CS1801 - Object Oriented Programming

Co-ordinator: Prof Adrian Johnstone / Dr Pierre-Malo Deniélou

Course value: 1.0

COURSE SUMMARY

To teach fundamental programming skills.

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

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

TEACHING AND 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.

KEY BIBLIOGRAPHY

- Herbert Schildt: Java: a Beginner's Guide, 4th Edition, McGraw-Hill, 2007, ISBN 0072263849
- Cary S. Horstmann: Big Java, 4th Edition, Wiley, 2010, ISBN 987-0-470-55309-1

FORMATIVE ASSESSMENT AND 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

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

CS1820 - Computing laboratory (robotics)

Co-ordinator: Prof. Dave Cohen

Course value: 0.5

COURSE SUMMARY

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

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

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

TEACHING AND LEARNING METHODS

Teacher led laboratory classes and self-driven, scripted group project work.Up to 4 hours of lectures and laboratory classes per week.

KEY BIBLIOGRAPHY

- J. Kelly: LEGO Mindstorms NXT-G Programming Guide, Technology in Action Press, 2007, ISBN 10: 1590598717
- Brian Bagnell: Maximum Lego NXT: Building Robots with Java Brains, Variant Press 2007, ISBN 10: 0973864915

FORMATIVE ASSESSMENT AND FEEDBACK

Five formative worksheetsVerbal feedback on progress will be given during the laboratory sessions.

SUMMATIVE ASSESSMENT

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

CS1830 - Computing laboratory (games)

Co-ordinator: Dr. Gregory Chockler / Dr. Chris Watkins

Course value: 0.5

COURSE SUMMARY

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

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

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

TEACHING AND LEARNING METHODS

Teacher led laboratory classes and self-driven, scripted project work.Up to 4 hours of lectures and laboratory classes per week.

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 AND FEEDBACK

Practical skills assessment – in Lab assessment. Verbal feedback on progress will be given during the laboratory sessions.

SUMMATIVE ASSESSMENT

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

CS1840 - Internet Services

Co-ordinator: Dr Carlos Matos

Course value: 0.5

COURSE SUMMARY

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

Course Content:

- Internet Basics: client-server model, protocols
- Internet Protocol Stack: example protocols such as HTTP, DNS, SMTP, TCP and IP
- Web Technologies: HTTP, XML
- Scripting languages: JavaScript (client-side) and PHP (server-side)
- Web Services: SOAP message exchange, WSDL service description, UDDI service discovery, introduction to REST,ATOM publishing Protocol

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

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by laboratory classes and tutorials.Up to 4 hours of lectures and laboratory classes per week for 10 weeks.

KEY BIBLIOGRAPHY

- C.D. Knuckles: Introduction To Interactive Programming On The Internet Using HTML and Javascript, Wiley, 2000, ISBN 9780471383666
- J.F. Kurose and K.W. Ross: Computer Networking: A Top-Down Approach Using The Internet, Addison-Wesley, 5 edition (27 Mar 2009), ISBN 9780131365483
- P.K Yuen and V. Lau: Practical Web Technologies, Addison Wesley, 2003, ISBN 0201750767

FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback on progress will be given during the weekly laboratory sessions. A zero-weighted 45 minutes mid-term test will be marked and returned with written feedback.

SUMMATIVE ASSESSMENT

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

CS1850 - Databases

Co-ordinator: Dr Carlos Matos

Course value: 0.5

COURSE SUMMARY

To provide the basic concepts of database technology.
To describe the need for database integrity and robustness.

To demonstrate the use of a modern database system in a web-based environment.

Course Content:

- Data modelling: views, subschema, data dictionary, data independence, entity relationship model
- The relational model: relations, attributes, domains, relational algebra
- Database design: normalisation, normal forms, entities and attributes
- SQL: basic 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

LEARNING OUTCOMES

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

- explain the 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 programs be familiar with the steps for the design and implementation of a database, from the user specifications to the final SQL program

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by laboratory classes and tutorials.
Normally 3 hours of lectures and laboratory classes per week.

KEY BIBLIOGRAPHY

- Database Systems Concepts (6th edition) A. Silbertschatz, H.F. Korth, S. Sudarshan McGraw Hill, 2011
- C.J. Date: An Introduction To Database Systems, 8th edition, Addison Wesley, 2003,

ISBN 0321197844

- P. Rob and C. Coronel: Database Systems: Design, Implementation, And Management, Course Technology Inc, 2002, ISBN 061906269X

- R. Elmasri and S. Navathe: Fundamentals Of Database Systems, 3rd edition, Addison Wesley, 2000, ISBN 0805317554

- D.M. Kroenke: Database Processing, Prentice Hall, 1997, ISBN 0137378424

FORMATIVE ASSESSMENT AND FEEDBACK

Immediate verbal feedback will be given on class exercises carried out in the laboratory sessions

SUMMATIVE ASSESSMENT

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

CS1860 - Mathematical structures

Co-ordinator: Prof. José Fiadeiro

Course value: 0.5

COURSE SUMMARY

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.

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

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

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by small group tutorials.Normally 3 hours of lectures per week.

