

UNIT GUIDE
COMP332 Programming Languages
Second Half 2011

Credit points: Three

Prerequisites: 40cp and (COMP225(P) or COMP229(P))

Teaching Staff

Convenor and Lecturer (weeks 1-7, 13)

Tony Sloane, Anthony.Sloane@mq.edu.au, E6A315, 9850 9582

Office hours: Monday 5-6pm (weeks 1-7,13)

Lecturer (weeks 8-12, 13)

Matthew Roberts, altmattr@zoho.com

Office hours: Monday 5-6pm (weeks 8-12,13), E6A340, 9850 9540

Co-badging

This unit is not offered under any other unit code.

Unit description

Programming, of one form or another, plays a central role in computing and consequently the study of programming languages comprises a vital component of modern computer science. Considering this observation, this unit has two aims which we pursue in parallel:

- to provide students with a comprehensive view of the features of modern programming languages, and
- to introduce students to the implementation of programming languages.

In particular, we discuss the syntax and semantics of programming languages while, at the same time, introducing participants to commonly applied language implementation techniques. Practical language implementation exercises supplement the lecture material.

Assumed Knowledge

COMP332 requires:

- Experience writing programs in at least one C-like programming language utilising the imperative and object-oriented styles.
- An understanding of algorithms and data structures, such as lists, stacks, and trees and experience programming with them.

Expected Learning Outcomes

Students successfully completing COMP332 will be able to:

1. Evaluate programming languages on a feature-by-feature basis and explain the meaning and use of different features.
2. Specify programming language syntax, semantics and translation using high-level formal notations.
3. Implement a small to medium-sized programming language using the standard phase-based compiler architecture.
4. Explain the implementation of a non-trivial software system and demonstrate its correctness using appropriate test cases.

Graduate Capabilities Developed

The COMP332 learning outcomes develop general graduate capabilities, including

- *Discipline Specific Knowledge and Skills (1-4)*
- *Critical, Analytical and Integrative Thinking (1, 4)*
- *Problem Solving and Research Capability (2, 3)*
- *Creativity and Innovation (2, 3)*
- *Effective Communication (1, 4)*

Classes and Workload

Each week COMP332 has three hours of lecture, a one-hour tutorial and a one-hour practical. The tutorials and practicals are timetabled together into a single two hour mixed class. Practical work starts in Week 1 and tutorial work starts in Week 2.

For details of days, times and rooms consult the University timetables webpage (<http://www.timetables.mq.edu.au>).

Macquarie University generally expects students to work for about three hours per credit point each week. Since COMP332 is a 3 credit point unit, it should therefore require about nine hours per week for average students. There are five hours per week of classes during teaching weeks; therefore an average student should expect to spend about four hours out of class per week on COMP332 during those weeks. In the lecture break when there are no scheduled classes, that time should be used for additional study.

Online Resources

COMP332 Web Home Page: <http://www.comp.mq.edu.au/units/comp332/>

In 2012 Macquarie University will be moving to a new online learning system called [iLearn](https://ilearn.mq.edu.au) (<https://ilearn.mq.edu.au>) which is based on Moodle. As part of a trial of iLearn in Semester 2, 2011, all Department of Computing units will use the iLearn system.

COMP332 will use iLearn for delivery of class materials, discussion boards, online self-tests, submission of work and access to marks and comments. Students should check the COMP332 Programming Languages, Second Half, 2011

iLearn site regularly for unit updates. Because COMP332 is using iLearn, materials will not be posted on this page and the unit will not be using the University's older Online Learning @ MQ site (WebCT/Blackboard).

Questions regarding the content of this unit, its tutorials or practicals should be posted to the appropriate discussion board on iLearn. In particular, any questions which are of interest to all students in this unit should be posted to one of these discussion boards, so that everyone can benefit from the answers.

Technology Used and Required

The practical work in this unit involves programming in the Scala language (<http://www.scala-lang.org>) which will give students experience with modern programming language features that we expect to see in mainstream languages in the future.

We will also use the Kiama language processing library (<http://kiama.googlecode.com>) that is being developed by our Programming Languages Research Group. Kiama provides high-level facilities for writing processors such as compilers in Scala and makes it possible for students to implement most of a compiler from scratch within the semester.

