UNSW Global

Course Outline

CP1511

Introduction to Programming

Diploma Program

UNSW Global Education

Term 2 2020

1. Staff

Position	Name	Email
Course Convenor & Lecturer	Dr Aarthi Natarajan	a.natarajan@unswglobal.unsw.edu.au

2. Course information

Units of credit (UOC): 6

Pre-requisite(s): none

Total course contact hours: 96

2.1 Course summary

This course introduces students to the basics of programming. Topics covered include:

- fundamental programming concepts
- the C programming language and use of a C compiler
- programming style
- program design and organisation concepts
- program testing and debugging

The course does not assume any previous programming experience.

2.2 Course aims

The course aims for students to become proficient in programming using a high level language, C. By the end of the course, students should be able to construct C programs to solve problems.

2.3 Course learning outcomes (CLO)

At the successful completion of this course you (the student) should be able to:

1. understand the core syntax & semantics of the C programming language including types, I/O, arrays, functions, pointers, structs, file manipulation and dynamic memory allocation
2. given a problem, solve it by proficiently constructing (designing, testing, debugging) a secure, reliable and correct C program
3. understand & employ fundamental data structures including stacks, queues and linked lists
4. use Linux and Unix-like operating systems to develop and test software

2.4 Relationship between course and program learning outcomes and assessments

Course Learning Outcome (CLO)	**Program Learning Outcome (PLO)**	**Related Tasks & Assessment**
CLO 1	Conceptual understanding of computer underpinnings (EA1.2)	Exam, Practical Exams, Labs, Assignments
CLO 2	Understanding of specialist bodies of engineering knowledge (EA1.3)	Exam, Practical Exams, Labs, Assignments
CLO 3	Understanding of underpinnings (EA1.1)	Exam, Practical Exams, Labs, Assignments
CLO 4	Understanding of specialist bodies of engineering knowledge (EA1.3)	Labs, Practical Exams, Assignments

3. Strategies and approaches to learning

3.1 Learning and teaching activities

This course involves a number of teaching activities:

Lectures – 4 hours per week

Lectures present theory and concepts, by way of case studies and practical examples. Lecture notes will be provided in advance of each class. There will be 4 hours of timetabled live streamed lectures each week.

Tutorials – 1 hour per week

Tutorials allow students to collaboratively work through example problems to illustrate lecture idea, and have concepts from lectures clarified by the tutor.

Lab Classes – 3 hours per week

Lab A Sessions:

These lab classes involve small exercises where students build systems that illustrate the ideas covered in lectures. Students show their work to the lab demonstrator for assessment and feedback. In **some** of the 2 hour Lab A classes, students may be required to work in pairs to write software. If students are required to form pairs, this will be indicated in the lab specification for that week. "Challenge exercises" will be provided for students who find the regular weekly exercises too easy and will allow students to earn bonus lab marks. Each lab is worth a possible 1.2 marks. The best 10 out of 11 lab marks are added up and capped at 10. During Lab A in week 12 a practice exam worth 2 marks will be held.

To obtain a mark for a lab exercise you should demonstrate the completed lab exercise to your tutor during a lab class and submit it using give.

You should normally get your lab work assessed during the week for which it is scheduled (i.e. you must complete the week 3 lab exercise during the week 3 lab). If you don't finish it during the lab, you may continue working on it during the week, but you both must submit it (using give) by Sunday 9:59 pm (in the same week as the lab) in order to get any marks for it. You must then also demonstrate your work to your tutor during the first hour of the following week's lab. The code that you submit by Sunday 9:59pm is what will be assessed.

Summary: to obtain any lab marks for the Week X lab, you must do 2 things:

1 submit your lab work by the following Sunday 9:59pm

demonstrate your work to your tutor in the week X lab class OR demonstrate your work at the start of the lab in week X+1

You cannot obtain marks by e-mailing lab work to tutors.

Lab exercises will be assessed using the following grade system:

Grade	Criteria
A+	Outstanding effort; must complete any challenges and go beyond the standard exercises
A	Complete, correct, and clear solution to standard lab exercises (worth full marks)
B	Most of the standard lab exercises completed, or all completed but with one or more major bugs
C	Partial solution only, much of lab not completed or has many glaring errors
D	Submitted something , but it's completely hopeless
.	Not attempted

Optional challenge exercises may be specified for some labs.

There will be more lab marks available than necessary to obtain full marks for the 10% lab component. In other words, the total lab mark will be capped, with a small bonus available for consistently outstanding work.

Lab B Sessions

In the 1 hour Lab B **classes, students work individually on programming. In some weeks (3, 5, 7, 9, 11) this hour will hold assessable practical exams**

These practical exams are assessed and help prepare students for the final exam. These are held in weeks 5, 7, 9, 11 and 12. They are each worth 2 marks.

Assignments – 2 during the course

Assignments are take-home problems that are larger in scope than Lab exercises and require students to use creativity to solve a challenging realistic problem. Each assignment requires students to understand the problem, design a solution, and implement and test their solution. Students will be required to maintain a reflective diary (blog) of their experiences during the assignments.

Online Forum

An online forum allows students to ask and answer questions on the tutorial, lab and assignment exercises, and on lecture material.

Final Exam

The format of the final exam is yet to be finalised. The format and mode of the exam will be advised closer to the end of the term. In the final exam, students will need to solve both theory and practical problems.