KEY BIBLIOGRAPHY

- Rosen: Discrete Mathematics And Its Applications, McGraw Hill, 2006, ISBN 0071244743

- Ross and Wright: Discrete Mathematics, Prentice Hall, 2003, ISBN 0130652474

FORMATIVE ASSESSMENT AND FEEDBACK

Individual discussion of non-assessed tutorial exercises and discussion of assessed worksheets in small group tutorials.

SUMMATIVE ASSESSMENT

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

CS1870 - Machine fundamentals

Co-ordinator: Prof. Elizabeth Scott

Course value: 0.5

COURSE SUMMARY

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.

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.

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

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by laboratory classes and group tutorials.Normally 3 hours of lectures and laboratory classes per week.

KEY BIBLIOGRAPHY

- Rosen: Discrete Mathematics And Its Applications, McGraw Hill, 2006, ISBN 0071244743

- Brookshear: Computer Science An Overview, Addison-Wesley, 2007, ISBN 0321524039

FORMATIVE ASSESSMENT AND FEEDBACK

Individual discussion of non-assessed tutorial exercises and discussion of assessed worksheets in small group tutorials.

SUMMATIVE ASSESSMENT

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

IY2760 - Introduction to Information Security

Co-ordinator: Dr Kostas Markantonakis

Course value: 0.5

COURSE SUMMARY

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.

Course Content:

- Introduction: What is security (covering notions of Confidentiality, Integrity, and Availability)? Security threats and risks. Security management (ISO/IEC 17799). Data Protection legislation.
- Elements of cryptography: Ciphers (DES/AES). Message Authentication codes (MACs). Public key ciphers and digital signatures (RSA).
- Access control: Access Control Lists, capabilities, security labels (MAC and DAC), and role-based access control.
- Database security: The access path, views for security, integrity controls.
- Personal computer security: Viruses, spyware, restricting access.
- Identity verification: Use and storage of conventional passwords. Dynamic password schemes. Biometric techniques. Use of tokens (dumb and intelligent), including the use of smart cards.
- CASE STUDY I: Security of a modern Operating System (e.g. Windows or Linux).
- Network security concepts: The concepts of security services and security mechanisms (as in ISO 7498-2). Firewalls.
- 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).
- Security standards bodies: Introduction to roles of ISO (notably ISO/IEC SC27), ITU and IETF. A brief introduction to (security related) standards developed and under development.
- CASE STUDY II: Security for a networked application e.g. SSL/TLS.

LEARNING OUTCOMES

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

- explain the issues involved in the successful and secure management of information resources;
- apply the background knowledge gained to further study of the specialised topics within the subject area.

TEACHING AND LEARNING METHODS

33 hours of lectures and 117 hours non-assessed coursework and private study. The coursework is designed to allow students to apply their knowledge to applications with which they are familiar.

KEY BIBLIOGRAPHY

- The main recommended text for this course is:
- D. Gollmann, Computer Security, John Wiley & Sons, 2005 (2nd edition).
- Useful background:
- C.P. Pfleeger and S. L. Pfleeger, Security in Computing, Prentice Hall, 2006 (4th edition).
- D. Saloman: Elements of Computer Security, Springer 2010
- K. Martin: Everyday Crytography, Oxford University Press 2012

FORMATIVE ASSESSMENT AND FEEDBACK

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

SUMMATIVE ASSESSMENT

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

CS2800 - Software engineering

Co-ordinator: Prof. Dave Cohen / Dr Carlos Matos

Course value: 0.5

COURSE SUMMARY

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.

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.

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.

TEACHING AND LEARNING METHODS

Lecture based delivery with class exercises and worksheets. Use of structured practice sessions.Up to 4 hours of lectures per week.

KEY BIBLIOGRAPHY

- Cary S. Horstmann: Object Oriented Design And Patterns, 2nd Edition, Wiley, 2005, ISBN 978 0471744870

- E. Freeman, E. Freeman: B. Bates and K. Sierra, Head First Design Patterns, O'Reilly, 2004, ISBN 978 0596007126

FORMATIVE ASSESSMENT AND FEEDBACK

Feedback will be given based on questions on the Moodle forum and any common mistakes made in coursework.

SUMMATIVE ASSESSMENT

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

CS2810 - Team Project

Co-ordinator: Dr Johannes Kinder / Dr Carlos Matos

Course value: 0.5

COURSE SUMMARY

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

Course Content:

- The software lifecycle: Models of software development, planning, documentation, costing, quality assurance, reliability modelling.
- Team structures: large teams, ownership, change control.
- Practical experience of standard industrial software engineering systems: Scrum project management, Version Control, IDE, Fault tracking, Team IDE etc.
- A medium scale eXtreme software development project, conducted by teams of between four and six students.

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.
- understand the use of parameterised reliability models to estimate software quality.

TEACHING AND 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.

KEY BIBLIOGRAPHY

- Richard Whitehead: Leading a Software Development Team, Addison Wesley, 2001 ISBN 10: 0201675269

- Cary S. Horstmann: Big Java, 3rd Edition, Wiley, 2007, ISBN 978 0470105542
- Ken Schwaber: Agile Software Development with Scrum. Pearson Prentice Hall 2002. ISBN 978 0132074896

FORMATIVE ASSESSMENT AND 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

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

CS2812 - Team Project in Artificial Intelligence

Co-ordinator: Dr Johannes Kinder / Dr Carlos Matos

Course value: 0.5

COURSE SUMMARY

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

Course Content:

- The software lifecycle: Models of software development, planning, documentation, costing, quality assurance, reliability modelling.
- Team structures: large teams, ownership, change control.
- Practical experience of standard industrial software engineering systems: Scrum project management, Version Control, IDE, Fault tracking, Team IDE etc.
- A medium scale eXtreme software development project, conducted by teams of between four and six students.

