

## COMP2091 COMPUTING 2

**Pre-requisites:** COMP1091

**Co-requisites:** Nil

**Exclusions:** COMP2011, COMP2071

**Session offered:** S1 or S2

**UOC:** 6

### **Course Information:**

This course builds on the introduction to problem solving initiated in COMP1091 by expanding the range of techniques for designing and implementing small-scale software. More features of the implementation language are explored, and common data structure abstractions and representations are examined.

### **Course Objectives:**

At the end of the course students should:

- a. Understand how algorithms and data structures combine to produce effective solutions to common problems.
- b. Have developed advanced skills in the design of reliable programs to solve small to medium problems.
- c. Be familiar with the full range of features of an implementation and interfacing language.
- d. Understand how to manipulate data stored in linked structures representing sequences, dictionaries, priority queues and graphs.
- e. Have knowledge of memory models, limitations of numeric data representation, and how data is passed to and from libraries.
- f. Understand common formats for representing data structures in files.

### **Learning Outcomes:**

On successful completion of the course the student should be able to:

1. Design and implement small to medium programs in ANSI C, using a range of suitable data structures.
2. Use advanced debugging techniques and tools to correct defects.
3. Interpret and manipulate XML and binary file formats.

### **Teaching Methods:**

Lectures: 3 *hrs/week* Tutorials: 1 *hr/week* Labs: 1.5 *hrs/week*  
(actually Tut-lab: 2.5 *hrs/week*)

### **Syllabus:**

- Abstract data types and data structures: sequential (lists, stacks, queues), dictionaries (binary trees, hashing), priority queues (heaps), graphs and networks. Implementation of associated algorithms in C. Informal analysis of efficiency.
- File structures: binary formats, direct access techniques, XML. Data compression, Huffman coding. Floating point representation and limitations.
- C features: conditional compilation, storage classes, memory management, function parameters, using profilers and debuggers. Interfacing to libraries. Introduction to multi-threading.

**Laboratory Program:**

- Using *gdb* and *gprof*, memory modelling, simple ADTs, empirical comparison of dictionary implementations, Huffman encoder/decoder, shortest path/connectivity algorithm, sorting, *flex*-based XML parser generator.
- Other possibilities: unlimited precision arithmetic, control system simulation, .digital filters (EE/CE)
- Assignment work extends and complements the laboratory program.

**Contribution to Graduate Attributes:** (from UNSW Graduate attributes – you need only write the corresponding number from the list)

- a. 2
- b. 3
- c. 4
- d. 9
- e. 10

**Recommended Text(s):**

- TBA.

**Further Text(s) and Reference(s):**

- TBA.
- Additional web reference materials according to topic.

**Assessment Methods:**

The assessment scheme in this course is based primarily on the demonstrated acquisition of practical skills.

**Assessment Weighting:** (include appropriate methods below with % weighting – and brief description)

- Final Examination (60 %) Held under laboratory conditions, primarily practical tasks, 2\_ hours.
- Laboratory Assessment (10 %) Tutorial participation and completion of laboratory work.
- Assignments (30 %) Three spread over session.

*(these methods and weightings are subject to review)*