Instructions will be provided on how to use Scala and Kiama on the laboratory machines and how to download it for use on your own machines.

Recommended Readings

Recommended Text

Programming Language Pragmatics (Third Edition) by Michael L. Scott, Morgan Kaufmann, 2009

We cover much of the material in this book. Earlier editions of this book can also be used, but be aware that chapter and section references may be different and some material may not be available. You will find the lecture material much easier to understand if you read the text and lecture notes before the lectures. The lecture schedule below describes the relevant sections of the text.

Supplementary Reading

The Macquarie library contains many books on programming languages and compilers that you may want to use to supplement the text and lecture notes. The following list mentions some that you may find helpful.

- Principles of programming languages: design, evaluation, and implementation. MacLennan.
- Programming languages: design and implementation. Pratt and Zelkowitz.
- Concepts of programming languages. Sebesta.
- Programming languages: concepts and constructs. Sethi.
- Introduction to compiler construction. Waite and Carter.
- Compilers: principles, techniques and tools. Aho, Sethi, and Ullman.
- Modern compiler implementation in Java. Appel.

Teaching Strategy

COMP332 is taught through lectures, tutorials and practical sessions in the laboratory. Lectures are used to introduce new material, give examples of programming language constructs and their implementations and put this material in a wider context. While lectures are largely one to many presentations, you are strongly encouraged to ask questions of the lecturer to clarify anything you might not be sure of.

Tutorial sessions are small group classes which give you the opportunity to interact with your peers and with a tutor who has a sound knowledge of the subject. You will be given problems to solve each week before each tutorial and you will be asked to submit your solutions in an electronic form (as a PDF or Microsoft Word document) through iLearn.

Preparing solutions to tutorial exercises before attending a tutorial is important because it will allow you to discuss the problems effectively with your tutor and maximise the comments you get on your work.

The practical sessions give you an opportunity to practice your programming skills under the supervision of your tutor. Each week you will be given a number of problems to work on; it is important that you keep up with these problems as doing so will help you understand the material in the unit and prepare you for the work in assignments.

In summary, each week you should:

- Attend lectures, take notes, ask questions.
- Submit answers to specific tutorial questions, as a PDF or Microsoft Word document, through iLearn.
- Attend your tutorial session, seeking comments from your tutor on your submitted work.
- Work on the practical exercises (or your assignment submission) and seek comments from the tutor on your work.
- Read assigned reading material, add to your notes and prepare questions for your lecturer or tutor.
- Read lecture notes and book chapters before attending the lectures on that material.
- Start working on any assignments immediately after they have been released.

Lecture notes or slides will be made available each week but these notes are intended as an outline of the lecture only and are not a substitute for your own notes or reading the required text and additional material.

Lecture Plan

Week	Topic	Reading
1	Introduction to programming languages and their implementation	1
1-2	Programming language syntax: specification and recognition	2.1-2.3.3
3	Names, scopes and bindings	3.1-3.6.3

Week	Topic	Reading
4-5	Semantic analysis and attribute grammars	4.1-4.4, 4.6
6	Data types	7.1-7.8, 7.10
7	Control flow	6.1-6.6 (not 6.5.4)
	<i>Lecture Recess - Two Weeks</i>	
8	<i>No lecture due to Labour Day holiday</i>	
9-10	Building a runnable program	14.1, 14.3-14.6
11	Subroutines and control abstraction	8.1-8.6
12	Data abstraction and object-oriented programming	9.1-9.4
13	Review, Exam Discussion	

Summary of Assessment Tasks

Task	Duration	Learning Outcomes Assessed	Weighting
Weekly Exercises	Weeks 1-12	1 and 2	10%
Assignment One: Syntax Analysis	Weeks 2-5	2, 3 and 4	10%
Assignment Two: Semantic Analysis	Week 6-8	2, 3 and 4	10%
Mid-Semester Examination	Week 7 45 minutes	1 and 2	15%
Assignment Three: Code Generation	Week 9-12	2, 3 and 4	10%
Final Examination	3 hours	1 and 2	45%

Weekly Exercises

In most weeks, you will be asked to submit a solution to one or more of the tutorial exercises. Completing a solution to the exercises should take no more than 30 minutes. Each week is worth 1% of the assessment with a maximum of 10% counting toward your final assessment. Your tutor will assess your solutions and provide comments in the next class.