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.
- understand the use of parameterised reliability models to estimate software quality.

TEACHING AND 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.

KEY BIBLIOGRAPHY

- Richard Whitehead: Leading a Software Development Team, Addison Wesley, 2001 ISBN 10: 0201675269

- Cary S. Horstmann: Big Java, 3rd Edition, Wiley, 2007, ISBN 978 0470105542
- Ken Schwaber: Agile Software Development with Scrum. Pearson Prentice Hall 2002. ISBN 978 0132074896
- Nils Nilsson: Artificial Intelligence, A new Synthesis. Morgan Kaufman 1998. ISBN 1558604677

FORMATIVE ASSESSMENT AND 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

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

CS2821 - Systems Programming

Co-ordinator: Dr Johannes Kinder / Dr Matthew Hague

Course value: 0.5

COURSE SUMMARY

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

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.

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.

TEACHING AND 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.

KEY BIBLIOGRAPHY

- Frantisek Franik: Memory as a Programming Concept in C and C++, Cambridge University Press, 2004 ISBN 10:0521520436

- Herbert Schildt: C++ A Beginners Guide, McGraw Hill 2003 (2nd Edition)
- W.R. Stevens, S.A.Rago: Advanced Programming in the UNIX Environment, Addison-Wesley Professional 2005 (2nd Edition)
- C.Abling, JP Vossen, C. Newham: Bash Cookbook, O'Reilly Media 2007

FORMATIVE ASSESSMENT AND 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

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

CS2830 - Robotics

Co-ordinator: Dr Hugh Shanahan

Course value: 0.5

COURSE SUMMARY

To introduce practical robotics

To introduce simple dynamic control 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: Kalman and Particle Filters
- A project to implement a robot that can construct a world map

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
- program significant robotic applications in Java on Lego NXT systems

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by laboratory classes. Self guided project work.Up to 3 hours of lectures and laboratory classes per week.

KEY BIBLIOGRAPHY

- John L. Craig: Introduction to Robotics: Mechanics and Control, Pearson Education, 2003, ISBN 10: 0131236296
- J. Kelly: LEGO Mindstorms NXT-G Programming Guide, Technology in Action Press, 2007, ISBN 10: 1590598717

- Brian Bagnell: Maximum Lego NXT: Building Robots with Java Brains, Variant Press 2007, ISBN 10: 0973864915

FORMATIVE ASSESSMENT AND FEEDBACK

Verbal feedback on progress will be given during the laboratory sessions.

SUMMATIVE ASSESSMENT

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

CS2844 - Computer Graphics

Co-ordinator: Dr Jackie Daykin

Course value: 0.5

COURSE SUMMARY

To introduce students to the concepts, algorithms and basic geometry needed for producing computer graphics

Course Content:

- Geometry: vectors, matrices, dot and cross product, transformations, finding intersections of simple geometrical objects, homogeneous representations of transformations, rendering pipeline
- Perspective: projection, viewpoint, computation of viewing transformation from camera position
- Appearance rendering: construction of spline curves and surfaces, lighting, reflection, texture, Gouraud and Phong shading, principles of ray tracing and photorealistic rendering, representation and reproduction of colour
- Modelling: Hierarchical modelling of scenes, and of complex objects with moving parts
- Implementation: practical implementation of a 3D animation

LEARNING OUTCOMES

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

- demonstrate a knowledge of basic computer graphics
- use mathematical techniques including co-ordinate geometry, homogenous vectors and projective environments in graphics applications
- design and implement a simple 3D animated scene

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by practical sessions.Up to 4 hours of lectures and laboratory classes per week.

KEY BIBLIOGRAPHY

- P.A. Egerton: Computer Graphics: Mathematical First Steps, Prentice Hall.
- James D. Foley: Introduction to Computer Graphics, Addison Wesley

- Hearn & Baker: Computer Graphics with OpenGL. Pearson Prentice Hall (2004)
- Chad Carter: Microsoft XNA Unleashed, Graphics and Game Programming for Xbox 360 and Windows. SAMS (2008)

FORMATIVE ASSESSMENT AND FEEDBACK

Non-assessed programming exercises and an assessed individual project. The non-assessed exercises will be discussed in class.

SUMMATIVE ASSESSMENT

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

CS2850 - Network Operating Systems

Co-ordinator: Prof Zhaohui Luo

Course value: 0.5

COURSE SUMMARY

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

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

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

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by practical sessions.Normally 3 hours of lectures and laboratory classes per week.

KEY BIBLIOGRAPHY

- A.S. Tanenbaum: Modern Operating Systems , 2nd Edition, Prentice Hall, 2001 , ISBN 01336006639
- W. Stallings: Operating Systems: Internals And Design Principles, 4th Edition, Prentice

Hall, 2001, ISBN 0136006329

- G. Nutt, Operating Systems: A Modern Perspective, 2nd Edition, Addison Wesley, 2002, ISBN 0201773449

FORMATIVE ASSESSMENT AND FEEDBACK

Written grades will be given for the assessed assignments.

