Computer Science is a very exciting subject to study. The sheer variety of technologies that are available and that keep being invented, and the transformations that they are inducing in all sectors of activity and the well-being of societies, create huge opportunities for graduates. In the latest Research Assessment Exercise, we ranked 11th in the UK for the quality of our research output, with over a quarter of our publications recognised as world leading, and a further half internationally excellent.

You will be taught by world leaders in several areas of research such as Machine Learning (the science of systems that can learn from data), Algorithms, Bioinformatics, Software Language Engineering, Distributed Systems, and Information Security. This means that you will be exposed to the methods and techniques that cutting-edge companies are looking for, to become leaders in their sectors.

ENTRY REQUIREMENTS

The courses listed below are open to Study Abroad, International Exchange and Erasmus students who study here for a full year, or for Term 2&3 only. Computer Science courses are not available to students who are here for the Autumn Term as all course assessments and exams take place in Term 3 only. Students must have sufficient evidence of previous experience and knowledge as stated in the individual course pre-requisites. Please note that these courses may be adjusted slightly over the coming months which may involve some changes to the course content, learning objectives and summative assessment.

The information contained in the course outlines on the following pages is correct at the time of publication but may be subject to change as part of our policy of continuous improvement and development.
**Level One:**

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
<th>Credits</th>
<th>Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS1801</td>
<td>Object Oriented Programming</td>
<td>1 unit</td>
<td>Full Year</td>
</tr>
<tr>
<td></td>
<td>• This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• You must be enrolled for the full academic year to take this course.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS1820</td>
<td>Computing Laboratory (robotics)</td>
<td>1/2 unit</td>
<td>Term 1</td>
</tr>
<tr>
<td></td>
<td>• This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• This course runs in Term 1. It begins in September.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS1830</td>
<td>Computing Laboratory (games)</td>
<td>1/2 unit</td>
<td>Term 2</td>
</tr>
<tr>
<td></td>
<td>• This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• This course runs in Term 2. It begins in January.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS1840</td>
<td>Internet Services</td>
<td>1/2 unit</td>
<td>Term 2</td>
</tr>
<tr>
<td></td>
<td>• This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• This course runs in Term 2. It begins in January.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS1860</td>
<td>Mathematical Structures</td>
<td>1/2 unit</td>
<td>Term 1</td>
</tr>
<tr>
<td></td>
<td>• This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• This course runs in Term 1. It begins in September.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS1870</td>
<td>Machine Fundamentals</td>
<td>1/2 unit</td>
<td>Term 2</td>
</tr>
<tr>
<td></td>
<td>• This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• This course runs in Term 2. It begins in January.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS1890</td>
<td>Software Design</td>
<td>1/2 unit</td>
<td>Term 1</td>
</tr>
<tr>
<td></td>
<td>• This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• This course runs in Term 1. It begins in September.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Level Two:

<table>
<thead>
<tr>
<th>Course</th>
<th>Description</th>
<th>Units</th>
<th>Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS2800</td>
<td>Software engineering - can only be taken if taking CS2810 in Term 2</td>
<td>1/2</td>
<td>Term 1</td>
</tr>
</tbody>
</table>
|          | - This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students  
|          | - This course runs in Term 1. It begins in September.                        |        |        |
|          | - Prerequisites: CS1801 or equivalent study                                  |        |        |

| CS2810   | Team Project - can only be taken if taking CS2800 in Term 1                 | 1/2    | Term 2 |
|          | - This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students  
|          | - This course runs in Term 2. It begins in January                          |        |        |
|          | - Prerequisites: CS1801, CS2800 or equivalent study                         |        |        |

| CS2821   | Systems Programming                                                        | 1/2    | Term 2 |
|          | - This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students  
|          | - This course runs in Term 2. It begins in January                          |        |        |

| CS2830   | Robotics                                                                    | 1/2    | Term 2 |
|          | - This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students  
|          | - This course runs in Term 2. It begins in January                          |        |        |
|          | - Prerequisites: CS1801, CS1820 equivalent study                            |        |        |

| CS2850   | Network Operating Systems                                                   | 1/2    | Term 1 |
|          | - This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students  
|          | - This course runs in Term 1. It begins in September.                        |        |        |
|          | - Prerequisites: CS1801 or equivalent study                                  |        |        |

| CS2855   | Databases                                                                   | 1/2    | Term 1 |
|          | - This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students  
|          | - This course runs in Term 1. It begins in September.                        |        |        |
### CS2860: Algorithms and Complexity
- **1/2 unit**
- **Term 2**
- This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students.
- This course runs in Term 2. It begins in January.

### IY2760: Introduction to Information Security
- **1/2 unit**
- **Term 1**
- This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students.
- This course runs in Term 1. It begins in September.

### IY2840: Computer and Network Security
- **1/2 unit**
- **Term 2**
- This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students.
- This course runs in Term 2. It begins in January.
- Prerequisites: CS2821 and IY2760 or equivalent.

### CS2846: Human-Computer Interaction
- **1/2 unit**
- **Term 2**
- This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students.
- This course runs in Term 2. It begins in January.
- Pre-requisite: CS1840 or equivalent.

## Level Three:

### CS3110: Bioinformatics
- **1/2 unit**
- **Term 2**
- This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students.
- This course runs in Term 2. It begins in January.

### CS3220: Fundamentals of Digital Sound and Music
- **1/2 unit**
- **Term 1**
- This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students.
- This course runs in Term 1. It begins in September.
- Prerequisites: CS2800 or equivalent study.
<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
<th>Credits</th>
<th>Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS3250</td>
<td>Data Visualisation and Exploratory Analysis</td>
<td>1/2 unit</td>
<td>Term 2</td>
</tr>
</tbody>
</table>
|             | • This course is **available** to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students  
|             | • This course runs in **Term 2. It begins in January.**                                    |
| CS3480      | Software Language Engineering                     | 1/2 unit| Term 2|
|             | • This course is **available** to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students  
|             | • This course runs in **Term 2. It begins in January.**                                       |
| CS3490      | Computational Optimisation                        | 1/2 unit| Term 2|
|             | • This course is **available** to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students  
|             | • This course runs in **Term 2. It begins in January.**                                       
|             | • Prerequisites: CS2860, CS2870 or equivalent study                                              |
| CS3510      | Functional Programming and Applications           | 1/2 unit| Term 1|
|             | • This course is **available** to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students  
|             | • This course runs in **Term 1. It begins in September.**                                      
|             | • Prerequisites: CS1801 or equivalent study                                                      |
| CS3580      | Advanced Data Communications                       | 1/2 unit| Term 1|
|             | • This course is **available** to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students  
|             | • This course runs in **Term 1. It begins in September.**                                      
|             | • Prerequisites: CS1840 or equivalent study                                                      |
| CS3750      | Concurrent and Parallel Programming                | 1/2 unit| Term 2|
|             | • This course is **available** to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students  
|             | • This course runs in **Term 2. It begins in January.**                                        
<p>|             | • Prerequisites: CS1801, CS2800 or equivalent study                                            |</p>
<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
<th>Unit</th>
<th>Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS3760</td>
<td>Information Security</td>
<td>1/2</td>
<td>Term 1</td>
</tr>
<tr>
<td></td>
<td>- This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- This course runs in Term 1. It begins in September.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS3920</td>
<td>Machine Learning</td>
<td>1/2</td>
<td>Term 1</td>
</tr>
<tr>
<td></td>
<td>- This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- This course runs in Term 1. It begins in September.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Prerequisites: A-level Mathematics or equivalent study</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS3940</td>
<td>Intelligent agents and multi-agent systems</td>
<td>1/2</td>
<td>Term 1</td>
</tr>
<tr>
<td></td>
<td>- This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- This course runs in Term 1. It begins in September.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Prerequisites: CS2820 or CS2821 or equivalent study</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IY3660</td>
<td>Applications of Cryptography</td>
<td>1/2</td>
<td>Term 1</td>
</tr>
<tr>
<td></td>
<td>- This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- This course runs in Term 1. It begins in September.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IY3840</td>
<td>Malicious Software</td>
<td>1/2</td>
<td>Term 2</td>
</tr>
<tr>
<td></td>
<td>- This course is available to all ERASMUS, INTERNATIONAL EXCHANGE and STUDY ABROAD students</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- This course runs in Term 2. It begins in January.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Prerequisites: CS1870, CS1840, CS2850 or equivalent study</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEPARTMENT OF: Computer Science</td>
<td>Academic Session: 2015-16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Course Code:</strong> CS1801</td>
<td><strong>Course Value:</strong> 1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Course Title:</strong> Object Oriented Programming</td>
<td><strong>Status:</strong> Compulsory</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Availability:</strong> Term 1 &amp; 2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This course can only be taken if both terms are attended.