Assignments

The three assignments involve specifying, implementing, testing and documenting parts of a programming language implementation. Each assignment is expected to take between 15 and 20 hours of work. Assignment specifications including standards of achievement will be distributed on a schedule as shown in the table above. We plan to return assignment submissions with comments within two weeks of submission.

As befits their role in the assessment, the assignments are a significant amount of work and usually involve applying unfamiliar ideas to practical problems. Hence you are unlikely to be able to do them in the week before they are due. Make sure you leave enough time to understand the relevant ideas and code before you try to solve the assignment.

You are encouraged to:

- set your personal deadline earlier than the actual one;
- keep backups of all your important files;
- make sure that no-one else picks up your printouts.

Late work will not be accepted. If you cannot submit on time because of illness or other circumstances, please contact the lecturer before the due date.

Examinations

The examination assessment of this unit is split into two components: a mid-semester examination and a final examination.

The compulsory mid-semester examination will be held in week 7 in the first hour of the lecture. This examination will test your understanding of the material from weeks 1-5, and will be 45 minutes long. You will receive comments about this examination to enable you to adjust your study before the final examination.

The final examination will be a three-hour examination held during the usual University examination period covering all topics. More information about the structure of the final examination will be provided during the semester.

Assessment Standards

COMP332 will be graded according to the following general descriptions of the letter grades.

- **High Distinction (HD, 85-100):** provides consistent evidence of deep and critical understanding in relation to the learning outcomes. There is substantial originality and insight in identifying, generating and communicating competing arguments, perspectives or problem solving approaches; critical evaluation of problems, their solutions and their implications; creativity in application.
- **Distinction (D, 75-84):** provides evidence of integration and evaluation of critical ideas, principles and theories, distinctive insight and ability in applying relevant skills and concepts in relation to learning outcomes. There is demonstration of frequent originality in defining and analysing issues or problems and providing solutions; and the use of means of communication appropriate to the discipline and the audience.

- **Credit (Cr, 65-74)**: provides evidence of learning that goes beyond replication of content knowledge or skills relevant to the learning outcomes. There is demonstration of substantial understanding of fundamental concepts in the field of study and the ability to apply these concepts in a variety of contexts; plus communication of ideas fluently and clearly in terms of the conventions of the discipline.
- **Pass (P, 50-64)**: provides sufficient evidence of the achievement of learning outcomes. There is demonstration of understanding and application of fundamental concepts of the field of study; and communication of information and ideas adequately in terms of the conventions of the discipline. The learning attainment is considered satisfactory or adequate or competent or capable in relation to the specified outcomes.
- **Fail (F, 0-49)**: does not provide evidence of attainment of all learning outcomes. There is missing or partial or superficial or faulty understanding and application of the fundamental concepts in the field of study; and incomplete, confusing or lacking communication of ideas in ways that give little attention to the conventions of the discipline.

The standards of achievement that will be used to assess each of the assessment tasks with respect to the letter grades are as follows. Where applicable, more specific versions of these standards will be provided with the assessment task descriptions.

Learning Outcome 1: Evaluate programming languages on a feature-by-feature basis and explain the meaning and use of different features.

P	Can correctly reproduce facts and definitions across a breadth of concepts, but lacks depth of understanding.
Cr or D	Exhibits breadth and depth of understanding of concepts. Can use terminology accurately in new contexts. Can express ideas in their own words and has an understanding of the limits of their understanding.
HD	As for Cr or D and is aware of the context in which the concepts are developed and their limitations. Able to generate and justify principles and hypotheses for existing or new concepts.

Learning Outcome 2: Specify programming language syntax, semantics and translation using high-level formal notations.

P	Can use notations to specify familiar language concepts in ways that are close to those discussed in lectures or notes.
Cr or D	As for P and can apply formal notations to describe language concepts that have not previously been seen.
HD	As for Cr or D, and can recognise the limitations of formal notations for specifying some language concepts and is able to propose alternatives.

Learning Outcome 3: Implement a small to medium-sized programming language using the standard phased-based compiler architecture.