SUMMATIVE ASSESSMENT

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

CS2860 - Algorithms and Complexity I

Co-ordinator: Dr Magnus Wahlstrom

Course value: 0.5

COURSE SUMMARY

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

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:
- Combinational optimisation: exhaustive search, heuristics, simulated annealing
- Game Theory: Min-max trees, alpha-beta trees
- String Matching: naïve and Rabin-Karp string matching algorithm

LEARNING OUTCOMES

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

- Understand and reason about alternative data structure
- Implement and reason about alternative implementations for basic algorithms
- Calculate the complexity of basic algorithms

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by laboratory classes and problems sessions. Normally 3 hours of lectures and labs per week.

KEY BIBLIOGRAPHY

- T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein: Introduction to Algorithms, 2nd edition, MIT Press, Cambridge MA, 2001. ISBN 0262531968
- Algorithm Design: foundations, analysis and Internet examples, Goodrich and

Tammasia, New York, Wiley, 2002. ISBN 0471383651

- Algorithms: S.Desgupta, C.H. Papadimitriou and U.V. Vazirani. http://www.cs.berkeley.edu/~vazirani/algorithms.html

- Data Structures and Algorithms: K.Mehlhorn and P. Sanders. http://www.mpi-inf.mpg.de/~mehlhorn/Toolbox.html

FORMATIVE ASSESSMENT AND FEEDBACK

Lab Based Assignments and In-Lecture assignments.Verbal feedback will be given during the laboratory sessions and feedback provided during lectures.

SUMMATIVE ASSESSMENT

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

CS2870 - Algorithms and Complexity 2

Co-ordinator: Prof Gregory Gutin

Course value: 0.5

COURSE SUMMARY

To teach basic graph theory, the design and analysis of fundamental graph theoretic algorithms, and basic of complexity theory.

Course Content:

- Basic Graph Properties, Algorithms and Applications: Graph models, basic notions in graph theory, graphic data structures, paths and walks in directed and undirected graphs, the shortest path problem, special classes of graphs, trees and cycles, minimum spanning trees algorithms, connectivity.

- Further Graph Properties, Algorithms and Applications: Matchings in graphs, Hamilton cycles and TSP. Acyclic digraph algorithm, strong connectivity algorithm and their applications. Graph colourings, greedy algorithm for interval graphs and its applications, edge colouring.

- NP-hardness: General notions, examples, simple transformations, introduction to exact algorithms, approximation algorithms and heuristics.

LEARNING OUTCOMES

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

- . implement and reason about fundamental graph-theoretic algorithms

- describe several applications of graph-theoretic algorithms

- understand the significance of NP-hard problems and use heuristic approaches

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by laboratory classes and problems sessions. Normally 3 hours of lectures per week for 10 weeks.

KEY BIBLIOGRAPHY

- T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, Introduction to Algorithms, 2nd Ed., MIT Press, Cambridge MA, 2001.

- G. Gutin, Algorithms and Complexity II, Lecture Notes.

FORMATIVE ASSESSMENT AND FEEDBACK

Two mandatory zero-weighted formative assignments (coursework).

SUMMATIVE ASSESSMENT

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

CS3001 - Extramural Year

Co-ordinator: Dr Alberto Paccanaro

Course value: 1.0

COURSE SUMMARY

To give knowledge of the software tools important for industry and the day-to-day running of a company. To give insight into the managerial skills needed in industry. This is an optional 'year out' between the second and third year, spent at an organisation or institution approved by the College as having a function relevant and suitable to Computer Science. A report will be submitted and assigned the value of 1 course unit.

In order to obtain College and Departmental approval for an extramural year proposal, a student must submit to the Computer Science Extramural Year Coordinator the following details:

- Name of student
- Name and address of firm or other institution where the extramural year be spent.
- Name and telephone number of a member of staff of this firm or institution who will be responsible for the student during the proposed extramural year.
- The name of a member of the staff of the RHUL Computer Science Department who has agreed to serve as academic supervisor for the extramural year.
- A statement of the proposed plan of work. This should be not more than about one side and not less than about half a side of single-spaced typed A4.
- Whether the placement will be salaried or not.

If approval for an extramural year is granted, the College Registry, rather than the individual student, will make arrangements with the local education authority, in cases where this is necessary.
Extramural year students are required to set up email forwarding, before they leave campus, so that they can be contacted with departmental information should the need arise (re projects, preliminary registration, accommodation, for example). As all such necessary information will be communicated by email, it is essential that students with no access to electronic mail should inform the office, before they leave campus, so that alternative arrangements can be made.

LEARNING OUTCOMES

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

- appreciate the importance of the impact of IT on commerce and industry focused on improving the efficiency and quality of information on the workforce

- understand imminent industry changes and what effect they will have
- apply their knowledge and understanding to practical situations

TEACHING AND LEARNING METHODS

N/A

KEY BIBLIOGRAPHY

- N/A

FORMATIVE ASSESSMENT AND FEEDBACK

The student is visited twice, during the secondment, at the Company.

SUMMATIVE ASSESSMENT

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

CS3010 - Advanced Topics Seminar

Co-ordinator: Dr Johannes Kinder

Course value: 0.5

COURSE SUMMARY

This course will give students the chance to see Computer Science as an area of research that is valid in and of itself and that research topics in CS can have surprisingly quick application.