**Learning Outcomes:**
By the end of this course a student should be able to:
- solve basic programming tasks
- understand and use basic object-oriented concepts
- appreciate the need for program documentation, testing, readability and modifiability

**Course Content:**
- *Program basics*: variables, types, scope, lifetimes
- *Control flow*: if-constructs, for-loops, while-loops
- *Data structures*: strings, sets, lists, trees, algorithms
- *Objects*: classes, dynamic dispatch, interface, inheritance, generics
- *Exceptions*: throwing and catching
- *File I/O*: streams, file reading, writing and copying

**Teaching & Learning Methods:**
Lecture based delivery, supported by laboratory classes and small group tutorials.
Normally 2 hours of lectures, 5 hours of laboratory classes and 1 tutorial hour per week.

**Details of teaching resource on Moodle:**
Course material including lecture notes will be on the course webpage.

**Key Bibliography:**

**Formative Assessment & Feedback:**
Small group tutorials in which students work through programming exercises under supervision, getting verbal feedback. A zero weighted test in the first term for which marks will be returned.

**Summative Assessment:**
- 90% invigilated, closed book  3 hours
- 10% coursework

Details of coursework submission deadlines will be published on the department website at the start of term

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
# COURSE SPECIFICATION FORM
for new course proposals and course amendments

<table>
<thead>
<tr>
<th>DEPARTMENT OF: Computer Science</th>
<th>Academic Session: 2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Course Code:</strong> CS1820</td>
<td><strong>Course Value:</strong> 0.5</td>
</tr>
<tr>
<td><strong>Status:</strong> Core</td>
<td><strong>Availability:</strong> Term 1</td>
</tr>
<tr>
<td><strong>Course Title:</strong> Computing laboratory (robotics)</td>
<td></td>
</tr>
</tbody>
</table>

# Aims:
To provide a practical approach to programming and the building of computer systems.
To introduce elementary robotics concepts
To reinforce the earning of elementary programming

# Learning Outcomes:
By the end of this course a student should be able to:

- Programme a mobile robot to execute pre-defined movements
- Understand the operation of basic sensors
- Understand the essentials of real time event-driven programming

# Course Content:
**Lego NXT system:** control brick capabilities, motors, sensor classes
**Sensor operation:** light sensor, ultrasonic sensors, pushbuttons
**Simple movement:** motors, gearing, idler wheels, tracked vehicles
**Actuators:** grippers, extensible arms

**Projects:** a sequence of scripted exercises, with opportunities for student led expansion

# Teaching & Learning Methods:
Teacher led laboratory classes and self-driven, scripted group project work.
Up to 4 hours of lectures and laboratory classes per week.

# Details of teaching resources on Moodle:
Course material including lecture notes will be on the course webpage including details of submission of worksheets, group project and individual report.

# Key Bibliography:

# Formative Assessment & Feedback:
Five formative worksheets
Verbal feedback on progress will be given during the laboratory sessions.

# Summative Assessment:
(100% Coursework )
individual report on group project coursework
Details of coursework submission deadlines will be published on the department website at the start of term

---

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s Policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
<table>
<thead>
<tr>
<th>DEPARTMENT OF: Computer Science</th>
<th>Academic Session: 2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Code: CS1830</td>
<td>Course Value: 0.5</td>
</tr>
<tr>
<td>Status: Core</td>
<td></td>
</tr>
<tr>
<td>Course Title: Computing laboratory (games)</td>
<td>Availability: Term 2</td>
</tr>
</tbody>
</table>

**Aims:**
To provide a practical approach to programming and the building of computer applications
To introduce elementary gaming concepts
To reinforce the earning of elementary programming

**Learning Outcomes:**
By the end of this course a student should be able to:
- Write simple games programmes
- Load and run single-player 2-D games
- Understand the essentials of real world simulation

**Course Content:**
- **Gaming graphics:** animation, shading, graphics engines
- **Gaming physics:** collision detection, ballistics
- **User experience:** classic game styles, scoring, statistics
- **Projects:** a sequence of scripted exercises, with opportunities for student led expansion

**Teaching & Learning Methods:**
Teacher led laboratory classes and self-driven, scripted project work.
Up to 4 hours of lectures and laboratory classes per week.

**Details of teaching resource on Moodle:**
Course material including lecture notes will be on the course webpage
Including details of submission of worksheets, group project and individual report.

**Key Bibliography:**
- Fletcher Dunn and Ian Barberry “3D Math Primer for Graphics and Game development
- “Essential Mathematics for Games and Interactive Applications” by James M. Van Verth and Lars M. Bishop
- Ian Millington – Artificial Intelligence for Games
- Ian Millington – Games Physics Engine Development
- David Eberly – Game Physics
- Christer Ericsson – Real Time Collision Detection
- Extra textbooks are available at the following web sites:
  - http://programarcadegames.com
  - http://inventwithpython.com

**Formative Assessment & Feedback:**
Practical skills assessment – in Lab assessment. Verbal feedback on progress will be given during the laboratory sessions.

**Summative Assessment:**
(100% Coursework)
- Written individual Programming individual Programming assignment Programming project group Project report individual

Details of coursework submission deadlines will be published on the department website at the start of term.
# COURSE SPECIFICATION FORM
for new course proposals and course amendments

<table>
<thead>
<tr>
<th>Department/School:</th>
<th>Computer Science</th>
<th>Academic Session:</th>
<th>2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Title:</td>
<td>Internet Services</td>
<td>Course Value:</td>
<td>0.5</td>
</tr>
<tr>
<td>Course Code:</td>
<td>CS1840</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Availability:</td>
<td>Term 2</td>
<td>Status:</td>
<td>Core</td>
</tr>
</tbody>
</table>

- **Aims:** To provide an introduction to internet technologies and their use in an increasingly e-centric industry

- **Learning Outcomes:** By the end of this course a student should be able to:
  - Understand and describe the client server model
  - Program both the client and server sides of a simple application
  - Use XML to construct Web pages
  - Understand the fundamentals of web services

- **Course Content:**
  - Internet Basics: protocols, delay and loss.
  - Web Technologies: HTML and related technologies
  - Scripting languages: Client and server side scripting
  - Web Services: Basic security concepts and their implications to networks

- **Teaching & Learning Methods:** Lecture based delivery, supported by laboratory classes, guided independent study
  - Up to 4 hours of lectures and laboratory classes per week

- **Details of teaching resources on Moodle:** Course material including lecture notes will be on the course Moodle page.


- **Formative Assessment & Feedback:** Verbal feedback on progress will be given during the weekly laboratory sessions

- **Summative Assessment:**
  - 10% individual pieces of course work
  - 90% Invigilated, closed book. 1.5 hours
  - Details of coursework submission deadlines will be published on the course website at the start of term

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department's policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
COURSE SPECIFICATION FORM

for new course proposals and course amendments

<table>
<thead>
<tr>
<th>DEPARTMENT OF: Computer Science</th>
<th>Academic Session: 2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Code: CS1860</td>
<td>Course Value: 0.5</td>
</tr>
<tr>
<td>Course Title: Mathematical structures</td>
<td>Status: Core</td>
</tr>
<tr>
<td>Availability: Term 1</td>
<td></td>
</tr>
</tbody>
</table>

Aims: To provide insights and skills in rigor and formal reasoning in a way that allows reasoning about behaviour, correctness and performance in a programming environment.

To provide basic knowledge of the formal structures for program data representation.

Learning Outcomes: By the end of this course a student should be able to:

- define and reason about sets, relations, functions and cardinality
- write and reason about recursive definitions and prove results by induction and contradiction
- represent and reason about problems using graphs
- use vectors and transformations to define and manipulate graphical objects
- have an understanding of basic probability and statistics suitable for use in studying artificial intelligence

Course Content: Sets: defining sets, logic notation, proofs by construction, counterexample and contradiction

Relations: relations, orderings, functions, bijections

Recursion: recursive definitions and induction, cardinality of infinite sets

Graph theory: graphs, trees and spanning trees, directed graphs

Vector spaces: vectors, transformations, bases, matrices, determinants

Probability: elementary and conditional probability, binomial distribution, random variables and Bayes theorem

Teaching & Learning Methods: Lecture based delivery, supported by small group tutorials.

Normally 3 hours of lectures per week.

Details of teaching resources on Moodle: Course material including lecture notes will be on the course webpage


Formative Assessment & Feedback: Individual discussion of non-assessed tutorial exercises and discussion of assessed worksheets in small group tutorials.

