1927 Higher Data Structures and Algorithms
Course Introduction

Summer Session 09x1 (2008/2009)

Pre-requisites/Background Knowledge

The official pre-requisite for COMP1927 is either COMP1917 (in 2008), COMP1911 (in 2006 or 2007), or COMP1921 (in session 2 2008).

We expect that by completing the pre-requisites, you gained a solid understanding of the the programming process and the C language. In particular, you should to be able to design, implement, test, and document small C programs (around 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, so we'll assume that you're fluent in C when we start. If you're not confident in C, we strongly suggest some revision of the COMP1917 material before the start of semester.

We also aim to expand your knowledge of tools and techniques for the development of software.

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 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
- be able to package a set of data structures and algorithms as an abstract data type
- be able to develop and maintain software systems in C that contain thousands of lines of code

Course Details

- **Course Code/Title:** COMP1927 Higher Data Structures and Algorithms
- **Units of Credit:** 6
- **Course WebSite:** http://www.cse.unsw.edu.au/~cs1927/
- **Handbook Entry:** http://www.handbook.unsw.edu.au/undergraduate/courses/2009/COMP1927.html

Web Pages

The course homepage can be accessed via:

http://www.cse.unsw.edu.au/~cs1927/

If, for any reason, WebCMS server goes offline, most of the course material is also accessible via:

http://www.cse.unsw.edu.au/~cs1927/09x1/index.html

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

Staff

Name	Room	Phone	Email	Role
Angela Finlayson	K17 412-04	9385 6852	angf or cs1927	Course administrator, Lecturer

Angela Finlayson will be available for several consultations each week on

- wednesday from 1-2pm and
- friday from 1-2pm.

For details of locations and for more consultation times please consult the consultations schedule.

You can also contact me via the course forum or via email.

Syllabus/Schedule

The following is a summary of the topics that will be covered in this course. The schedule is tentative and will be refined over the next week; it may also vary as the semester proceeds. There will be different levels of detail on the different topics (some will provide overviews, some will be more detailed).

Week	**Topic** (Textbook reference)
Week 1	Introduction; C Revision (Ch.3) ; Elementary Sorting Methods (Ch. 6)
Week 2	Quicksort & Merging and Mergesort (Ch.7+8)
Week 3	Abstract Data Types (Ch.4)
Week 4	Recursion and Trees (Ch.5)
Week 5	Symbol Tables & Binary Search Trees (Ch.12)
Week 6	Balanced Trees (Ch.13)
Week 7	Graph Properties and Types (Ch.17)
Week 8	Graph Search (Ch.18)

Full details of the schedule will be given on the course web site.

Textbook

COMP1927 follows the contents of the pair of books:

- 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.

We will also be producing our own lecture notes, but these will not be nearly as detailed as the textbook.

Teaching Strategies

COMP1927 includes lectures, tutorials, labs, assignments and a text book.

Lectures aim to convey basic information about the course content and to model the practices and techniques involved in software development (i.e. we do demos).

The most important components of the course, however, are the tutorials, labs and assignments. Tutorials aim to clarify and refine the knowledge that you got from lectures and from reading the text book. Labs and assignments are where *you* get to put together and practise all of the ideas from the lectures, tutes and text. The *only* way to develop the skills to do effective software development is by *practising* them. The labs and assignments 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 assignments and lab exercises, you're wasting the course's most valuable learning opportunities.

Lectures

Each week there will be five hours of lectures during which theory, practical demonstrations and case-studies will be presented. The sixth hour of lectures (ie the last hour of friday lectures) aims to deliver revision of the more difficult material that has already been presented in the course. This will include new examples and demonstrations and will be more interactive in nature. This hour is flexible and topics for revision may be chosen by popular demand.

Day	**Time**	**Location**
Wednesday	9am-12	Civil Engineering G1
Friday	9am-12	Civil Engineering G1

Lectures convey a small amount of information about the course content, but their main aim is to try to stimulate you to think about concepts and techniques. 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 *not* to simply get the tutor to give you the answers; the aim is to focus on just one or two of the exercises and work through them in detail, discussing as many aspects, alternative approaches, fine details, etc. as possible. You need to be active and ask questions in tutorials to get the most out of them and to obtain tutorial marks.

There may not be enough time in each tutorial to finish every question. This is ok. These questions can be used as extra practice questions for your own private study.

Lab Classes

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. These exercises will be released by the Friday of the week preceding the lab class. You should attempt them *before* going to the lab, to maximise the usefulness of the class. The exercises will need to be submitted (for our records) and assessed by your tutor during the lab. The tutor should provide feedback on your approach to the problem and on the style of your solution. Labs may be marked at the *latest* , during the following week's lab.

Assignments

In the assignments, you will work on more substantial programming exercises or analytical tasks.

All assignments will 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.

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, especially after completing at least your first semester of study.

Plagiarism will not be tolerated; assignment submissions will be checked for similarity and anyone found involved in plagiarism will be awarded an overall mark of zero for the entire course. This does *not* look good on your academic record.

Note, in contrast to the above, that re-use of algorithms and data structures developed by others (outside this course) is actively *encouraged*. Programmers are naturally lazy and will borrow ideas from others if it helps them to get the job done quicker. In real life, you would probably borrow code as well as algorithms (but of course you would always give credit to the original author). In this course, you may re-use ideas freely but *may not re-use code*.

Late Submission

Any 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.

Exam

There will be a three-hour closed-book *written* exam for this course on friday the 6th of February (A week after lectures finish). 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. You will not be expected to remember all of C's details; concise reference information will be provided with the exam paper. 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.

Note that supplementary exams are typically more difficult than the original exam to reflect the fact that you have had extra preparation time.

Assessment

Your final mark in this course will be based on components from the assignment work, tutorial participation, labs and the exam. Note that the exam is a hurdle, so that if you score less than 20% in the final exam, you cannot pass 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.

Tutor mark	10 marks
Assn1	10 marks
Assn2	10 marks
Assn3	10 marks
Final exam	60 marks

Tutor mark

The tutor mark is a combination of your tutorial mark and your lab mark.

The tutorial mark is based upon tutorial attendance and participation. You can earn 0.5 of a mark for each tutorial from weeks 1 to 8, which results in a maximum mark of 4. You can only receive these marks if you *attend and participate*.

Each week there will be lab exercises worth 3 marks each, resulting in a possible 24 lab marks in total.

You tutor mark is determined by adding together your tutorial mark and your lab mark and dividing by 2. Your tutor mark will be cut off at 10. This means that you can miss a few lab exercises or tutorials and still get full marks. As a consequence, no special considerations apply for missing a few weeks of tutes/labs due to illness.

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.

Credits for Material

Ideas and material for the COMP1927 this session have been borrowed from

- notes and slides by John Shepherd (COMP1927 08s2)
- slides by manuel chakravarty (COMP1927 08s1)
- slides by Nandan Paramesh (COMP1921 08x1)
- web pages by Richard Buckland and Maurice Pagnucco (COMP1921 06s2/07s2)
- slides and books by Robert Sedgewick