Course Content:

- A series of two hour seminars where any topic in Computer Science is discussed. Seminar speakers from a variety of areas will be picked including academic staff, graduate students and invited speakers.

LEARNING OUTCOMES

By the end of the course students should be able to:

- Appreciate that Computer Science is an ongoing area of research.
- Grasp that ideas themselves are worth understanding.
- Understand that ideas in Computer Science are continually finding application.
- Critically appraise and analyse concepts that are not yet part of mainstream academic Computer Science.
- Engage in and contribute to discussions that occur during Scientific and technical talks.

TEACHING AND LEARNING METHODS

Seminars

KEY BIBLIOGRAPHY

- Pre-seminar reading lists available electronically may be provided.

FORMATIVE ASSESSMENT AND FEEDBACK

N/A

SUMMATIVE ASSESSMENT

Oral assessment based on involvement in seminar either via questions or discussions in seminar itself or via social networks.

CS3110 - Bioinformatics

Co-ordinator: Dr Alberto Paccanaro

Course value: 0.5

COURSE SUMMARY

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

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.
- Dynamic Programming, sequence alignments, substitution matrices.
- Phylogenetic trees.
- Systems biology: gene expression analysis, Protein-Protein Interaction analysis, biological networks, clustering

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.

TEACHING AND LEARNING METHODS

Lecture based delivery. Normally 3 hours of lectures per week.

KEY BIBLIOGRAPHY

- Understanding Bioinformatics Marketa Zvalebil, Jeremy O. Baum Garland Science, 2007
- R.Durbin, S.Eddy, A.Krogh, and G.Mitchinson: Biological sequence analysis, Cambridge University Press, 1998 : ISBN-10: 0521629713
- R.Duda,P.Hart,D.Stork: Pattern classification, John Wiley & Sons, 2001:ISBN-10: 0471056693
- B.Levin: Genes VIII, Prentice Hall, 2004: ISBN-10: 0131239244

- A.M.Lesk: Introduction to Bioinformatics, Oxford University Press, 2005: ISBN-10: 0199277877

FORMATIVE ASSESSMENT AND FEEDBACK

Lectures in which students work through exercises getting verbal feedback.

SUMMATIVE ASSESSMENT

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

CS3220 - Fundamentals of Digital Sound and Music

Co-ordinator: Dr Chris Watkins

Course value: 0.5

COURSE SUMMARY

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.

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

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

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by laboratory classes.Up to 4 hours of lectures and laboratory classes per week.

KEY BIBLIOGRAPHY

- Gareth Loy: Musimatics, the Mathematical Foundations of Music Volume 1. MIT Press 2010. ISBN 13:978-0262516556

- Gareth Loy: Musimatics, the Mathematical Foundations of Music Volume 2. MIT Press 2010. ISBN 13: 978-0262516563

- William Sethares: Tuning, Timbre, Spectrum and Scale. Springer 2nd edition 2005. ISBN 13: 978-1849969222

FORMATIVE ASSESSMENT AND FEEDBACK

Return of marked zero-weighted assignments, verbal feedback on project work during the laboratory sessions.

SUMMATIVE ASSESSMENT

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

CS3450 - Software Verification

Co-ordinator: Dr Axel Legay / Prof. José Fiadeiro

Course value: 0.5 TU

COURSE SUMMARY

The aim of this course is to introduce students to system verification using model-checking techniques that are now adopted in industry. Model checking is a formal technique that, over the last two decades or so, has proved to be a very attractive approach towards verifying the correctness of computer-based control systems. It allows for desired behavioral properties of a given system to be verified on the basis of a suitable model of the system through systematic inspection of all states of the model.

Course Content:

- Elements of language theory and automata
- Formal modelling of concurrent systems and logical requirement specifications
- Temporal logic (linear and branching)
- (Symbolic) model checking
- Introduction to the SPIN and SMV model checkers

LEARNING OUTCOMES

On completion of the course, students should be able to:

- Appreciate the need for verifying the correctness of systems against formal specification of requirements.
- Express requirements on concurrent systems using temporal logic (linear and branching).
- Model concurrent systems using automata-based design languages.
- Master the principles of model checking.
- Use model-checkers such as SPIN and SMV to verify specifications against models of concurrent systems.

TEACHING AND LEARNING METHODS

Lecture-based delivery, supported by practical sessions.Block-mode teaching – up to 9 hours per week of labs and lectures.

KEY BIBLIOGRAPHY

- "Principles of Model-Checking", Christel Baier and Joost-Pieter Katoen, MIT Press 2008, ISBN-13: 978-0262026499

FORMATIVE ASSESSMENT AND FEEDBACK

Practical skills assessment carried out within laboratory sessions. Verbal feedback provided during sessions.