Summative Assessment: 90% invigilated, closed book 1.5hrs

10% coursework

Details of coursework submission deadlines will be published on the department website at the start of term

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
## Course Specification Form

*for new course proposals and course amendments*

<table>
<thead>
<tr>
<th>DEPARTMENT OF:</th>
<th>Computer Science</th>
<th>Academic Session:</th>
<th>2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Course Code:</strong></td>
<td>CS1870</td>
<td><strong>Course Value:</strong></td>
<td>0.5</td>
</tr>
<tr>
<td><strong>Course Title:</strong></td>
<td>Machine fundamentals</td>
<td><strong>Status:</strong></td>
<td>Core</td>
</tr>
<tr>
<td><strong>Availability:</strong></td>
<td>Term 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Aims:
- To explain the theory and use of logic in the description, specification and behaviour of machine processes.
- To provide insights and skills for dealing with large and infinite objects in a way that allows them to be implemented in a programming environment.

### Learning Outcomes:
By the end of this course a student should be able to:
- use formal logic to design, reason about and minimise switching circuits
- write basic programs in assembly language
- understand binary representations of signed and unsigned integers
- write regular expressions to describe sets and build deterministic automata to recognise these sets
- use automata to design and reason about sequential flow systems

### Course Content:
- **Numbers**: binary number systems, two's complement notation.
- **Logic**: propositions, logical formulae, truth tables and logical equivalences. Predicates, proofs and logical inference. Normal forms.
- **Networks**: series and parallel switching circuits, network minimisation.
- **Automata**: regular expressions, automata, Thompson’s construction and the subset construction. Push down automata. Turing machines and non-computability.
- **Low level languages**: assembly language programming.

### Teaching & Learning Methods:
Lecture based delivery, supported by laboratory classes and group tutorials. Normally 3 hours of lectures and laboratory classes per week.

### Details of teaching resources on Moodle:
Course material including lecture notes will be on the course webpage

### Key Bibliography:

### Formative Assessment & Feedback:
Individual discussion of non-assessed tutorial exercises and discussion of assessed worksheets in small group tutorials.

### Summative Assessment:
- 90% invigilated, closed book  
  1.5hrs
- 10% coursework
  1.0hrs
Details of coursework submission deadlines will be published on the department website at the start of term

---

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
**COURSE SPECIFICATION FORM**

*for new course proposals and course amendments*

<table>
<thead>
<tr>
<th>DEPARTMENT OF: Computer Science</th>
<th>Academic Session: 2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Code: CS1890</td>
<td>Course Value: 0.5</td>
</tr>
<tr>
<td>Course Title: Software Design</td>
<td>Status: Core</td>
</tr>
<tr>
<td>Availability: Term 1</td>
<td></td>
</tr>
</tbody>
</table>

**Aims:**

The practical experience of analysing, explaining and performing software aims to provide a foundation for the subsequent core course CS2800, Software Engineering, CS2810, Team project and CS2855 Data bases.

**Learning Outcomes:**

- Identify common software requirements and how these map to software components. You will be able to recognise how these requirements have been discharged in existing systems and critique their effectiveness.
- Understand several techniques and notations that make it possible to document software design. You will understand that Software Engineering supports communications of design ideas and this will allow you to see how software engineering is a team activity.
- Understand the importance of the several activities of a professional software engineer. This will include techniques from agile software development, but will focus on requirements acquisition and software design.
- Apply several techniques to design software based on user requirements. You will be able to judge the appropriateness of designs produced using these techniques, both formally and informally.
- Analyse and critique the design of existing software. This will include the user experience of software as a measure of its fitness for purposes.

**Course Content:**

The course addresses concepts required for performing software design activities. Interpreting requirements, identifying software components, documenting and understanding various stages of development.

This course will emphasise problem based learning. Students will progress through case studies in critiquing software, acquiring and expressing requirement, designing software and documenting their designs.

You will learn to use current industry standard tooling such as eclipse, UML. You will see how design is achieved in various current software engineering processes, including the waterfall and agile processes.

**Teaching & Learning Methods:**

Lecture based delivery, supported by laboratory classes

Normally 3-4 hours of lectures and laboratory classes per week.

**Details of teaching resources on Moodle:**

Course material including lecture notes will be on the course webpage.

**Key Bibliography:**


**Formative Assessment & Feedback:**

Oral feedback during lab sessions

**Summative Assessment:**

40% Invigilated closed book  1.5 hours
60% - coursework (several pieces) Group coursework Individual

Details of coursework submission deadlines will be published on the course website at the start of term.

---

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
**COURSE SPECIFICATION FORM**
for new course proposals and course amendments

<table>
<thead>
<tr>
<th>Department/School:</th>
<th>Computer Science</th>
<th>Academic Session:</th>
<th>2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Title:</td>
<td>Software engineering</td>
<td>Course Value:</td>
<td>0.5</td>
</tr>
<tr>
<td>Course Code:</td>
<td>CS2800</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Availability:</td>
<td>Term 1</td>
<td>Status:</td>
<td>Core PR</td>
</tr>
<tr>
<td>Pre-requisites:</td>
<td>CS1801 or equivalent</td>
<td>Recommended:</td>
<td></td>
</tr>
</tbody>
</table>

*Can only be registered for if taking CS2810 in term 2*

**Aims:**
Introducing software Engineering tools and techniques through practical experience of design and development that enable each individual programmer to contribute effective, working, documented code, as part of a team, in a timely fashion.

**Learning Outcomes:**
By the end of this course a student should be able to:
- understand the software engineering techniques and managerial discipline required to work as part of a team
- understand and use basic object-oriented concepts
- appreciate the need for program documentation, testing, readability and modifiability
- use appropriate tools to support software development: Version control, programming standards, a modern IDE,
- be able to use test driven development to deliver a small scale project.

**Course Content:**
- **Software engineering:** models of software development, planning,
- **Object oriented design:** Notation for design, identifying objects, classes, attributes and methods. Class relationships, design patterns
- **Programming methodologies:** program structure, style, and layout. Coding standards, test driven development
- **Testing:** Program analysis, black and white box testing, defensive programming, system, integration and acceptance testing.
- Use of a variety of modern software engineering tools: in particular version control, debugger, code style checkers, junit and how these can be integrated into an industry standard IDE.

**Teaching & Learning Methods:**
Lecture based delivery with class exercises and worksheets. Use of structured practice sessions.
Up to 4 hours of lectures per week.

**Details of teaching resources on Moodle:**
Course lecturer will use Moodle for demonstration videos, links to on line resources, topic based course content description, feedback and coursework submission. There will also be a question and answer forum for eliciting student feedback in an anonymous fashion.

**Key Bibliography:**

**Formative Assessment & Feedback:**
Feedback will be given based on questions on the Moodle forum and any common mistakes made in coursework.

**Summative Assessment:**
- 60% invigilated, closed book 2 hours
- 40% coursework

Details of coursework submission deadlines will be published on the department website at the start of term.

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
### COURSE SPECIFICATION FORM
for new course proposals and course amendments

<table>
<thead>
<tr>
<th>DEPARTMENT OF: Computer Science</th>
<th>Academic Session: 2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Code: CS2810</td>
<td>Course Value: 0.5</td>
</tr>
<tr>
<td>Course Title: Team Project</td>
<td>Status: Core PR</td>
</tr>
<tr>
<td>Prerequisites: CS1801, CS2800 or equivalent</td>
<td>Recommended:</td>
</tr>
<tr>
<td>Can only be registered for if taken CS2800 in term 1</td>
<td></td>
</tr>
</tbody>
</table>

#### Aims:
To appreciate the role of the computer professional through the practical experience of developing medium scale software as part of team.

#### Learning Outcomes:
By the end of this course a student should be able to:
- work in a team to produce a substantial product using software engineering techniques
- apply managerial discipline and a professional attitude.
- understand why project cost and effort is hard to estimate and why project quality is hard to prescribe.
- Design software to be reliable and secure

#### Course Content:
The software lifecycle: Models of software development, planning, documentation, costing, quality assurance, reliability and security.

Team development, communication, managing risks and conflicts.

Practical experience of standard industrial software engineering agile project management, use of version control in a team, IDEs, etc.

A medium scale agile software development project, conducted by teams of students

#### Teaching & Learning Methods:
Student led team work and team (sprint) meetings with the team supervisor acting as the commissioning user.

Sessions providing group based structured practical experience of software engineering methods and tools.

Up to 4 hours of lectures and laboratory sessions per week.

#### Details of teaching resources on Moodle:
All course material and practical session guides will be available on Moodle. Student groups will have access to group’s wikis and forums. A class forum will be used to allow timely feedback for any issues arising during the practical course.

#### Key Bibliography:

#### Formative Assessment & Feedback:
The project management system and version control system will provide a complete record of the project process and individual participation. Verbal feedback will be provided during the team supervisor meetings.

#### Summative Assessment:
- 40% invigilated, closed book  2 hours
- 60% coursework – group and individual

Details of coursework submission deadlines will be published on the department website at the start of term.