P	Can implement the basic features of a programming language similar to examples provided. Uses basic standards for code comprehension such as variable naming or documentation.
Cr or D	As for P and can use provided general techniques to implement language concepts whose detailed implementation in code has not previously been discussed. Has well-developed skills for writing comprehensible, modular and well-documented code.
HD	As for Cr or D and can develop new techniques to implement language concepts beyond those provided.

Learning Outcome 4: Explain the implementation of a non-trivial software system and demonstrate its correctness using appropriate test cases.

P	Adheres to basic standards for presentation of written work. Can produce a basic description of the main aspects of a software system. Can describe basic test cases for the software under study.
Cr or D	Adheres to all academic standards for presentation of written work. Able to describe all relevant aspects of a software system to an appropriate level of detail. Can articulate the principles behind the design of suite of test cases.
HD	As for Cr or D and can critically evaluate aspects of the software system under study and the appropriateness of testing as a method for demonstrating software correctness.

These assessment standards will be used to give a numeric mark out of 100 to each assessment submission during marking. The mark will correspond to a letter grade for that task according to the University guidelines. The final raw mark for the unit will be calculated by combining the marks for all assessment tasks according to the percentage weightings shown in the assessment summary.

On occasion your raw mark for the unit may not be the same as the Standardised Numeric Grade (SNG) which you receive as the result. Under the Senate guidelines, raw marks may be scaled to ensure that there is a degree of comparability across the university, so that units with the same past performances of their students should achieve similar results.

Extension Requests and Special Consideration

The only exception to not completing an assessment task in the time allocated or not sitting an examination at the designated time is because of serious documented illness or unavoidable disruption. In these circumstances you may wish to consider applying for Special Consideration. Information about unavoidable disruption and the special consideration process is available in the University Special Consideration policy (http://www.mq.edu.au/policy/docs/special_consideration/policy.html). The relevant form is on the web (<http://www.registrar.mq.edu.au/Forms/APScons.pdf>).

If you apply for Special Consideration and it is judged by the Department of Computing that your performance on an examination has been affected adversely by the circumstances documented in the consideration request, you will be required to sit a Supplementary Examination. The Supplementary Examination will normally be scheduled after the official examination period, but may be earlier in the case of a mid-semester examination. For details of the Special Consideration policy specific to the Department of Computing, see the Department's special consideration policy page (http://www.comp.mq.edu.au/undergrad/policies/special_consideration_policy.htm).

Changes to COMP332 since the Last Offering

This offering of COMP332 is very similar to the 2010 offering. The only major change is the detail of the language to be implemented in the assignments. We also plan to provide more detailed guidance on the Scala language than in previous offerings.

Useful Resources

All students should familiarize themselves with the policies that govern learning and teaching at Macquarie University, in particular:

- Macquarie University Academic Honesty Policy
http://www.mq.edu.au/policy/docs/academic_honesty/policy.html
- Macquarie University Assessment Policy and Code of Practice
<http://www.mq.edu.au/policy/docs/assessment/policy.html>
http://www.mq.edu.au/policy/docs/assessment/policy_code_of_practice.html
- Department of Computing Special Consideration Policy
http://www.comp.mq.edu.au/undergrad/policies/special_consideration_policy.htm
- Macquarie University Special Consideration Policy
http://www.mq.edu.au/policy/docs/special_consideration/policy.html

HELP301 is a service provided by the Department of Computing where tutors are available outside class time to answer questions. If you have a general question, any of the tutors may be able to help you; if your question is specific to COMP332, you will need to find the COMP332 tutor. Consult the HELP301 timetable in the laboratories for details of tutor availability.

Enquires relating specifically to COMP332 that can't be addressed by a tutor should be directed to the Unit Convenor or Lecturer.

The Department of Computing has established a Staff-Student Liaison Committee at each level (100, 200, 300) to provide all students studying a Computing unit the opportunity to discuss related issues or problems with both fellow students and staff. Information about the 300-level Liaison Committee will be made available in lectures and on the Web at:

<http://www.comp.mq.edu.au/undergrad/info/liaison/300-level/>

Macquarie University also provides a range of Academic Student Support Services. Details of these services can be accessed at <http://www.student.mq.edu.au>.

If all other avenues have been exhausted, serious issues relating to study in the Department of Computing or in this unit in particular should be raised with the Department's Director of Teaching (Dr. Christophe Doche) or the Head of Department (Prof. Bernard Mans).