SUMMATIVE ASSESSMENT

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

CS3470 - Compilers and code generation

Co-ordinator: Prof Elizabeth Scott

Course value: 0.5

COURSE SUMMARY

To describe how to construct and implement interpreters and compilers for modern processors

Course Content:

- Lexical analysis: input buffering, regular expressions, finite state automata, hash coded symbol tables, LEX
- Syntax analysis: context free grammars, derivations, recursive descent and table-based LR parsing, YACC Semantics: abstract parse trees, syntax directed translation, intermediate forms and three address code, attributes, annotated parse trees, semantic rules and attribute grammars, translation schemes and L-attribute grammars
- Error detection and recovery: classes of error and error recovery in top down parsers
- Intermediate code generation: using attribute grammars, three address code
- Code improvement: basic blocks and code improvement techniques, flow graphs, loop improvement and fusion, directed a cyclic graphs for identifying code improvements
- Complier generation with tool support

LEARNING OUTCOMES

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

- explain the role and structure of a compiler and the standard stages of compilation
- to build a DFA based lexical analyser for a set of tokens specified using regular expressions
- Advanced students should be able to construct regular expressions, which define specified set of strings
- use grammars to define context free languages and to build parsers for them
- describe syntax directed translators and use them to construct intermediate code
- describe various types of error detection and recovery

- generate three address code from source code.

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by small group tutorials.Normally 3 hours of lectures per week

KEY BIBLIOGRAPHY

- Aho, Lam, Sethi, Ullman: Compilers: Principles, Techniques, and Tools, Addison Wesley, 2006 ISBN: 10: 0321486811

- J. Tremblay and P.G. Sorenson: The theory and practice of compiler writing, McGraw Hill, 1985, ISBN 0-07-065161-2

- N. Wirth: Algorithms + Data Structures = Programs, Prentice Hall, 1976.

FORMATIVE ASSESSMENT AND FEEDBACK

Return of coursework grades and comments, in-class discussion of coursework solutions

SUMMATIVE ASSESSMENT

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

CS3490 - Computational Optimisation

Co-ordinator: Prof Gregory Gutin

Course value: 0.5

COURSE SUMMARY

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.

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

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.

TEACHING AND LEARNING METHODS

Lecture based delivery.Normally 3 hours of lectures per week plus independent guided study

KEY BIBLIOGRAPHY

- M.W. Carter and C.C. Price: Operations Research: A practical Introduction, CRC,

2001 (001.424 CAR): ISBN-10: 0849322561

- Z. Michalewicz and D.B. Fogel: How to Solve It: Modern Heuristics, Springer, 2000: ISBN-10: 3540224947
- F. Glover and M. Laguna: Tabu Search, Kluwer, 1997: ISBN-10: 079239965X
- J. Bang-Jensen and G. Gutin: Digraphs: Theory, Algorithms and Applications, Springer, 2000: ISBN-10: 1852336110

FORMATIVE ASSESSMENT AND FEEDBACK

Return of marked zero-weighted assignments. Some solutions discussed in class.

SUMMATIVE ASSESSMENT

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

CS3510 - Functional Programming and Applications

Co-ordinator: Prof Zhaohui Luo

Course value: 0.5

COURSE SUMMARY

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.

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

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

TEACHING AND LEARNING METHODS

Normally 3 hrs per week of lectures and practicals.

KEY BIBLIOGRAPHY

- M. Odersky, L. Spoon and B. Venners. Programming in Scala. 2nd ed. Dec 2010.
- S. Thompson. Haskell: The Craft of Functional Programming. 3rd ed. 2012.
- C.S. Horstmann. Scala for the Impatient. Addison-Wesley 2012

FORMATIVE ASSESSMENT AND FEEDBACK

Written assignment with written feedback

SUMMATIVE ASSESSMENT

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

CS3580 - Advanced Data Communications

Co-ordinator: Dr Zhiyuan Luo

Course value: 0.5

COURSE SUMMARY

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

Course Content:

- Introduction: layering, abstraction, and standards. The ISO/OSI reference model. The Internet reference model.
- 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.

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

TEACHING AND LEARNING METHODS

Lecture based delivery.Normally 3 hours of lectures per week.

KEY BIBLIOGRAPHY

- James F. Kurose and Keith W. Ross: Computer Networking: a top-down approach featuring the Internet, Addison-Wesley: ISBN-10: 0321227352

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

- William Stallings: Cryptography and Network Security: principles and practice, Prentice Hall: ISBN-10: 0131873164

FORMATIVE ASSESSMENT AND FEEDBACK

Return of marked zero-weighted assignments.

SUMMATIVE ASSESSMENT

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

CS3750 - Concurrent and Parallel Programming

Co-ordinator: Dr Hugh Shanahan

Course value: 0.5

COURSE SUMMARY

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.

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.
- Threading paradigm: implementation in Java, launching threads, shared states and race conditions, locking, conditional locking and deadlock.
- Message Passing paradigm: 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. Multi-core processors and GPU's. Distributed computing. Grid and Cloud computing.

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

TEACHING AND LEARNING METHODS

This course will be taught in block mode, normally over a period of 4 weeks.Normally 10 hours of lectures, tutorials and laboratory sessions per week.

KEY BIBLIOGRAPHY

- Cary S. Horstmann: Big Java, 4th Edition, Wiley, 2007, ISBN 987-0-470-55309-1
- A.Grama,A.Gupta,G.Karypis,V.Kumar: Introduction to Parallel Computing:ISBN-10: 0201648652

FORMATIVE ASSESSMENT AND FEEDBACK

Written assignment with peer feedback during tutorials.

SUMMATIVE ASSESSMENT

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

CS3760 - Information Security

Co-ordinator: Dr Kostas Markantonakis

Course value: 0.5

COURSE SUMMARY

To providing a general introduction to Information Security. To provide the background knowledge needed to study the subject area at an advanced level.