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
# Systems Programming

**Course Title:** Systems Programming  
**Course Code:** CS2821  
**Course Value:** 0.5  
**Recommended:**  
**Availability:** Term 2  
**Status:** Optional

**Prerequisites:** CS1801 or equivalent

## Aims:
To develop an understanding of the interaction between application software and the operating system.

## Learning Outcomes:
By the end of this course a student should be able to:
- Demonstrate a working understanding of program execution using a simplified model of the main memory hierarchy,  
- Explain the implementation of data structures at the level of memory references,  
- Be able to interact with the operating system via system calls,  
- Understand memory management, interrupts, sockets and basic threading in C.  
- Write simple shell scripts to automate the processing of batch jobs.

## Course Content:
- **Modelling the computer:** CPU/main memory backing store, the pigeon hole model of memory, indexing and pointer hopping, heap management.  
- **Performance:** abstracting performance, counting basic operations, impact of cache.  
- **Procedural languages:** dynamic memory handling in C and in comparison to Java; interrupts, pipes, sockets, file i/o, and starting processes in C.  
- **Scripting:** batch processing of simple shell commands, handling of command line arguments, filesystem navigation, loops and conditionals.

## Teaching & Learning Methods:
Lecture based delivery, supported by laboratory classes and tutorials.  
Minimally 3 hours of lectures and laboratory classes per week. Students may attend additional labs organised to deal with large class sizes.

## Details of teaching resources on Moodle:
Course material including lecture notes will be on the Moodle course page

## Key Bibliography:
- C.Abling, JP Vossen, C. Newham: Bash Cookbook, O'Reilly Media 2007  

## Formative Assessment & Feedback:
Written Assessment – solutions discussed during lectures and lab sessions. Written marks and comments on assessed exercises. Students encouraged to distribute and score each other’s work.

## Summative Assessment:
- **80% invigilated, closed book  2 hours**  
- **20% coursework**

Details of coursework submission deadlines will be published on the department website at the start of term.

---

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department's policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
## COURSE SPECIFICATION FORM

for new course proposals and course amendments

<table>
<thead>
<tr>
<th>DEPARTMENT OF: Computer Science</th>
<th>Academic Session: 2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Code: CS2830</td>
<td>Course Value: 0.5</td>
</tr>
<tr>
<td>Status: Option</td>
<td></td>
</tr>
<tr>
<td>Course Title: Robotics</td>
<td>Availability: Term 2</td>
</tr>
<tr>
<td>Prerequisites: CS1801, CS1820 or equivalent</td>
<td>Recommended:</td>
</tr>
</tbody>
</table>

**Aims:**
To introduce practical robotics
To introduce simple dynamic control systems

**Learning Outcomes:**
By the end of this course a student should be able to:
- implement mobile robots that construct simple real world maps
- understand the basics of PID feedback systems
- understand the principles of localisation
- program significant robotic application in Java on Lego NXT systems

**Course Content:**
- **Motion control:** line following, distance sensor balancing, dead reckoning
- **Sensors:** characterisation of sensors by dynamic range, sensitivity and repeatability
- **PID control:** proportional, integral and derivative terms, loop tuning
- **Maze solving and search:** wall following, Tremaux, heuristic search
- **Localsation:** Probabilistic interpretations of pose, Kalman filters and Particle Filters

**Teaching & Learning Methods:**
Lecture based delivery, supported by laboratory classes. Self guided project work. Normally 3 hours of lectures and laboratory classes per week.

**Details of teaching resources on Moodle:**
Course lecturer will use Moodle for everything including the submission of assessed report.

**Key Bibliography:**

**Formative Assessment & Feedback:**
Verbal feedback on progress will be given during the laboratory sessions.
Written feedback on projects reports and coursework.

**Summative Assessment:**
- 70% Invigilated, closed book 2 hours
- 30% Coursework

Details of coursework submission deadlines will be published on the course website at the start of term.

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
**COURSE SPECIFICATION FORM**  
*for new course proposals and course amendments*

<table>
<thead>
<tr>
<th>DEPARTMENT OF: Computer Science</th>
<th>Academic Session: 2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Course Code:</strong> CS2850</td>
<td><strong>Course Value:</strong> 0.5</td>
</tr>
<tr>
<td><strong>Status:</strong> Option</td>
<td></td>
</tr>
<tr>
<td><strong>Course Title:</strong> Operating Systems</td>
<td><strong>Availability:</strong> Term 1</td>
</tr>
<tr>
<td><strong>Prerequisites:</strong> CS1801 or equivalent</td>
<td><strong>Recommended:</strong></td>
</tr>
</tbody>
</table>

**Aims:** To introduce students to the principles of the function and architecture of network operating systems.

**Learning Outcomes:** By the end of this course a student should be able to:
- demonstrate an understanding of the principles of computer operating systems
- evaluate the theory and practice of an existing operating system
- write basic shell scripts
- understand the use of network services at operating systems level

**Course Content:**
- *Introductory topics*: role of an operating system, historical perspectives, computer architecture
- *Processes*: process management and scheduling, inter-process interaction, UNIX and Windows
- *Memory*: fixed and dynamic partitioning, swapping and paging, virtual memory, page replacement algorithms, implementation issues
- *File systems*: implementation and maintenance, file systems in UNIX and Windows, access control, security issues, distributed file systems, network management issues
- *UNIX shell*: starting programs, input and output streams, pipes, filters, UNIX utilities

**Teaching & Learning Methods:** Lecture based delivery, supported by practical sessions. Normally 3 hours of lectures and laboratory classes per week.

**Details of teaching resources on Moodle**

**Key Bibliography:**

**Formative Assessment & Feedback:** Written grades will be given for the assessed assignments.

**Summative Assessment:** 80% invigilated, closed book 2 hours
20% coursework
Details of coursework submission deadlines will be published on the department website at the start of term.

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
## Course Specification Form

for new course proposals and course amendments

<table>
<thead>
<tr>
<th>DEPARTMENT OF: Computer Science</th>
<th>Academic Session: 2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Code: CS2855</td>
<td>Course Value: 0.5</td>
</tr>
<tr>
<td>Status: Option</td>
<td></td>
</tr>
<tr>
<td>Course Title: Databases</td>
<td>Availability: Term 1</td>
</tr>
<tr>
<td>Prerequisites: CS1801 or equivalent</td>
<td>Recommended:</td>
</tr>
</tbody>
</table>

### Aims:
By the end of this course a student should be able to:
- issues involved in database design and the theory of the relational view of data
- describe the crucial issues concerning database integrity and recovery from failure
- write SQL queries
- be familiar with the steps for the design and implementation of a database, from the user specifications to the final design
- implement an interface to an SQL database using an API

### Learning Outcomes:
Data modelling; views, subschema, data dictionary, data independence, entity relationship model. T
The relational model: relations, attributes, domains, relational algebra.
Database design: normalisation, normal forms, entities and attributes SQL, basis SQL, correspondence between the relational model and SQL commands, simple queries, combination and sub queries administration and implementation, integrity, recovery from failure, concurrency, deletion and updating, forms, report writing, programmatic access to SQL databases.

### Course Content:
Lecture based delivery, supported by practical sessions.
Normally 4 hours of lectures and laboratory classes per week.

### Teaching & Learning Methods:
Course material including lecture notes will be on the Moodle course page

### Key Bibliography:

### Formative Assessment & Feedback:
Verbal feedback on progress will be given during the laboratory sessions. Written feedback coursework

### Summative Assessment:
- 60% invigilated, closed book 2 hours
- 40% coursework
Details of coursework submission deadlines will be published on the department website at the start of term.

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
**Course Specification Form**  
for new course proposals and course amendments

<table>
<thead>
<tr>
<th>Department/School:</th>
<th>Computer Science</th>
<th>Academic Session:</th>
<th>2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Title:</td>
<td>Algorithms and Complexity</td>
<td>Course Value:</td>
<td>0.5</td>
</tr>
<tr>
<td>Course Code:</td>
<td>CS2860</td>
<td>Availability:</td>
<td>Term 2</td>
</tr>
<tr>
<td>Pre-requisites:</td>
<td>CS1801, CS1860 or equivalent</td>
<td>Status:</td>
<td>Core</td>
</tr>
</tbody>
</table>

**Aims:**
To teach the design of algorithms and data structures from the point of view of time and space complexity.

**Learning Outcomes:**
By the end of this course a student should be able to:
- understand and reason about alternative data structure representations implement and reason about alternative implementations for basic algorithms
- calculate the complexity of basic algorithms

**Course Content:**
- Complexity: counting, big-O notation, best-case, worst-case and average-case analysis
- Sorting algorithms: implementation and analysis of bubble sort, insertion sort, merge sort, quick sort, heap sort
- Searching algorithms: implementation and analysis of linear search, binary search, binary search trees, hash coding
- Topics drawn from the following three areas:
  - Combinatorial optimisation: exhaustive search, heuristics, simulated annealing
  - Game theory: Min-max trees, alpha-beta trees
  - String theory: naive and Rabin-Karp string matching algorithm

**Teaching & Learning Methods:**
Lecture based delivery, supported by laboratory classes and problems sessions.

Normally 3 hours of lectures per week.

