Course Outline

COMP1927
COMPUTING 2

Semester 1 2014

Contents

- Course Staff
- Course Details
- Course Summary
- Course Aims
- Student Learning Outcomes
- Assumed Knowledge
- Teaching Rationale
- Teaching Strategies
- Assessment
- Academic Honesty and Plagiarism
- Resources for Students
- Course Evaluation and Development

Course Staff

Contact Details

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

Consultations

I'm in the secure area on K17, Level 5. Give me a ring when you're outside on 5 6032. If possible, send me a quick email to let me know in case you're planning to attend the consultation, as I might be outside my office for a few minutes or so and not hear the phone.

Staff Name	Day/Time	Location
Gabriele Keller	Wednesday 1500-1600	K17-501I
Gabriele Keller	Friday 900-1000	K17-510I

Course Details

Course Code: COMP1927
Course Title: 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/2014/COMP1927.html

Course Summary

By the end of this course, we want you to know a range of core data structures and algorithms and be able to use them to build demonstrably high-quality (correct, efficient) software using the C programming language. You should also understand the connection between algorithms and data structures, and the trade-offs with respect to performance, memory consumption and simplicity of the operations.

Course Aims

The aim of this course is to get you to think like a computer scientist. How does a scientist, let alone a computer scientist, actually think?

What many types of scientists try to do is to understand natural systems and processes. For example, a geologist tries to understand the structure of the earth, a biologist tries to understand living organisms, a chemist tries to understand materials and reactions, and so on. Computer scientists don't, as the name might suggest, simply try to understand the structure and behaviour of computers, but are more concerned with understanding software systems. Also, unlike other scientists, computer scientists frequently build the objects that they study.

During this course, we'll be looking at ways of creating, analysing and understanding software. Ultimately, you should be able to answer the question "Is this piece of software any good?" and be able to provide sound reasons to justify your answer.

This course follows on from the introductory C programming courses COMP1917 or COMP1921. We cover additional aspects of the C programming language that were not covered in these course, and also look at some programming tools which were not covered (in detail) in COMP1917 or COMP1921. However, this course is not simply a second C programming course. The focus is on the ideas and abstractions behind the data structures and algorithms that are used.

COMP1927 is a critical course in the study of computing at UNSW, since it deals with many concepts that are central to your future studies in the area. Whether you are studying Computer Science, Software Engineering, Bioinformatics, Computer Engineering, or even a discipline outside the realm of computing, understanding a range of algorithms and data structures and how to use them will make you a much more effective computing problem solver in the future..

Student Learning Outcomes

After completing this course, students will:

- 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

This course contributes to the development of the following graduate capabilities:

Graduate Capability	Acquired in
scholarship: understanding of their discipline in its interdisciplinary context	lectures, assignments
scholarship: capable of independent and collaborative enquiry	lab work, assignments
scholarship: rigorous in their analysis, critique, and reflection	tutorials
scholarship: able to apply their knowledge and skills to solving problems	tutorials, lab work, assignments
scholarship: capable of effective communication	blog
leadership: enterprising, innovative and creative	assignments
leadership: collaborative team workers	lab work
professionalism: capable of independent, self-directed practice	assignments

Assumed Knowledge

The official pre-requisite for this course is either COMP1917 or COMP1921.

Whether or not you satisfy the pre-requisite, we assume that:

- you can program in the C programming language, and are familiar with arrays, strings, pointers, dynamic memory allocation, recursion
- you are able to design, implement, debug, test and document small C programs (up to several hundred lines of code)
- you are able to use a debugger (at least the most basic commands, like stepping through a program, setting break points, inspecting data)
- write simple make files
- you are familiar with the Linux environment on CSE computers

Installing Linux, possibly as a virtual machine, on your own computer would be a bonus. We usually try to design the exercises and assignments so that you can do them on Mac OS or Windows as well, but can't give any guarantees.

Teaching Rationale

Computer Science is, to a large extent, a practical discipline, and so there COMP1927 has an emphasis on practice. Lectures will include exercises where we examine the practice of developing and analysing programs. Tutorials aim to develop analysis and understanding via practical case studies. Lab Classes also provide practice in program development and analysis. Assignment provide large case studies of software development.

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. If you slack off on the assignments and lab exercises, you're wasting the course's most valuable learning opportunities.

The University requires us to assess how well you have learned the course content, and the primary approach to achieving this is via a final exam. An exam is the ultimate summative assessment tool; it gives you a chance, at the end of the course, to demonstrate everything that you've learned.

Especially labs, but also assignments are primary a learning, not an assessment tool, so in an ideal world, I would ask you to do them because you wanted to learn, but award no marks for them, just give you informal feedback. However, even with the best intentions, few people actually manage to do the exercises and assignments when they are not marked and compulsory, so they will make up about 1/3 of your final mark.

Lectures

Each week there will be three hours of lectures during which theory, practical demonstrations and case-studies will be presented. The fourth hour of lectures (Fri 2-3) will be used on demand to review some topics.

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 (shhhhhh) keep quiet.

Note that I will not post the lecture slides before the lecture, to encourage you to concentrate on the fresh material. However, I will always give you textbook references to read for each lecture on the web site.

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. 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 must be active and ask questions in tutorials. Ideally, students should run the entire tute themselves, with the tutor being a moderator and providing additional explanations.

Labs

Lab classes aim to give you practice in problem-solving and program development. Each week, there will be a number of small exercises to work on. These exercises will be released on 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.

Assignments

In the assignments, you will work on more substantial (hundreds of lines of code) programming exercises 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.

Assessment

Your final mark in this course will be based on components from the assignment work, labs and the exam. Note that the exam is a hurdle, so that if you fail the exam badly enough, you cannot pass the course. The following formula describes precisely how the mark will be computed and how the hurdle will be enforced:

```
labs    = total mark for all labs   (out of 10)
ass1    = mark for assignment 1     (out of  10)
ass2    = mark for assignment 2     (out of  10)
tutes   = mark for tutorial attendance & participation (out of  5)
exam    = mark for final exam       (out of 65)

okExam  = (exam > 30/65)        (after scaling)

mark    = labs+ass1+ass2+tutes+exam
grade   = HD|DN|CR|PS  if mark >= 50 && okExam
        = FL           if mark <  50 && okExam
        = UF           if !okExam
```

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.

Resources for Students

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** (Text ref)
Week 1	Introduction; C Revision (Ch.3)
Week 2	Abstract Data Types (Ch.4)
Week 3	Recursion and Trees I (Ch.5)
Week 4	Recursion and Trees II (Ch.5)
Week 5	Elementary Sorting Methods (Ch.6)
Week 6	Quicksort & Merging and Mergesort (Ch.7+8)
Week 7	Symbol Tables & Binary Search Trees (Ch.12)
Week 8	Balanced Trees (Ch.13)
Week 9	Hashing (Ch.14)
Week 10	Graph Properties and Types (Ch.17)
Week 11	Graph Search (Ch.18)
Week 12	DAGs, Minimum Spanning Trees, Shortest Path (Ch. 21)

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

Resources for Students

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 make available our lecture slides, and all of the exercises from lectures. Solutions for all tutorial questions and lab exercises will also be made available (after the due date, of course)

Course Evaluation and Development

This course is evaluated each session using the CATEI system. While this helps us to fix problems in the following semester, I also welcome feedback during the course, either by talking to me directly or to a tutor, as this allows me to address potential issues straight away,