Course Content:

- Introduction: What is security (covering notions of Confidentiality, Integrity, and Availability)? Security threats and risks. Security management (the DTI code of practice). The Data Protection Act.
- Elements of cryptography: Ciphers (DES). Authentication codes (MACs). Public key ciphers and digital signatures (RSA).
- Access control: Access Control Lists, capabilities, security label, role-based access control.
- Database security: The access path, views for security, integrity controls.
- Personal computer security: Viruses, restricting access.
- Identity verification: Use and storage of conventional passwords. Dynamic password schemes. Biometric techniques. Use of tokens (dumb and intelligent), smart cards.
- CASE STUDY I: Unix security.
- Network security concepts: Security services and security mechanisms (as in ISO 7498--2).
- Authentication and key distribution: 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). Firewalls.
- Security standards bodies: Introduction to roles of ISO, ITU, CEN, ETSI and BSI. The main roles of ISO/IEC SC21 and SC27 and a brief introduction to (security related) standards.
- CASE STUDY II: Security for a LAN. Internet Privacy Enhanced Mail (PEM).

LEARNING OUTCOMES

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

- demonstrate a thorough understanding of the successful and secure management of information resources
- apply the background knowledge gained to more advanced study of the specialised topics within the subject area

TEACHING AND LEARNING METHODS

Lecture based delivery.Normally 3 hours of lectures per week.

KEY BIBLIOGRAPHY

- The main recommended text for this course is:
- D. Gollmann, Computer Security, John Wiley & Sons, 2005 (2nd edition).
- Useful background:
- C.P. Pfleeger and S. L. Pfleeger, Security in Computing, Prentice Hall, 2006 (4th edition).
- D. Saloman: Elements of Computer Security, Springer 2010
- K. Martin: Everyday Crytography, Oxford University Press 2012

FORMATIVE ASSESSMENT AND FEEDBACK

Return of marked zero-weighted exercises. Some solutions will be discussed in class

SUMMATIVE ASSESSMENT

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

CS3810 - Individual project – Half Unit

Co-ordinator:

Course value: 0.5

COURSE SUMMARY

To provide the opportunity to demonstrate independence and originality, to plan and organise a large project over a long period, and to put into practice the techniques taught throughout their degree course.

Course Content:

- Related to Individual Project

LEARNING OUTCOMES

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

- understand complex ideas and identify solutions to a problem,
- apply scientific and software/hardware techniques to solving a problem,
- analyse the solution to a problem and evaluate the associated results,
- demonstrate the ability to plan, organise and present the project work both in written form and orally.

TEACHING AND LEARNING METHODS

1 hour supervision per week. In addition to this there will be an additional 1 hour for an end of term presentation, 1 hour for a demonstration session and 4 hours of professional lectures.

KEY BIBLIOGRAPHY

- C. W. Dawson: The Essence of Computing Projects, A Student's Guide, Prentice Hall, 2000: ISBN-10: 013021972X
- M. Berndtsson, J. Hansson, B. Olsson, B. Lundell: Planning and Implementing your Final Year Project-with Success: A Guide for Students in Computer Science and Information Systems, Springer-Verlag, 2006: ISBN-10: 1852333324

FORMATIVE ASSESSMENT AND FEEDBACK

Weekly meetings with supervisor to provide verbal feedback on progress and guidance on future work.

SUMMATIVE ASSESSMENT

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

CS3821 – Full Unit - Individual project

Co-ordinator:

Course value: 1.0

COURSE SUMMARY

To provide the opportunity to demonstrate independence and originality, to plan and organise a large project over a long period, and to put into practice the techniques taught throughout their degree course.

Course Content:

- Related to individual project
- Legal, social, ethical and professional issues

LEARNING OUTCOMES

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

- understand complex ideas and identify solutions to a problem,
- apply scientific and software/hardware techniques to solving a problem,
- analyse the solution to a problem and evaluate the associated results
- demonstrate the ability to plan, organise and present the project work both in written form and orally,
- understand legal, social, ethical and professional issues.

TEACHING AND LEARNING METHODS

1 hour supervision per week. In addition to this there will be an additional 1 hour for an end of term presentation, 1 hour for a demonstration session and 4 hours of professional lectures.

KEY BIBLIOGRAPHY

- C. W. Dawson: The Essence of Computing Projects, A Student's Guide, Prentice Hall, 2000: ISBN-10: 013021972X
- M. Berndtsson, J. Hansson, B. Olsson, B. Lundell: Planning and Implementing your Final Year Project - with Success!: A Guide for Students in Computer Science and Information Systems, Springer-Verlag, 2002: ISBN-10: 1852333324
- Richard Epstein: The case of the Killer Robot, John Wiley & Sons, 1997: ISBN 978-0471138235.

FORMATIVE ASSESSMENT AND FEEDBACK

Weekly meetings with supervisor to provide verbal feedback on progress and guidance on future work.One mandatory independent study report on a topic in legal, social, ethical and professional issues will be marked and returned.

SUMMATIVE ASSESSMENT

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

CS3822 - Individual Project in Artificial Intelligence

Co-ordinator:

Course value: 1.0

COURSE SUMMARY

To provide the opportunity to demonstrate independence and originality, to plan and organise a large project over a long period, and to put into practice the techniques taught throughout their degree course.
The topic of the project will be related to artificial intelligence.