**Details of teaching resources on Moodle:**
Course lecturer will use Moodle for everything including the submission of worksheets, group project and individual report.

**Key Bibliography:**
  [http://www.cs.berkeley.edu/~vazirani/algorithms.html](http://www.cs.berkeley.edu/~vazirani/algorithms.html)
- Data Structures and Algorithms: K.Mehlhorn and P. Sanders.

**Formative Assessment & Feedback:**
Lab Based Assignments and In-Lecture assignments.
Verbal feedback will be given during the laboratory sessions and feedback provided during lectures.

**Summative Assessment:**
90% Exam  
10% Coursework

Details of coursework submission deadlines will be published on the course website at the start of term

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
**COURSE SPECIFICATION FORM**
for new course proposals and course amendments

<table>
<thead>
<tr>
<th>DEPARTMENT OF:</th>
<th>Computer Science</th>
<th>Academic Session: 2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Code:</td>
<td>IY2760</td>
<td>Course Value: 0.5</td>
</tr>
<tr>
<td>Status:</td>
<td></td>
<td>Option</td>
</tr>
<tr>
<td>Course Title:</td>
<td>Introduction to Information Security</td>
<td>Availability: Term 1</td>
</tr>
<tr>
<td>Prerequisites:</td>
<td>CS1801 or equivalent</td>
<td></td>
</tr>
</tbody>
</table>

**Aims:**

Application of computer technology and the successful management of information resources is fundamental to the success of business both now and in the future. The significance of information to commercial enterprises means that securing this information is of fundamental importance.

This course is concerned with providing a general introduction to the subject of Information Security as a whole. The key technologies are introduced and, after taking this course, the student will be equipped with the necessary background knowledge to go on and gain an in-depth understanding of specialised topics within the subject area.

**Learning Outcomes:**

On successful completion of this course, students will be able to:

- Demonstrate an understanding of the successful and secure management of information resources and obtain a detailed introduction to the subject of information security.
- Appreciate the different cryptographic algorithms, their use, advantages and disadvantages.
- Apply the above identified cryptographic primitives in the review and evaluation of cryptographic protocols.
- Identify, through the case studies how information security may be influenced by real world design and implementation decisions.
- Appreciate the rational decisions in the design of a number tokens and secure elements.
- Apply the back ground knowledge gained to Reuther study of the specialised topics with the subject area.

**Course Content:**

- *Elements of cryptography:* Ciphers (DES/AES), Message Authentication codes (MACs). Public key ciphers and digital signatures (RSA).
- *Identity verification:* use and storage of conventional passwords. Dynamic password schemes. Biometric techniques. Use of tokens (dumb and intelligent), including the use of secure elements such as smart cards and trusted execution environments (TEEs).
- *Access control:* Access Control Lists, capabilities, security labels (MAC and DAC), and role-based access control.
- *CASE STUDY I:* electronic payments (EMV). Examine the overall security functionality provided by widely utilised Europay-Mastercard-VISA (EMV) standard.
- *Network security concepts:* the concepts of security services and security mechanisms (as in ISO 7498-2) firewalls.
- Personal computer security: viruses, spyware, restricting access.
- *Authentication and key distribution:* The importance and relatedness of the concepts of key management and entity authentication in a network. Objectives of an entity authentication protocol. Some fundamental protocols (e.g. Kerberos). Using authentication protocols for key distribution, and other approaches to key establishment (including public key certificates and X.509).
- Cyber Physical security, Examine the security provisions, strengths and weaknesses of existing multi-application smart card platforms and operating systems along with the security of embedded systems and tokens.
- *CASE STUDY II:* Chip migration for financial institutions. Examine the relevant information security, business, design, architectural and other factors that may influence the adoption of chip card technology by financial institutions.

**Teaching & Learning Methods:**

3 hours of lectures and non-assessed coursework and private study.

The coursework is designed to allow students to apply their knowledge to applications with which they are familiar.

**Details of teaching resources on Moodle:**

Course material including lecture notes will be on the Moodle course page.

**Key Bibliography:**

- N. Ferguson and B. Schneier, practical cryptography, 2003

**Formative Assessment & Feedback:**

This course will contain mandatory non-assessed coursework. Answers will be marked and returned with feedback.

**Summative Assessment:**

- 100% invigilated, closed book 2 hours

Details of coursework submission deadlines will be published on the department website at the start of term.

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
### COURSE SPECIFICATION FORM
for new course proposals and course amendments

<table>
<thead>
<tr>
<th>DEPARTMENT OF:</th>
<th>Computer Science</th>
<th>Academic Session:</th>
<th>2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Code:</td>
<td>IY2840</td>
<td>Course Value:</td>
<td>0.5</td>
</tr>
<tr>
<td>Status:</td>
<td>Option</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Course Title:</td>
<td>Computer and network security</td>
<td>Availability:</td>
<td>Term 2</td>
</tr>
<tr>
<td>Prerequisites:</td>
<td>CS1801 - or equivalent, IY2760</td>
<td><strong>Can only be registered for if taken IY2760 in term 1</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Aims:**
This course is intended to provide a detailed exposition of computer and network security, which will build on knowledge acquired on previous computer science courses.

**Learning Outcomes:**
- Identify and exploit the software vulnerabilities that can be introduced into programs through language features and poor programming practice
- Discuss the countermeasures that can mitigate the exploitation of such software vulnerabilities
- Introduce (briefly) malicious software (malware) as a typical consequence of a successful software exploitation
- Provide pointers to/discuss academic and/or industry research-oriented publications on the subject

**Course Content:**
- Software vulnerabilities and hands on hacking-oriented attacks
- Memory errors
- Web
- Network
- Countermeasures
- Pointers to research papers

**Teaching & Learning Methods:**
Lecture based delivery, guided independent study
Normally 3 hours of lectures per week

**Details of teaching resources on Moodle:**
Course material including lecture notes will be on the Moodle course page

**Key Bibliography:**
- Hacking: The art of exploitation, 2nd edition, Jon Erickson
- IP spoofing demystified, by route/daemon9, Phrack (http://www.phrack.org/issues.html?issue=48&id=14&mode=txt)

**Formative Assessment & Feedback:**
Written feedback based on summative coursework

**Summative Assessment:**
- **60% invigilated, closed book 2 hours**
- **40% coursework (report and code deriving from hands on programming challenges)**

Details of coursework submission deadlines will be published on the department website at the start of term

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
### COURSE SPECIFICATION FORM
for new course proposals and course amendments

<table>
<thead>
<tr>
<th>DEPARTMENT OF:</th>
<th>Computer Science</th>
<th>Academic Session: 2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Code:</td>
<td>CSxxxx</td>
<td>Course Value: 0.5</td>
</tr>
<tr>
<td>Status:</td>
<td>Option</td>
<td></td>
</tr>
<tr>
<td>Course Title:</td>
<td>Human-Computer-interaction</td>
<td>Availability: Term 2</td>
</tr>
<tr>
<td>Prerequisites:</td>
<td>CS1801 or equivalent</td>
<td></td>
</tr>
</tbody>
</table>

This course is still under validation – so course details are still under construction. No details available at the time.

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
## COURSE SPECIFICATION FORM
for new course proposals and course amendments

<table>
<thead>
<tr>
<th>Department/School:</th>
<th>Computer Science</th>
<th>Academic Session:</th>
<th>2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Title:</td>
<td>Bioinformatics</td>
<td>Course Value:</td>
<td>0.5</td>
</tr>
<tr>
<td>Course Code:</td>
<td>CS3110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Availability:</td>
<td>Term 2</td>
<td>Status:</td>
<td>Optional</td>
</tr>
<tr>
<td>Pre-requisites:</td>
<td>CS1801 or equivalent</td>
<td>Co-requisites:</td>
<td>None</td>
</tr>
</tbody>
</table>

### Aims:
To introduce the main approaches currently in use in bioinformatics, with special emphasis on the analysis of DNA and protein sequences emerging from genome sequencing projects and genome-wide experimental assays.

### Learning Outcomes:
By the end of this course a student should be able to:
- assess the main approaches currently in use in bioinformatics;
- demonstrate an understanding of the analysis of DNA and protein sequences.