3.2 Expectations of students

Students are expected to:

- attend all lectures, and ask questions, but otherwise not disturb other students
- attend all tutorials and actively participate in the discussions
- attend all lab classes and work diligently on the exercises
- do all of the assignment work themselves, asking only the forum or tutors for help

On the course forum, students should:

- use relevant/meaningful message titles on all posts
- ask questions clearly and provide sufficient background information that the question can be reasonably answered
- not post significant pieces of code, especially code for assignments

4. Course schedule and structure

This course consists of 8 hours of class contact hours per week. You are expected to take an additional 5 hours outside classes to complete assessments, readings, and exam preparation.

Week	**Lectures**	**Tutorial and Labs**	**Assessment**	Related CLO
Week 1	Introduction to course/Linux/C; data types; variables, simple I/O, expressions, It Statements	Create/run first C programs on Linux		1,4
Week 2	Loops	arithmetic and simple control		1,2,4
Week 3	Memory and Functions	complex expressions, iteration		1,2,4
Week 4	Arrays	functions, exploring memory		1,2,4
Week 5	Chars, strings	array manipulation	Assignment 1 Released Practical Exam 1	1,2,4
Week 6	Pointers and extra C (for loops, pre/post increment, multi-file compilation)	working with characters and strings		1,2,4

Week 7	Dynamic memory allocation, C implementation memory models, struct	command line arguments	Practical Exam 2	1,2,4
Week 8	Linked Lists	malloc, pointers, struct	Assignment 1 Due	1,2,3,4
Week 9	Abstract Data Types, including stacks and queues	programming with linked-lists	Practical Exam 3 Assignment 2 released	1, 2, 3, 4
Week 10	Recursion including recursion with linked lists	programming and testing ADTs		1, 2, 3, 4
Week 11	Introduction to searching, sorting and complexity	programming with linked lists using recursion	Practical Exam 4	1, 2, 3, 4
Week 12	Exam Information and Revision	sorting and a practice exam	Practical Exam 5 Assignment 2 due	1, 2, 3, 4

5. Assessment

5.1 Assessment tasks

Assessment task	Length	Weight	Due	CLOs
Assessment 1: Programming assignment (arrays)	3 weeks	15%	Monday week 8	1, 2, 4
Assessment 2: Programming assignment (linked data structures)	3 weeks	15%	Friday week 12	1, 2, 3, 4
Assessment 3: Lab exercises (the best 10 out of 11 will be taken)	Throughout semester	10%	Weekly, on Sunday	1, 2, 3, 4
Assessment 4: Practical Exams	Throughout semester	10%	Weeks 5,7,9, 11,12	1,2,3, 4
Assessment 5: Final Exam		50%		1, 2, 3, 4

There are 3 hurdle requirements on the final exam.

Hurdle requirement #1: in the final exam you must solve a task by writing a program that uses an **array**.

Hurdle Requirement #2: in the final exam you must solve a task by writing a program that uses a **linked list**.

Hurdle Requirement #3: in the final exam, you must score at least 40%

You **cannot** pass CP1511 unless you achieve **ALL** of the above hurdles.

Final Mark

Your final mark for this course will be computed using the above assessments as follows:

CourseWorkMark	=	PracticalExamMark + LabMark + Ass1Mark + Ass2Mark	out of 50
ExamMark	=	ExamPracMark + ExamTheoryMark	out of 50
ExamOK	=	ExamMark ≥ 22.5/50 && Pass (List Hurdle & Array Hurdle)	true/false
FinalMark	=	CourseWorkMark + ExamMark	out of 100
FinalGrade	=	UF, if !ExamOK && FinalMark ≥ 50 FL, if FinalMark < 50/100 PS, if 50/100 ≤ FinalMark < 65/100 CR, if 65/100 ≤ FinalMark < 75/100 DN, if 75/100 ≤ FinalMark < 85/100 HD, if FinalMark ≥ 85/100	

5.2 Assessment criteria and standards

In all programming work, the primary assessment criterion is correctness (i.e. does the code produce the expected output/behaviour according to the exercise specification). This will be tested by executing code against a variety of test cases, some of which are available to students, and others of which are used after submission for assessment purposes. Code is also expected to be expressed clearly, with consistent formatting and using relevant variable names.

5.3 Submission of assessment tasks

All assignments will be submitted online via CSE's submission system. Late assignments submissions will be penalized. The exact penalty will be specified in

the assignment specification - typically it is 2% reduction in maximum mark for every hour late.

If you are unable to submit an assignment by the due date, due to medical reasons or other reasons which significantly affect your ability to carry out your work, you should contact the lecturer as soon as possible, preferably well before the assignment deadline. If the lecturer considers that your ability to complete the assignment on time has been adversely affected, an extension may be granted to make up for the time you were unable to work on the assignment.

Lab exercises must be submitted by the end of the Sunday after the lab class. Demonstrators will then look at the exercise, and assess it, possibly asking you to explain what you did in the following lab class. Failure to complete and submit a lab exercise results in a mark of zero for that lab. Lab exercises **must be marked in the following lab**, and cannot be marked at a later week.

5.4. Feedback on assessment

Assignments will be marked after the submission deadline and annotated with comments by the tutor. You can discuss the tutor's comments in a lab class after you have received the feedback.

Lab demonstrators will discuss your lab submission with you during the lab class in the week following the submission.

6. Readings and resources

The course website will make available all lecture material along with code examples.