Course Outlines

Year 1

PRINCIPLES OF PROGRAMMING

Description

Aims:

This module provides an introduction to computer programming using two different programming styles, imperative and functional programming. Its primary aim is to develop core design, programming, and problem-solving skills, with a secondary aim of building confidence in the ability to take on and learn new programming languages within a short space of time. In order to achieve these aims the module has a substantial practical element in the form of compulsory lab classes where students work through sets of programming exercises to apply the programming concepts introduced during the module lectures.

Learning outcomes:

On successful completion of the module, a student will be able to:

Design, implement, and debug small programs using two different programming paradigms, given straightforward specifications.
Develop straightforward algorithms to solve a range of common programming problems.
Compare and contrast the different paradigms, understanding the relative advantages and disadvantages of each.
Use common programming tools such as compilers, editors and debuggers.
Gain a basic understanding of how programs use computer memory, memory management, and the binary representation of data values.
Indicative content:

The following are indicative of the topics the module will typically cover:

Core Programming Concepts:

What is programming?
Programming languages.
Programs and algorithms.
Compilers and tools.
Running and debugging programs.
Introduction to Imperative Programming:

Core imperative programming ideas: sequence, selection, iteration, recursion, assignment and variables.
Types and type checking.
Functions, parameters, scope and lifetime.
Arrays and files.
Pointers, memory allocation, and memory management.
Introduction to Functional Programming:

Core functional programming ideas.
Recursion and recursive data structures such as lists.
Lazy evaluation.
Role of types.
Higher order functions.
List comprehension.
Recursive functions.
Program design in the small:

Designing and implementing small programs.
Implementing and using basic algorithms and data structures.
Good programming and design practice.
Requisites:

To be eligible to select this module as optional or elective, a student must be registered on a programme and year of study for which it is formally available.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 1   Undergraduate (FHEQ Level 4)
Teaching and assessment
Mode of study
In person
Methods of assessment
90% Exam
10% Coursework
Mark scheme
Numeric Marks

THEORY OF COMPUTATION
Description
Aims:

The module aims to introduce students to formal logical reasoning and to fundamental concepts in the theory of computation and formal languages.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Identify and reason with the logical content of arguments.
Recognise, write down and reason about automata and language grammars.
Carry out standard proofs and refutations involving logic and computational models.
Indicative content:

This module introduces students to the theory of computation. The first half of the module will focus on mathematical logic, including: propositional logic, first-order logic, proof by induction and modal logic. The second half will focus on fundamentals of computation, automata and language theory.

Requisites:

To be eligible to select this module as optional or elective, a student must: (1) be registered on a programme and year of study for which it is a formally available; and (2) have A-level Mathematics (or equivalent).

Module deliveries for 2024/25 academic year
Intended teaching term: Term 2 Undergraduate (FHEQ Level 4)
Teaching and assessment
Mode of study
In Person
Methods of assessment
100% Exam
Mark scheme
Numeric Marks

OBJECT-ORIENTED PROGRAMMING
Description
Aims:

This module provides a thorough introduction to object-oriented (OO) programming. The key object-oriented concepts such as class design, inheritance, dynamic binding and interfaces, are introduced along with design strategies for smaller sized object-oriented programs. Material is also included on data representation, the implementation of common data structures such as lists, trees and hash tables, basic cloud-based applications, and on development tools and practices. To learn how to apply object-oriented design concepts, and practice programming skills, the module has a substantial practical element, with weekly lab classes, sets of exercise questions, and an individual programming coursework.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Design and implement smaller-scale object-oriented programs.
Have a good knowledge of key object-oriented principles and design strategies.
Read, write, and process data sets using object-oriented techniques.
Understand how core data structures are implemented using classes.
Develop basic object-oriented applications that run in a cloud environment.
Use programming tools such as an integrated development environment (IDE), debugger, and version control.
Indicative content:

The following are indicative of the topics the module will typically cover:

Object-Oriented Principles:

Taking an object-oriented approach to software development.
Classes and the use of abstractions.
Key design principles and development practices.
Object-Oriented Programming:

Designing and implementing classes.
Identifying and implementing class relationships.
Using abstract classes, interfaces and inheritance.
Developing smaller-scale programs taking an object-oriented approach.
Effective object-oriented design and programming practices.
Documenting designs using class diagrams.
The design of basic cloud-based applications.
Data Representation:

Implementing common data structures.
Storing data in files.
Processing data to extract information.
Requisites:

To be eligible to select this module as optional or elective, a student must: (1) be registered on a programme and year of study for which it is a formally available; and (2) have taken Principles of Programming (COMP0002) in Term 1.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 2   Undergraduate (FHEQ Level 4)
Teaching and assessment
Mode of study
In person
Methods of assessment
90% Exam
10% Coursework
Mark scheme
Numeric Marks

ALGORITHMS
Description
Aims:

The aim of this module is to provide students with an introduction to data structures and algorithms that are fundamental to programming and to understand computation. We will apply these techniques to computational problems, with the aim to develop problem-solving skills, to develop the ability to analyse the efficiency of algorithms, and to encourage a thoughtful approach to the analysis and design of computational problems.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Understand a variety of fundamental data structures and be able to use them effectively in the design and implementation of algorithms.
Understand a variety of fundamental algorithmic problems and algorithms for solving them.
Analyse the space and time efficiency of algorithms and understand how to design efficient algorithms.
Indicative content:

The following are indicative of the topics the module will typically cover:

Analysis of algorithms
Abstract Data Types
Sorting Algorithms
Searching Algorithms
Graphs Algorithms
String-processing Algorithms
Requisites:

To be eligible to select this module as optional or elective, a student must be registered on a programme and year of study for which it is a formally available.

Students are expected to have basic experience of programming in Python. Resources for self-study and Programming Tutor support will be provided in advance of the module.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 2 Undergraduate (FHEQ Level 4)
Teaching and assessment
Mode of study
In person
Methods of assessment
80% Exam
20% Group activity
Mark scheme
Numeric Marks

INTRODUCTORY MATHEMATICS FOR COMPUTER SCIENCE

Description
Aims:

The module aims to provide a grounding in mathematics and statistics most relevant to a Computer Science undergraduate degree.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Apply concepts in mathematics and statistics to frame problems and reason about them.
Indicative content:

The following are indicative of the topics the module will typically cover:

Mathematics:

Trig, exp and log.
Polynomials.
Limits and continuity.
Differential calculus of one and two variables.
Integration.
Series summation and power series.
Complex numbers.
Vector spaces.
Matrices.
Linear Algebra.
Statistics:

Probability, descriptive statistics.
Discrete random variables.
Continuous random variables.
Statistical estimation.
Hypothesis testing.
Requisites:

To be eligible to select this module as optional or elective, a student must be registered on a programme and year of study for which it is a formally available.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 2 Undergraduate (FHEQ Level 4)
Teaching and assessment
Mode of study
In person
Methods of assessment
100% Exam

Mark scheme
Numeric Marks

DISCRETE MATHEMATICS FOR COMPUTER SCIENTISTS

Description

Aims:

The module aims to equip first year computer science students with knowledge of foundational mathematics and logic that will be needed for future computer science modules. To provide students with basic tools and skills for mathematical problem solving, proof and refutation.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Analyse and solve typical problems in discrete mathematics and logic.
Identify and reason with the logical content of arguments.
Carry out standard mathematical proofs and refutations.
Indicative content:

The following is indicative of the topics the module will typically cover:

The first part of the module will focus on foundational discrete mathematics, including but not necessarily limited to: functions and relations, permutations, group theory, set theory, cardinalities, diagonalisation, linear algebra and combinatorics. The module continues with mathematical reasoning, logical notation and proof by mathematical induction.

Requisites:

To be eligible to select this module as an optional or elective, a student must: (1) be registered on a programme and year of study for which it is a formally available; and (2) have passed A-level Mathematics (or an appropriate equivalent).

Module deliveries for 2024/25 academic year
Intended teaching term: Term 1 Undergraduate (FHEQ Level 4)
Teaching and assessment
Mode of study
In person
Methods of assessment
100% Exam
Mark scheme
Numeric Marks

ENGINEERING CHALLENGES
Description
The challenges of the 21st century require more than just theoretical knowledge in order to solve them. Design, professional and technical skills are sorely needed to help communities tackle big problems in crucial areas such as energy, infrastructure and security, as well as health and sustainability.

The aim of the Engineering Challenges module is to introduce you to real-world project-based working as well as some of the techniques and skills that you will need both in the rest of your degree and beyond. The module is split into two design projects, known as Challenges. Each Challenge is designed to address part of the design process and provide you with opportunities to integrate knowledge and skills developed in other modules such as experimental methods, modelling and prototyping as well as developing and improving skills such as creative thinking, critical thinking, decision-making, problem-solving, designing, effective team working, self-directed learning, technical writing, and communication.