### Course Content:
Basic molecular biology: introduction to the basic components of living cells, their functions and interactions, and to other concepts essential to understanding the use of computers in biology Sequence alignments, substitution matrices. Phylogenetic trees Dynamic Programming Systems biology: gene expression analysis, Protein-Protein Interaction analysis, biological networks, clustering

### Teaching & Learning Methods:
Lecture based delivery. Normally 3 hours of lectures per week.

### Details of teaching resources on Moodle:
A dedicated course page will be available for this course on Moodle.

### Key Bibliography:
- Understanding Bioinformatics Marketa Zvalebil, Jeremy O. Baum Garland Science, 2007

### Formative Assessment & Feedback:
Lectures in which students work through exercises getting verbal feedback.

### Summative Assessment:
- 90% Invigilated, closed book 2 hours
- 10% Coursework

Details of coursework submission deadlines will be published on the course website at the start of term

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department's policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
**COURSE SPECIFICATION FORM**

for new course proposals and course amendments

<table>
<thead>
<tr>
<th>DEPARTMENT OF: Computer Science</th>
<th>Academic Session: 2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Course Code:</strong> CS3220</td>
<td><strong>Course Value:</strong> 0.5</td>
</tr>
<tr>
<td><strong>Course Title:</strong> Fundamentals of Digital Sound and Music</td>
<td><strong>Status:</strong> Option</td>
</tr>
<tr>
<td><strong>Prerequisites:</strong> CS1801 , CS2800 or equivalent</td>
<td><strong>Availability:</strong> Term 1</td>
</tr>
</tbody>
</table>

| **Aims:** | To give students an understanding of the fundamental computational ideas relating to the processing of sound and music on a computer, from the basic mathematics underlying signal processing through to the use of up-to-date technological solutions to the problems of storing, synthesizing and analyzing speech and music. |
| **Learning Outcomes:** | By the end of this course a student should be able to: |
| | • demonstrate an understanding of lossy compression algorithms for sound and music |
| | • use Fourier analysis to synthesize and transform sounds in the frequency domain |
| | • understand the MIDI format, related JAVA APIs, and write a synthesizer from scratch |
| | • understand the basic acoustic properties of speech and standard techniques for speech synthesis and recognition |
| **Course Content:** | **Digital Signal Processing for Sound:** spectral analysis, Fourier transforms, Shannon-Nyquist theorem, phone, CD and SACD bandwidth. |
| | **Compression algorithms:** lossless compression, MP3, Ogg-vorbis, perceptual coding. |
| | **Music Technology:** sequencing and synthesis, MIDI format, Java API, sequences, digital instruments and synthesis of instrumental sounds, interfaces |
| | **Signal Processing:** digital post-processing and effects. |
| | **Harmony:** pythagorean harmony, temperaments, harmony, Helmholtz theory of consonance. |
| | **Speech:** acoustic properties of speech, spectrograms, phonetics, vowel formants, stress assignment, text to speech systems, automatic speech recognition using HMMs, language models, decoders and acoustic models. |
| | **Labs:** frequency domain representations of sounds, digital signal processing, sound-effects |
| **Teaching & Learning Methods:** | Lecture based delivery, supported by laboratory classes. |
| | Up to 4 hours of lectures and laboratory classes per week. |
| **Details of teaching resources on Moodle:** | Course lecturer will use Moodle for everything including the submission of assignments, Digital audio programming project |
| **Formative Assessment & Feedback:** | Return of marked zero-weighted assignments, verbal feedback on project work during the laboratory sessions. |
| **Summative Assessment:** | 80% invigilated, closed book  2 hours |
| | 20% coursework |
| | Details of coursework submission deadlines will be published on the department website at the start of term |

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
DEPARTMENT OF: Computer Science  

<table>
<thead>
<tr>
<th>Course Code:</th>
<th>Course Value:</th>
<th>Status:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS3250</td>
<td>0.5</td>
<td>Option</td>
</tr>
</tbody>
</table>

Course Title: Visualisation and Exploratory Analysis  
Prerequisites: CS1801 or or equivalent  

**Aims:** The course aims to teach the principles and arts of statistical visualisation and exploratory analysis of data.

**Learning Outcomes:** By the end of the course students should be able to:

- Have some of the skills and wisdom that are needed for open-ended exploratory analysis of data, and for the analytical presentation of the results of statistical analyses.
- Understand and be able to construct linear projections of multivariate data; they should also have experience in using some non-linear dimension reduction methods.
- Have some practical experience of using standard graph visualisation methods.
- Be aware of the dangers of data snooping, through practical experience of this.
- Be able to make reasoned choices in representational mode, glyph design, and colour design for presentation graphics.

**Course Content:**

- Construction of informative bivariate plots.
- Visualisation of multivariate data.
- Dimensional reduction. Non-linear methods (t-SME, isomap, Proxigrams).
- Exploratory cluster analysis.
- Standard methods for visualisation of relational and graph data (Gephi).
- Importance of guarding against “snooping”.
- Basic principles of colour scale design and glyph choice.
- Use of a modern visualisation package D3.

**Teaching & Learning Methods:**

Lecture based delivery, laboratory classes, guided independent study. Normally 3 hours of lectures / laboratory per week.

**Details of teaching resources on Moodle:** Course lecturer will use Moodle.

**Key Bibliography:**

- Visualising data, by William S. Cleveland.
- The elements of graphing data by William S. Cleveland.
- The visual display of quantitative data, by Edward Tufte.
- Envisioning information by Edward Tufte.

**Formative Assessment & Feedback:**

- Individual written feedback on Bivariate plotting and data transformation.
- Individual written feedback on multivariate and graph visualisation, and on using D3.

**Summative Assessment:**

80% invigilated, closed book 2 hours  
20% coursework  
Details of coursework submission deadlines will be published on the department website at the start of term.

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
### COURSE SPECIFICATION FORM

for new course proposals and course amendments

<table>
<thead>
<tr>
<th>Department/School:</th>
<th>Computer Science</th>
<th>Academic session:</th>
<th>2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Title:</td>
<td>Software language engineering</td>
<td>Course Value:</td>
<td>0.5</td>
</tr>
<tr>
<td>Course Code:</td>
<td>CS3480</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Availability:</td>
<td>Term 2</td>
<td>Status:</td>
<td>Option</td>
</tr>
<tr>
<td>Pre-requisites:</td>
<td>CS1801, CS2800 and CS2810 or equivalent</td>
<td>Co-requisites:</td>
<td></td>
</tr>
</tbody>
</table>

**Aims:**

This course is about models of entire domains, or meta-models. In this view, a model representing an application is an instance of a class of meta-models. One way of thinking about this is that models are written in a domain specific language; a programming language tailored to a particular area.

This course compliments course CS3470 compiler theory which takes a more formal view of language processor implementation. Neither course is a co-requisite for the other, but students interested in compilers, code generation and modelling would benefit from pursuing both courses.

**Learning Outcomes:**

- Understand meta-modelling and domain specific language design and implementation.
- Learn the significance of DSL’s and meta-modelling in advanced software engineering design and implement small specialised languages.

**Course Content:**

- Domains
- Meta-modelling
- Domain Specific Languages
- Concrete and abstract syntax, generalised vs near deterministic parsing
- Code generation
- The eclipse modelling framework
- Case studies

**Teaching & Learning Methods:**

- Formal lectures; supervised laboratories; unsupervised exercises, private study.

**Details of teaching resources on moodle**

- Course lecturer will use Moodle

**Key Bibliography:**

- Interest in this area in both software engineering and programming language research is rapidly expanding, and the course will draw on research papers from conferences including SLE (Software Language Engineering) and LDTA (Language Descriptions, Tools and Applications), as well as the following texts.

**Formative Assessment & feedback:**

- Written feedback on two mandatory non-assessed worksheets

**Summative Assessment:**

- 80% invigilated, closed book 2 hours
- 20% coursework

- Details of coursework submission deadlines will be published on the department website at the start of term

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
**COURSE SPECIFICATION FORM**

for new course proposals and course amendments

<table>
<thead>
<tr>
<th>Department/School:</th>
<th>Computer Science</th>
<th>Academic Session:</th>
<th>2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Title:</td>
<td>Computational Optimisation</td>
<td>Course Value:</td>
<td>0.5</td>
</tr>
<tr>
<td>Course Code:</td>
<td>CS3490</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Availability:</td>
<td>Term 2</td>
<td>Status:</td>
<td>Optional</td>
</tr>
<tr>
<td>Pre-requisites:</td>
<td>CS1801, CS2860, CS2870 or equivalent</td>
<td>Co-requisites:</td>
<td></td>
</tr>
</tbody>
</table>

**Aims:** To introduce the basic models of computational optimisation and the basic algorithms for solving computational optimisation problems. To demonstrate the theoretical and computational methods of analysing computational optimisation algorithms and will discuss available software packages for solving problems.

**Learning Outcomes:** By the end of this course a student should be able to:
- demonstrate an understanding of the basic models of computational optimisation
- apply the basic algorithms for solving computational optimisation problems
- evaluate theoretical and computational methods of analysing computational optimisation algorithms
- use enhanced algorithmic and mathematical skills.

