COMP1921 Computing 1B (Session 2, 2014)

Units of credit: 6

Handbook entry: http://www.handbook.unsw.edu.au/undergraduate/courses/2014/COMP1921.html

Staff

Name	Role	Email	Extension	Room
Michael Thielscher	Lecturer-in-charge	mit "at" cse.unsw.edu.au	57129	401I
Mei Cheng Whale	Course administrator	meicheng "at" cse.unsw.edu.au	55683	217B-02

Aims

This course assumes basic programming skills, and develops concepts and skills such as

- computer organisation
- elementary algorithms (such as searching for and sorting data)
- elementary data structures (such as linked lists and trees)
- problem abstraction
- problem solving

Program development and testing, and programming style are important aspects of the course as well. In essence, the aim is to make you a competent programmer.

Assumed Knowledge

- You should have completed a first course in C programming, and be able to write, execute and test simple C programs.
- You should have familiarity with the Unix/Linux operating system.

Learning Outcomes

By the end of the course, you should:

- devise algorithmic solutions for well-specified problems and express these solutions as C programs
- develop suites of test cases to ascertain the accuracy and robustness of your programs
- repair any errors discovered in testing, and analyse and improve the performance of your programs
- understand and explain the run-time behaviour of your programs at the machine level
- make effective use of a number of common, useful data structures

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

Graduate Attribute	How Acquired
the skills involved in scholarly enquiry	reading documentation and lab work
the capacity for analytical/critical thinking and creative problem solving	developing programs in tutes, labs and assignments
the ability to engage in independent and reflective learning	exploring the C language in labs
the skills to locate, evaluate and use relevant information	reading documentation
the capacity for enterprise, initiative and creativity	developing programs in tutes, labs and assignments
a respect for ethical practice and social responsibility	building correct programs
the skills of effective communication	discussions in tutorials

Course in a nutshell

COMP1921 consists of lectures, tutorials, labs, an assignment and a final exam.

Lectures: are scheduled into a 2-hour block on Monday and a 1-hour block on Tuesday from weeks 1 to 12. The schedule of lectures can be found on the course website. The material presented during lectures will not be comprehensive. Students are advised to read the course textbook to augment the ideas being presented: the depth and breadth of information presented in the text and references of a similar standard should be used as a guide for your learning.

Should you have any questions on the content or presentation, please don't hesitate to ask during lectures or post them on the course forum. Usually other people will have similar questions.

Tutorials: Tutorials commence in Week 2 and run until Week 13. Tutorials offer a vital chance for you to discuss problems and their solution in more depth with talented programmers. Please make the most of the opportunity by preparing for them and engaging in the discussion.

We will make tutorial questions available by 5pm on the Thursday of the week before they are to be discussed. In general, tutors will not be providing answers to the tutorial questions directly. Their role is to facilitate the tutorial by steering the direction of the solution development and presentation. To get the most from tutorials you must participate in finding and presenting solutions to tutorial questions.

Resources for Students

The recommended text for this course is:

- Alistair Moffat: *Programming, Problem Solving, and Abstraction with C*, Pearson Educational, Australia, Revised edition 2013, ISBN 978-1-48-601097-4.

The following books might also serve as additional reference material:

- Brian W. Kernighan and Dennis M. Ritchie: *The C Programming Language*, 2nd edition, Prentice Hall, USA, 1988, ISBN 0-13-110370-9.
- Paul Davies: *The Indispensable Guide to C*, Addison Wesley, England, 1995, ISBN 0-201-62438-9.

Assessment

Marking generally follows the 40-60% rule:

- the first 50% of the marks are the easiest to get (by far)
- the marks for the harder part of the problem are harder to get (unfair, I know)

Typically (does not apply to the lab exercises):

- all programs will be 'human' marked for style, design and correctness
- submitting a non-compiling program, the maximum mark will be 50%...
  - ... because a program may be excellent in design but fail to compile for a trivial problem
- if a program compiles:
  - and **does not do** the basics, the mark will be less than 50%
  - and **does do the basics**, the mark will be 50%
  - the basics generally consists of setting up the correct data structure and input processing and possible handling simple cases
- when you submit a program it is generally tested with some input to see if the output is correct
  - each set of input-output is called a *testcase*
    - you will see whether your program passes the given testcases or not
    - we may use more testcases at marking time to test programs
  - *there are no marks for passing a testcase*
    - it is just 'evidence' for the marker that the program is working

Labs

Lab exercises will be released by 5pm on Thursday of the week before they are to be attempted.

It is important that you can do the lab exercises on your own. The exercises are the best indicator of what is expected from you in the course. If you can do them, on your own, and 'on time', then you should have no trouble passing this course. (Most lab exercises take between 15 and 30 minutes.)

Submitting your lab solutions for assessment requires you to use the *submit* command shown at the end of every exercise. The number of exercises each week may vary, but is generally 4.

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, that receives 0 labmarks.

- If code compiles without warnings, runs successfully, is readable, 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 *submit* your code by midnight on the ***Sunday*** after the lab.
- the reason for this strictness is promptness of marking
- 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. Your total labmark will be scaled at the end of the course.

Quizzes

There are 2 'mid-session' quizzes: one each in weeks 4 and 9.

- The quizzes are held instead of the lab in that week, in the same computer laboratory of course.
- Quizzes consist entirely of programming exercises and could include producing test cases.
- The concise C-reference guide will always be available.

Assignment

The assignment is a significant programming exercise and must be done individually. Please pay particular attention to the plagiarism rules. Details of the marking of the assignment will be included in the assignment specification. **15% penalty** will be applied to the (maximum) mark for every 24 hours late after the deadline.

Final Examination

There will be a 3-hour final exam. The examination consists of a ''theory'' component and a ''coding'' component. The coding part of the exam is worth more than the non-coding part in the exam mark. The exercises in the coding part of the final exam are much like the lab exercises in style and difficulty.

Final Course Mark

The marks that you accumulate throughout the course will be scaled as follows:

Component	**Marks**
labs	10
quiz_week4	5
quiz_week9	15
assignment	15
final_exam_theory	20
final_exam_coding	35
final_course_mark	sum of above

There is an important condition:

- **let coding_mark = quiz_week4 + quiz_week9 + final_exam_coding**
- **if (coding_mark * 100 / 55 < 50)**
- **then final_course_mark = coding_mark**

In other words, 55% of your final course mark comes from actual coding (behind a computer, on your own), and you need to pass this part of the course.

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 your first semester of study.

Supplementary Examination

You cannot apply for a supplementary examination for any reason. You can apply formally for *special consideration*:

- a supplementary examination may or may not then be granted, at the discretion of the lecturer in charge
- a supplementary examination will not be granted if you cannot pass having passed the examination

Supplementary examinations can be *full* examinations, or they can test just one component in the course such as your programming skills. They may also include an oral component. Note that a supplementary examination is typically more difficult than the original examination to reflect the fact that you have had more preparation time.

Supplementary Assessment:

- if you have received special consideration then:
  - if the supp exam mark is higher than (any) previous final exam mark, then it replaces your final exam mark
  - if the supp exam mark is lower, then it is ignored

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.

Students suggested that more detailed explanations could be given of some of the more difficult topics like memory management (`malloc`) and data types. They also put forward the idea to actually write code in lectures. More discussion on how to go about solving a given problem was suggested too. Students expressed a preference for labs to be due by Sunday midnight.

Acknowledgments

Slides, notes, and other material are partially from previous lecturers of this course, especially John Shepherd, and Alistair Moffat's book.