Course Content:

- Related to individual project
- Legal, social, ethical and professional issues

LEARNING OUTCOMES

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

- understand complex ideas and identify solutions to a problem,
- apply scientific and software/hardware techniques to solving a problem
- analyse the solution to a problem and evaluate the associated results
- demonstrate the ability to plan, organise and present the project work both in written form and orally,
- understand legal, social, ethical and professional issues.

TEACHING AND LEARNING METHODS

1 hour supervision per week. In addition to this there will be an additional 1 hour for an end of term presentation, 1 hour for a demonstration session and 4 hours of professional lectures.

KEY BIBLIOGRAPHY

- C. W. Dawson: The Essence of Computing Projects, A Student's Guide, Prentice Hall, 2000: ISBN-10: 013021972X.
- M. Berndtsson, J. Hansson, B. Olsson, B. Lundell: Planning and Implementing your Final Year Project - with Success!: A Guide for Students in Computer Science and Information Systems, Springer-Verlag, 2002: ISBN-10: 1852333324.
- Richard Epstein: The case of the Killer Robot, John Wiley & Sons, 1997: ISBN 978-0471138235.

FORMATIVE ASSESSMENT AND FEEDBACK

Weekly meetings with supervisor to provide verbal feedback on progress and guidance on future work.One mandatory independent study report on a topic in legal, social, ethical and professional issues will be marked and returned.

SUMMATIVE ASSESSMENT

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

CS3920 - Computer Learning

Co-ordinator: Prof Volodya Vovk

Course value: 0.5

COURSE SUMMARY

To introduce modern machine learning techniques, and to develop the theoretical background in learning methods and to give practical experience in developing computer learning systems. To understand a variety of applications such as image recognition and face recognition.

Course Content:

- Introduction to machine learning: basic ideas of learning theory; notion of a learning machine; main applications
- Classical learning algorithms, such Neural Networks and Nearest Neighbours methods; their properties
- Support Vector Machines; using Support Vector Machines for classification; Support Vector regression
- Kernel Techniques:the dual form of Support Vector Machines; the kernel trick; kernelized Nearest Neighbours; Kernel Ridge Regression
- Model selection and evaluation

LEARNING OUTCOMES

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

- demonstrate knowledge of the theoretical background in machine learning methods
- demonstrate an understanding of how to solve machine learning problems involving massive numbers of parameters
- Describe ways to apply successfully machine learning techniques in pattern recognition, medicine, and other areas

TEACHING AND LEARNING METHODS

Lecture based delivery.Normally 3 hours of lectures and labs per week.

KEY BIBLIOGRAPHY

- Vladimir N. Vapnik: Statistical Learning Theory, Wiley, 1997: ISBN-10: 0471030031
- John Shawe-Taylor and Nello Cristianini: Kernel Methods for Pattern Analysis,

Cambridge University Press, 2004: ISBN-10: 0521813972

FORMATIVE ASSESSMENT AND FEEDBACK

In lecture coursework with feedback in lectures.

SUMMATIVE ASSESSMENT

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

CS3930 - Computational Finance

Co-ordinator: Dr Yuri Kalnishkan

Course value: 0.5

COURSE SUMMARY

To understand the key role played by the advent of derivatives, financial instruments which facilitate managing financial risks. To understand pricing derivatives (and associated strategies of dynamic hedging) using advanced computational models are required.

Course Content:

- Introduction: financial markets; the rules of the game.
- Taxonomy of securities: main kinds of derivative securities and underlying markets.
- Mathematical techniques: Wiener process; diffusion processes as mathematical models of price dynamics; stochastic differential equations; computer simulations.
- Pricing and hedging in the Black-Scholes world: risk-neutral valuation; the Black-Scholes equation and analytic formulae; the "Greeks" and their use.
- Beyond the Black-Scholes world: application issues; computational models; fractals and their use in finance.
- Efficient markets hypothesis: theory vs empirical evidence.
- Risk management: Value at Risk.
- Coursework Project: implementing valuation algorithms for different derivatives (e.g., in MatLab); solving numerical and theoretical problems.

LEARNING OUTCOMES

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

- demonstrate an understanding of mathematical and computational models of underlying and derivative securities;
- master techniques for pricing derivatives and for dynamic hedging;
- apply these models and techniques for creating computer programs.

TEACHING AND LEARNING METHODS

Lecture based delivery, supported by practical classes.Normally 3 hours of lectures and laboratory classes per week.

KEY BIBLIOGRAPHY

- John Hull, Options, Futures and Other Derivatives, 6th edition, Prentice-Hall, 2007.
- ISBN-10: 0131977059 (earlier editions starting from the 2nd are acceptable)
- Paul Wilmott Introduces Quantitative Finance, John Wiley, 2001. ISBN-10: 0471498629
- Paul Wilmott, Sam Howison, and Jeff Dewynne, The Mathematics of Financial Derivatives: A Student Introduction, Cambridge University Press, 1995. ISBN-10: 0521497892

FORMATIVE ASSESSMENT AND FEEDBACK

Return of grades and written comments on courseof grades and written comments on courswork projecte submission of courseworkwork project

SUMMATIVE ASSESSMENT

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