**Course Content:**

- **Introduction:** algorithm efficiency and problem complexity
- **Linear programming (LP):** LP model, formulating problems as LP problems, graphical solution, simplex method, duality in LP, decomposition of LP problems, LP software
- **Integer Programming (IP):** IP models, branch-and-bound algorithm
- **Computational optimisation problems:** greedy-type algorithms, construction heuristics and local search for the TSP
- **Heuristics:** DMERN problem and signed graphs; heuristics for DMERN, experimental analysis of embedded network and TSP heuristics, theoretical analysis of heuristics, meta-heuristics

**Teaching & Learning Methods:** Lecture based delivery. Normally 3 hours of lectures per week plus independent guided study

**Details of teaching resources on Moodle:** The course is available with course slides and other information via Moodle

**Key Bibliography:**


**Formative Assessment & Feedback:** Return of marked zero-weighted assignments. Some solutions discussed in class.

**Summative Assessment:** 100% invigilated, closed book 2 hours

Details of coursework submission deadlines will be published on the department website at the start of term.

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
# COURSE SPECIFICATION FORM

for new course proposals and course amendments

<table>
<thead>
<tr>
<th>Department/School:</th>
<th>Computer Science</th>
<th>With effect from Academic Session:</th>
<th>2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Title:</td>
<td>Functional Programming and Applications</td>
<td>Course Value:</td>
<td>0.5</td>
</tr>
<tr>
<td>Course Code:</td>
<td>CS3510</td>
<td>Status:</td>
<td>Optional</td>
</tr>
<tr>
<td>Availability:</td>
<td>Term 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pre-requisites:</td>
<td>CS1801 or equivalent</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Aims:**
This course introduces the principles of functional programming (FP). Functional programming has become more and more popular in recent years because it promotes code that’s safe, concise, and elegant, and makes it easier to write parallel code for parallel computing. You will also study how FP may be combined with other programming paradigms (eg, OOP) in languages such as Scala, a language that fuses FP and OOP in a practical package for applications.

**Learning Outcomes:**
By the end of the course students should be able to:
- Understand the basic principles of functional programming; its basic ideas, foundations, advantages as compared with imperative programming paradigm
- Master the fundamental programming concepts such as recursion, abstraction, higher-order functions and data types
- Grasp skills in developing elegant, efficient, correct, and reusable programs, using the key features in functional programming languages
- Understand the advantages and disadvantages of different programming paradigms as compared to each other and, on the basis of this, the basic ideas of combining different programming styles in a uniform paradigm
- Understand the basic foundational principles of functional programming and some of the advanced issues such as genericity, polymorphism and typing in functional programming

**Course Content:**
Introduction to the basic principles of functional programming (FP) as compared to the imperative object-oriented programming such as Java
Some basic features in an FP language: recursion, abstraction and higher-order functions
Basic typing and data types such as lists and tuples
High-order functions and pattern matching
Classes and objects in a hybrid language (such as Scala)
Functional paradigm as compared with the imperative paradigm
Applications (eg. In parallel programming)
Basics in some advanced topics such as genericity, monads and dependent typing

**Teaching & Learning Methods:**
Normally 3 hrs per week of lectures and practicals.

**Details of teaching resources on Moodle:**
Course lecturer will use Moodle except coursework submission.

**Key Bibliography:**
C.S. Horstmann. Scala for the Impatient. Addison-Wesley 2012

**Formative Assessment & Feedback:**
Written assignment with written feedback

**Summative Assessment:**
80% invigilated, closed book 2 hours
20% coursework
Details of coursework submission deadlines will be published on the department website at the start of term

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
# Advanced Data Communications

**Course Code:** CS3580  
**Course Value:** 0.5  
**Status:** Option  
**Course Title:** Advanced Data Communications  
**Availability:** Term 1  
**Prerequisites:** CS1801, CS1840 or equivalent

## Aims:
To cover a range of important topics in modern data communications including text, image, audio and video transfer over networking infrastructures and recent trends in network management and network security.

## Learning Outcomes:
By the end of this course a student should be able to:
- explain how multimedia communications work
- describe coding and compressing algorithms for text, image and video
- describe Internet technologies in terms of supporting QoS
- demonstrate an understanding of most common security terms and concepts
- describe modern network security mechanisms and their applications
- explain Internet network management issues and possible solutions

## Course Content:
*Multimedia communications:* multimedia information representation, coding and compression, applications and standards, quality of service (QoS) requirements.  
*Internet:* IP addressing, routeing algorithms and routeing protocols, RIP, OSPF, the Internet multicast model, scheduling and queue management.  
*Security in communication networks:* security issues, security mechanisms, secure protocols.  
*Network management:* network management issues, infrastructure and framework for Internet management.

## Teaching & Learning Methods:
Lecture based delivery. Normally 3 hours of lectures per week.

## Key Bibliography:
- Fred Halsall: Multimedia Communications: applications, networks, protocols and standards, Addison-Wesley

## Details of teaching resources on Moodle:
Course material including lecture notes will be on Moodle

## Formative Assessment & Feedback:
Return of marked zero-weighted assignments.

## Summative Assessment:
- **100% Invigilated, closed book 2 hours**
  Details of coursework submission deadlines will be published on the department website at the start of term

---

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
## Course Specification Form

**DEPARTMENT OF:** Computer Science  
**Academic Session:** 2015-16  
**Course Code:** CS3750  
**Status:** Option  
**Course Title:** Concurrent and Parallel Programming  
**Course Value:** 0.5  
**Availability:** Term 2  
**Prerequisites:** CS1801, CS2800 or equivalent  

| **Aims:** | To introduce the principles and practical implementations of concurrent programming, to implement algorithms in a concurrent fashion, and to give an understanding of the variety of different concurrent architectures available. |
| **Learning Outcomes:** | By the end of this course a student should be able to:  
- understand the theoretical underpinnings of concurrent programming  
- understand deadlock and race conditions in shared states  
- understand threaded and message passing paradigms and be familiar with writing software using these paradigms  
- implement a variety of different algorithms concurrently and understand how their performance varies  
- describe wide variety of concurrent platforms, ranging from multi-core processors to PC farms to more widely distributed computing |
| **Course Content:** | CSP: events, processes, choice, synchronisation, traces, transition diagrams, interleaving, specification.  
*Behaviour of processes*: explosion of possible states in concurrent processes in comparison to sequential programming, unexpected traces, deadlock.  
*Java and CSP*: JCSP, implementing basic ideas of CSP in JCSP.  
*Threading paradigm*: implementation in Java, launching threads, shared states and race conditions, locking, conditional locking and deadlock.  
*Message Passing paradigm*: implementation in MPI, synchronization, broadcasting, one-to-one communication, synchronous and asynchronous communication.  
*Algorithms*: search, sort, graphs and dynamic programming implemented concurrently, limitations of concurrency.  
*Architectures*: SIMD and MIMD, shared and distributed memory. Hyper-Threading, multi-core processors and supercomputing. PC farms and Beowulf clusters. Distributed computing |
| **Teaching & Learning Methods:** | Lecture based delivery, with laboratory classes, guided independent study  
Normally 3 hours of lectures per week. |
| **Details of teaching resources on Moodle:** | Course Lecturer will use Moodle for everything except the submission of coursework |
| **Formative Assessment & Feedback:** | Verbal feedback on progress will be given during the laboratory sessions  
Written feedback from summative assessment  
Written feedback from mandatory formative assessment |
| **Summative Assessment:** | 80% Invigilated, closed book 2 hours  
20% Coursework (3 pieces of coursework)  
Details of coursework submission deadlines will be published on the course website at the start of term |

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
### DEPARTMENT OF: Computer Science

<table>
<thead>
<tr>
<th>Course Code:</th>
<th>CS3760</th>
<th>Course Value:</th>
<th>0.5</th>
<th>Status:</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Title:</td>
<td>Information Security</td>
<td>Availability:</td>
<td>Term 1</td>
<td>Recommended:</td>
<td></td>
</tr>
</tbody>
</table>

| Prerequisites: | CS1801 or equivalent | Can’t be registered for this course if registering for IY2760 |


| Teaching & Learning Methods: | Lecture based delivery. Normally 3 hours of lectures per week. |


| Details of teaching resources on Moodle: | Course material including lecture notes will be on Moodle |

| Formative Assessment & Feedback: | Return of marked zero-weighted exercises. Some solutions will be discussed in class |

| Summative Assessment: | 100% invigilated, closed book 2 hours |

Details of coursework submission deadlines will be published on the department website at the start of term.

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
COURSE SPECIFICATION FORM
for new course proposals and course amendments

<table>
<thead>
<tr>
<th>Department/School:</th>
<th>Computer Science</th>
<th>Academic Session:</th>
<th>2015-16</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Course Title:</strong></td>
<td>Machine Learning</td>
<td><strong>Course Value:</strong></td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(UG courses = unit value, PG courses = notional learning hours)</td>
<td></td>
</tr>
<tr>
<td><strong>Course Code:</strong></td>
<td>CS3920</td>
<td><strong>Availability:</strong></td>
<td>Term 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Status:</strong></td>
<td>Optional</td>
</tr>
<tr>
<td><strong>Pre-requisites:</strong></td>
<td>CS1801 or equivalent A-level Mathematics</td>
<td><strong>Co-requisites:</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Aims:** The course will teach main ideas of machine learning with a particular emphasis on kernel methods.

