No schedule is available (yet)

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.

Course Evaluation and Development

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. Also, please feel free to contact your tutor or lecturer with any feedback or suggestions for improvement during the session.

Credits for Material

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