Overview

Faces

Studying Computer Science

Dos and don’ts

Final Year Project

Infos
Faces
John Tucker (Professor)

Mathematical theories of specification and computation. Topological data types. Synchronous concurrent algorithms and their application. Logic, algebra and applications.

Head of Department
Lecturing staff and their research interests.
Phil Grant (Senior lecturer)

Logic Programming and expert systems.
Multimedia communications.
Parallel processing.
Evolutionary computing.

Dean of Science
Min Chen (Professor)

Computer Graphics and visualisation.
Multimedia communications.
Computational geometry.
Distributed computing.
Faron Moller (Professor)

Models of concurrent computation.
Modal and temporal logic.
Equivalence and model checking of infinite state systems.

CS-116 Modelling Computing Systems
Mike Webster (Professor)

Numerical analysis.
Computational fluid dynamics.
Parallel processing.
Multimedia.
Neal Harman (Lecturer)
Algebraic specification of microprocessors. Algebraic specification
languages and environments. Formal specification methodologies.
Mark Jones (Lecturer)

Markus Roggenbach (Lecturer)

Mathematical theories of specification.
Algebraic specification.
Process algebra.
Tools support for specification languages.
Programming languages.

CS-132 Algorithms and Computation
Anton Setzer (Lecturer)

Logic, proof theory, type theory, and programming with dependent types.

: Induction Lecture 22 September 2004
Theory of object oriented programming. Theorem proving.
John Sharp (Lecturer)

Data flow computing.
Program design environments.
Parallel Processing.
Functional programming.

CS-121 Data Structures
Roger Stein (Lecturer)

Knowledge based systems.
Verification.
Intelligent machine interfaces.

CS-134 Professional Issues and Software Engineering
Barry Blundell (Lecturer)

3D volumetric display.

CS-113 From Languages to Hardware
Oliver Kullmann (Lecturer)

Complexity Theory.
Algorithms for hard problems.
Combinatorial optimisation.
Combinatorics.
Propositional proof systems.
Satisfiability problem and generalisations.
UB (Lecturer)

Logic, proof theory and applications.
Domain theory.
Theorem provers and program synthesis.

CS-125 Logic Programming
Head of year One
Monika Seisenberger (Tutor)

Formal methods.
Proof Theory.
Program Synthesis.

CS-151 Introduction to Computing
Andy Gimblett (Tutor)

Formal methods.
Operating systems.
Data communications and networks.

CS-101 Computing Skills
Chris Whyley (Tutor)

Data structures.
Object technology.
Compilers.
Volume graphics.

CS-111 Program Design
CS-131 Programming Laboratory
Edwin Beggs (Lecturer, Mathematics)

Operator Algebras.
Noncommutative differential structures.
Non-trivially associated tensor categories.
Hopf algebras.
Integrable field Theories.

MAM111
Logic and Foundation of Mathematics
Roger Hindley (Lecturer, Mathematics)

Mathematical logic.
Lambda-calculus.
Combinatory logic.
Type-theory.

MAM113
Mathematics for Computation
Studying Computer Science

Year One

10 compulsory courses. Lab classes, weekly tutorials, all compulsory. Exams at the end of each teaching block. Marks do not count towards the final degree.
Progression rules for Year One:

- All core modules passed ($\geq$ 40%). Core modules are:
  - CS-111 Program Design
  - CS-116 Modelling Computing Systems
  - CS-121 Data Structures
- All modules $\geq$ 20%.
- At least 80 credits (120 possible).
Year Two

All 12 courses compulsory. Tutorials, group project, coursework, all compulsory. Exams.
At the end of Year Two chose from Project Selection Brochure your Final Year Project.
Industrial Placements.

Final Year

Project compulsory (30 credits). Choose 9 courses (10 credits each) from given list. Exams.
Why maths matters

Computer Science (like any other science) emphasizes understanding and analysis.

This requires abstraction, that is, concentration on the essential aspects of a problem while omitting less important details.

The result of abstraction is a simplified (but possibly still rather complex) model of the original problem that can be understood and analysed.

For this we need the language and methods of mathematics.
We teach exactly the maths you need, starting almost from scratch.

Therefore, it doesn’t matter whether you have A-levels in maths or not.

Maths is needed also for apparently ‘practical’ subjects (like e.g. Computer Graphics).

Conversely problems in computer science have initiated fascinating new developments in mathematics.

Believe it or not, computer science maths is great fun.
Dos and Don’ts

Do

- be ahead (not behind)
- meet deadlines (coursework, reports)
- attend lectures, practicals and tutorials (will be monitored!)
- ask
  - in or after the lecture
  - in the tutorials
  - in (joint) appointments (not only shortly before the exams)
○ fellow students
Don’t be complacent,
don’t plagiarize,
don’t panic,
don’t wait.
Some statistics: CS-125 Logic Programming
first class (marks $\geq 70\%$) 20%
failure (marks $< 40\%$) 32%
Average marks: 48%

first class result average attendance $> 80\%$
failure average attendance $< 50\%$

attendance $> 80\%$ average result 66%
attendance $< 50\%$ average result 36%

: Induction Lecture 22 September 2004
Final Year Project

Your Final Year Project will be a substantial piece of scientific work. It has crucial influence on your final mark.

At our Undergraduate Colloquium at Gregynog you will have the opportunity to give a scientific talk on your project (along with other invited speakers from academia and industry).

You will learn how to write a good scientific document and how to give a good scientific talk in your first year tutorials.
Gregynog

Annual Undergraduate Student Colloquium
http://www.swan.ac.uk/compsci/undergrad/gregynog/

: Induction Lecture 22 September 2004
Pictures taken by students last year.

“Rather like Hogwarts, there were lots of stairs, although somewhat disappointingly they remained stationary.
throughout.”