**Learning Outcomes:** By the end of this course a student should be able to:

- demonstrate knowledge of the theoretical background in machine learning methods
- have a basic understanding of the main advantages and limitations of various approaches to machine learning and specific machine-learning algorithms
- be able to implement basic versions of several machine-learning algorithms
- have a basic understanding of some ways to apply the ideas and algorithms of machine learning in industry, medicine and other fields

**Course Content:** Nearest neighbours for classification and regression: interesting distances
Discriminant analysis
Ridge regression and Lasso
Support vector machines for classification and regression
Kernel trick and its applications to the algorithms covered so far
Practical useful kernels, including string kernels.

**Teaching & Learning Methods:** Lecture based delivery, supported by laboratory classes.
Normally 3-4 hours of lectures/labs per week. Guided independent study

**Details of teaching resources on Moodle:** Course Lecturer will use Moodle except the submission of coursework.

**Key Bibliography:**

**Formative Assessment & Feedback:** In lectures / labs, Immediate help and verbal feedback

**Summative Assessment:**
80% Invigilated, closed book 2 hours – Calculators are permitted
20% Coursework – individual programming & Programming/theoretical questions
Details of coursework submission deadlines will be published on the course website at the start of term

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
# Course Specification Form

**Department of:** Computer Science  
**Academic Session:** 2015-16

<table>
<thead>
<tr>
<th>Course Code:</th>
<th>CS3940</th>
<th>Course Value:</th>
<th>0.5</th>
<th>Status:</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Title:</td>
<td>Intelligent agents and multi-agent systems</td>
<td>Availability:</td>
<td>Term 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prerequisites:</td>
<td>CS1801 or equivalent CS2821</td>
<td>Recommended:</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Aims:
To introduce the student to the concept and design of an agent and multi-agent system, and the main applications for which they are appropriate. To introduce a contemporary platform for implementing agents and multi-agent systems.

## Learning Outcomes:
By the end of this course a student should be able to:
- understand the notion of an agent and understand the characteristics of applications that lend themselves to an agent-oriented solution
- understand the key issues associated with constructing agents capable of intelligent autonomous action
- understand the key issues in designing societies of agents that can effectively cooperate in order to solve problems
- understand the main application areas of agent-based solutions, and be able to develop a meaningful agent-based system

## Course Content:
*Introduction:* agents and objects, expert systems, distributed systems; typical application areas for agent systems.  
*Intelligent Agents:* abstract architectures for agents; tasks for agents, the design of intelligent agents - reasoning agents, agents as reactive systems; hybrid agents (e.g., PRS); layered agents (e.g., Interrap).  
*Multi-Agent Systems:* classifying multi-agent interactions - cooperative versus non-cooperative; zero-sum and other interactions; cooperation - the Prisoner’s dilemma and Axelrod’s experiments; interactions between self-interested agents: auctions systems; negotiation; argumentation; interaction languages and protocols: speech acts, KQML/KIF, the FIPA framework, ontologies, coordination languages; interactions between benevolent agents: cooperative distributed problem solving (CDPS), partial global planning; coherence and coordination; applications of intelligent agents and multi-agent systems.

## Teaching & Learning Methods:
Lecture based delivery, supported by tutorial sessions. Normally 3 hours of lectures per week.

## Key Bibliography:

## Details of teaching resources on Moodle:
Course lecturer will use moodle for everything including the submission of assignments 1 & 2

## Formative Assessment & Feedback:
Verbal feedback provided through the tutorial sessions.

## Summative Assessment:
- 70% Invigilated, closed book 2 hours  
- 30% Coursework  
Details of coursework submission deadlines will be published on the course website at the start of term

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
**DEPARTMENT OF:** Computer Science – ISG group  
**Academic Session:** 2015-16

<table>
<thead>
<tr>
<th>Course Code:</th>
<th>IY3660</th>
<th><strong>Course Value:</strong></th>
<th>0.5</th>
<th><strong>Status:</strong></th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Course Title:</strong></td>
<td>Applications of Cryptography</td>
<td><strong>Availability:</strong></td>
<td>Term 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Prerequisites:</strong></td>
<td>Cs1801 or equivalent</td>
<td><strong>Recommended:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Aims:**  
The aim of this module is to explain the fundamentals behind cryptography and how it is deployed in real systems. Topics that will be explored include:

- Appreciate the full range of security services that can be provided by cryptography
- Compare a number of different cryptographic mechanisms that can be used to provide these security services
- Explain the full nature of the architecture within which cryptography is deployed in a security system
- Identify appropriate cryptographic standards that should be followed when implementing cryptography
- Explain the rationale for the design decisions taken in a number of widely deployed cryptographic systems

**Learning Outcomes:**

- Fundamentals: security services, security models, basic attacks on cryptosystems
- Cryptographic mechanisms: symmetric and public-key encryption, hash functions, MACs, digital signatures, authentication protocols
- Real world cryptography: key management, implementation issues, cryptographic standards, crypto politics
- Applications: case studies of systems such as 3G, EMV, SSL/TLS

**Course Content:**

- Lecture based delivery, supported by Laboratory sessions.
- Normally 3 hours of lectures/ Labs per week.

**Key Bibliography:**

- D.R. Stinson, Cryptography: Theory and Practice, CRC Press
- K.M. Martin: Everyday Cryptography, OUP
- C. Paar and J. Pelzl, Understanding Cryptography, Springer
- J. Katz and Y. Lindell, Introduction to Modern Cryptography, CRC Press
- N. Ferguson, B. Schneier and T. Kohno, Cryptography Engineering, Wiley

**Details of teaching resources on Moodle:**

Course lecturer will use moodle

**Formative Assessment & Feedback:**

Written and verbal comments on assignments

**Summative Assessment:**

- 70% Invigilated, closed book 2 hours
- 30% Coursework-Report

Details of coursework submission deadlines will be published on the course website at the start of term

---

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.
## Course Specification Form

**DEPARTMENT OF:** Computer Science – ISG group  

**Course Code:** IY3840  
**Course Value:** 0.5  
**Status:** Option  

**Course Title:** Malicious Software  
**Availability:** Term 1

**Prerequisites:** CS1801, CS2850, CS1840, CS1870 or equivalent

### Aims:
Cybercrime has become both more widespread and harder to battle. Researchers and anecdotal experience show that the cybercrime scene is becoming increasingly organized and consolidated, with strong links also to traditional criminal networks. Modern attacks are indeed stealthy and often profit oriented. Malicious software (malware) is the traditional way in which cybercriminals infect user and enterprise hosts to gain access to their private, financial, and intellectual property data. Once stolen, such information can enable more sophisticated attacks, generate illegal revenue, and allow for cyber-espionage.

### Learning Outcomes:
- Understand what malicious software is
- Understand the malware landscape of the early days and what efforts and challenges the AV industry was facing to fight malware threats; (basic) static analysis to analyze and detect malware; (basic) assembly and reverse engineering notions are provided
- Understand of dynamic analysis. Packing and algorithmic-agnostic unpacking is introduced as an initial step toward full dynamic analysis; sandboxes and limits of dynamic analysis and sandboxes.
- Understand the underground economy and cybercrime that surrounds malicious software
- Have a critical appreciation of some of the newer research trends that are likely to influence software security work in the coming years

### Course Content:
By mixing a practical, hands-on approach with the theory and techniques behind the scene, the course discusses the current academic and underground research in the field, trying to answer the foremost question about malware and underground economy, namely, "Should we care?"

Students will learn how traditional and mobile malware work, how they are analyzed and detected, peering through the underground ecosystem that drives this profitable but illegal business. Understanding how malware operates is of paramount importance to form knowledgeable experts, teachers, researchers, and practitioners able to fight back. Besides, it allows us to gather intimate knowledge of the systems and the threats, which is a necessary step to successfully devise novel, effective, and practical mitigation techniques.

### Teaching & Learning Methods:
Lecture based delivery, supported by Laboratory sessions.

Normally 3 hours of lectures/ Labs per week.

### Key Bibliography:
Details will be given at the beginning of the term if any are required.

### Details of teaching resources on Moodle:
Course lecturer will use moodle

### Formative Assessment & Feedback:
Verbal and written comments on Hands on programming challenges / coursework

### Summative Assessment:
60% Invigilated, closed book 3 hours
40% Hands on programming challenges / coursework

Details of coursework submission deadlines will be published on the course website at the start of term

The information contained in this course outline is correct at the time of publication, but may be subject to change as part of the Department’s policy of continuous improvement and development. Every effort will be made to notify you of any such changes.