Challenge 1, the first design project, is a department-based individual project on a topic relevant to the research conducted within the department of your chosen degree programme. The aim here is to introduce you to your department and give you a sense of the wider context and application beyond the classroom. You will also start to learn skills like information gathering, giving and receiving feedback and presenting your work. Challenge 2, the second design project, is interdisciplinary, with departments paired up or grouped together to work on aspects of one large project where you will also start to think about the social impact of your engineering solutions, whilst designing it to technically work. In this hands-on and active module, you will learn to apply teamwork and communication skills as well as basic project management tools/practices.

This Engineering Challenges module is part of the Engineering faculty's cross-faculty teaching framework called the Integrated Engineering Programme (IEP) and is designed to be integrated with the ENGF0002 DPS and ENGF0003 MMAI modules taught in parallel during the same term. This means that material and skills will be referenced and referred to across the three modules.

Intended Learning Outcomes

Throughout this module you'll gain a better understanding of your discipline and its context within the wider world, the design process and the interaction between society and engineering. You will start developing your skills and abilities in information gathering, engineering design, communication, creative and critical problem solving, decision making and teamwork in groups that are multi-disciplinary in addition to those that only comprise colleagues from your chosen discipline.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 1 Undergraduate (FHEQ Level 4)
Teaching and assessment
Mode of study
In person
Methods of assessment

30% Coursework
50% Group activity
10% Other form of assessment
10% Dissertations, extended projects, and projects
Mark scheme
Numeric Marks

DESIGN AND PROFESSIONAL SKILLS I

Description

This module is intended to introduce you to engineering, design, technical communication and academic writing, as well as other engineering professional practices and skill sets necessary for your future employability. Technical communication in most cases can be broken down into writing, technical argument, and explanation, data visualisation as well as presentation. Specific components will include (but are not limited to) the following: an introduction to your chosen engineering discipline, the design process, critical and creative thinking, decision-making, problem solving, ethics and sustainability, industry standards and professional conduct, teamworking, leadership, project management, career preparation and employability. It will aim to deliver these through instruction, engagement, practice, and feedback. The module is taught through the context or application of your own chosen engineering discipline, computer science, which is supported through the teaching of technical skills required by the profession and industrial practices. Moreover, this module includes two intensive team-based projects, known as Scenarios, which are designed to help you practice the aforementioned technical and professional skills alongside developing your technical understanding and practice of computer science.

This Design and Professional Skills I module is part of the engineering faculty's cross-faculty teaching framework called the Integrated Engineering Programme (IEP) and is designed to be integrated with the ENGF0001 Engineering Challenges module taught in parallel during Term 1. This means that content and skills will be referenced and referred to across these two modules.

Intended Learning Outcomes

By the end of the module, you should be able to:

outline the basic elements of the design cycle, understand the underlying processes, and use these to tackle real engineering problems

describe an engineering problem and its constraints in a concise written or spoken report.

give examples of ethical issues related to engineering and recall tools that can be used to analyse future problems as they arise

give examples of how your specific discipline interacts with sustainability and describe their responsibilities as an engineer in this regard

acknowledge the basic tools for critical thinking and problem-solving and use these to tackle real-world problems, whether engineering or otherwise

identify and describe the utility of important structural features in writing and presentation (such as introductions, conclusions, and topic sentences), use these effectively, and evaluate your use in others' work

analyse the audience for a given communication (report, video or talk/presentation) and determine the appropriate point of view, level of detail, and jargon describe and practice basic concepts in team-working, project management and engineering leadership, and recall the facilities available to you to develop your abilities in this area

recognise the level of conduct expected of them by the body governing their profession, explain why such codes of conduct are necessary; and recall where they are provided for all to witness

report/present on discipline specific material such as programming with Python

Module deliveries for 2024/25 academic year
Intended teaching term: Terms 1 and 2 Undergraduate (FHEQ Level 4)
Teaching and assessment
Mode of study
In person
Methods of assessment
50% Coursework
10% Viva or oral presentation
40% Group activity
Mark scheme
Numeric Marks

Year 2
COMPUTER ARCHITECTURE AND CONCURRENCY
Description
Aims:

The module aims to provide a working knowledge of the hardware and architecture of a modern computer system, particularly focusing on concurrency aspects and those that have an impact on writing multithreaded software. Students will gain a fundamental understanding of the concurrency abstraction and how it impacts both computer architecture and software design. We will look at computer architecture aspects that directly impact multithreaded software such as the memory hierarchy, cache coherence/consistency and hardware multithreading. We will learn how to design correct multithreaded Java software based on a solid theoretical understanding of concurrency principles and the Java Memory Model. The module will cover an understanding of concurrency from low-level aspects (such as spin locks implemented in assembly language) to high-level design patterns used within the Java concurrency package.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Use the concurrency abstraction to reason about concurrent systems.
Write and understand modern assembly language programs.
Describe how high-level languages are translated to assembly language and subsequently machine code.
Describe the internal structure of processors and different forms of parallelism employed.
Describe how modern computers handle memory and input/output including key concurrency aspects such as cache coherence, memory consistency/visibility and interrupt-driven thread switching.
Describe how operating systems schedule application level threads onto the CPU resources.
Describe how synchronization mechanisms in high-level languages are implemented in terms of more primitive hardware concurrency instructions.
Reason about and resolve safety aspects of multithreaded Java including interference and visibility issues.
Write safe and efficient multithreaded Java code using the monitor design and other patterns.
Compare and contrast the traditional Java concurrency mechanisms with those of the new Java concurrency package mechanisms.
Correctly and safely use the thread-safe data and control structures within the new Java concurrency package.
Indicative content:

The following are indicative of the topics the module will typically cover:

Top-down high level overview of a computer: the main components making up a computer and how they interact.
The concurrency abstraction.
Assembly language and machine code.

Translation of high-level languages into machine code.
Internal structure of a processor: the control unit (CU) and datapath.
Parallelism within the CPU.
Key aspects of how operating systems handle application-level threads.
Memory hierarchy, cache structure and cache coherence mechanisms.
Creating and managing Java threads.
Understanding and reasoning with the Java Memory Model (JMM) specification.
Java traditional synchronization mechanisms (and how they relate to low-level hardware instructions)
Monitor design patterns and conditional variables.
Reasoning about the correctness of concurrent programs: safe data structures.
Requisites:

To be eligible to select this module as optional or elective, a student must: (1) be registered on a programme and year of study for which it is a formally available; (2) understand basic computer arithmetic (binary/hex manipulation, fixed-size arithmetic, 2s- and 16s-complement forms, etc.); and (3) have strong Java programming skills.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 1 Undergraduate (FHEQ Level 5)
Teaching and assessment
Mode of study
In person
Methods of assessment
100% Exam
Mark scheme
Numeric Marks

LOGIC
Description
Aims:

The module aims to introduce and familiarise students with logical and mathematical inference. Students learn a number of logical inference methods for classical logics and for non-classical logics.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Understand how axiomatic systems can be used for propositional and predicate logic.
Understand the notions of soundness and completeness.
Understand how propositional and predicate tableaus work.
Have familiarity with other logics, including modal and temporal logics.
Analyse algebras of relations.
Indicative content:

The following are indicative of the topics the module will typically cover:

Propositional logic, Predicate logic, Modal Logic and Temporal Logic:

Review of syntax and semantics.
Deduction and Inference.
Truth tables.
Decidability of propositional logic.
Mathematical proofs:

Proof by contradiction.
Induction and structured induction.
Hilbert systems.
Axioms and inference rules for propositional logic.
Axioms and inference rules for predicate logic.
Axioms and inference rules for modal and temporal logics.
Tableau construction for propositional logic, predicate logic, modal logics.
Soundness and completeness theorems for first order logic.
Semi-decidability of first order logic.
Undecidability of arithmetic.
Algebras of Relations:

Algebras of binary relations
Kleene Algebra
Relation Algebra
Other Algebras of Relations.
Requisites:

To be eligible to select this module as optional or elective, a student must: (1) be registered on a programme and year of study for which it is a formally available; (2) have taken Theory of Computation (COMP0003) and Algorithms (COMP0005); and (3) have some programming experience (as the assessment will require them to implement a program in C).

Module deliveries for 2024/25 academic year
Intended teaching term: Term 1   Undergraduate (FHEQ Level 5)
Teaching and assessment
Mode of study
In person
Methods of assessment
95% Exam
5% Coursework
Mark scheme
Numeric Marks

SOFTWARE ENGINEERING
Description
Aims:

This module focusses on designing and building software systems, which these days are often large, complex and long-lived. They are worked on by teams of engineers and changed constantly over their lifetimes. We will look at principles and patterns of software design, where to apply them, and how they may inform our design choices. We will also look at techniques for ensuring that systems you build behave correctly. We show how the application of these makes it possible to evolve systems effectively in a rigorous way.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Appreciate the wider engineering context that forms the background to developing complex, evolving software-intensive systems.
Apply a range of design patterns and principles to solve particular design problems.
Apply a range of refactoring techniques to improve code quality.
Use a range of tools and techniques for automated software testing, including test-driven development.
Manage risk in making changes to an existing software system through rigorous engineering practices.
Reflect on the appropriateness of different software engineering techniques in different circumstances, and on the quality of the design of an application.
Intended learning outcomes:

On successful completion of the module, a student will be able to:

Object Oriented Design Principles.
Design Patterns.
Software Architecture.
Testing.
Tools and Processes.
Requisites:

To be eligible to select this module as optional or elective, a student must: (1) be registered on a programme and year of study for which it is a formally available; and (2) have taken a module or course in Java programming.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 1	Undergraduate (FHEQ Level 5)
Teaching and assessment
Mode of study
In Person
Methods of assessment
50% Group activity
50% Coursework

Mark scheme
Numeric Marks

INTERMEDIATE MATHEMATICS FOR COMPUTER SCIENCE
Description
Aims:

This module aims to provide a grounding in mathematics and statistics most relevant to a Computer Science undergraduate degree.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Reason about series of functions and multivariate functions.
Employ numerical methods to solve mathematical problems.
Decompose and study the properties of matrices and vector spaces.
Model probabilities using measure theory.
Use advanced statistical methods to reason about hypotheses.
Indicative content:

The following are indicative of the topics the module will typically cover:

Mathematics:

Sequences and Series of Functions.
Multivariate Differential Calculus.
Optimisation.
Numerical Methods.
Eigenvalues and Matrix reduction.
Euclidian Spaces and Orthogonal Matrices.
Statistics:

Continuous Random Variables.
Joint Probabilities and Marginal Distributions.
Hypothesis Testing, Bootstrapping.
Requisites:

To be eligible to select this module as optional or elective, a student must be registered on a programme and year of study for which it is formally available.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 2   Undergraduate (FHEQ Level 5)
Teaching and assessment
Mode of study
In person
Methods of assessment
80% Exam
20% In-class activity
Mark scheme
Numeric Marks

SYSTEMS ENGINEERING
Description
Aims:

The module aims to take students through the experience of Research and Development (R and D) processes by developing a reference Proof of Concept design and prototype implementation for a complex and substantial software application over an extended period of time.

Students will work with an external project partner to obtain requirements and get feedback for a project partner-defined problem. Experience of working with team organisation and management; develop and work to a plan, with weekly lab checks, project partner reports and deliverable milestones to be met; develop a practical understanding of how the research experimentation and development of a software system is organised and carried out, both in terms of team working skills and the technical knowledge required; practice using relevant skills and knowledge in completing tasks and milestones; develop writing, media production, presentation and communication (technical and customer facing) skills; and learn how to evaluate their work and team and individual performance.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Understand how to work effectively with an external project partner and professional behaviour.
Understand a software development process involving research and experimentation in an area of Computer Science.
Organise a team and work effectively in a team.
Understand how to undertake research to identify and use relevant ideas and technologies when developing software.
Understand how to develop and test good quality software.
Understand Research and Development approaches to requirements capture, HCI factors, and team-based development relevant to the projects undertaken.
Write documentation and present results.
Understand technical writing skills and share their awareness of legal issues as they apply to software development.
Indicative content:

Students will undertake a systems engineering project within the Computer Science Industry Exchange Programme (IXN) structure. The students will work in small teams, where each team works with an external project partner to develop a cutting-edge solution to a challenging problem defined by the project partner. The project partner will typically be from one of the following types of organisation: public sector, SMEs, large companies, charity, healthcare/NHS, external research institution, or UCL research centre.

The project will be structured as a single major project portfolio, with a series of milestones, where each milestone involves completing a stage of the project: requirements, research into potential solutions, experiments and prototypes, definition of reference design, implementation, testing, and delivery. Each of these stages is to be documented on the projects team website. Each portfolio milestone is also a point of assessment.

A number of teaching methods will be employed, including hackathon-style sessions, project sprints in labs and specific lectures. Weekly communications are needed with project partner, with weekly lab checks and project partner reports to track the project progress and team working. An Elevator Pitch to companies is to be given by each team. The final milestone will require teams to deliver a working version of their application, along with a project website documenting the project results. Any publication of results on behalf of the project partner is seen as an advantage.

In addition, each student will write a technical report assessing the work done, the team members, and reflecting on what they have learnt, in particular the area of technologies they have studied. Each team is allocated a PGTA, who they will meet every teaching week to report on progress and get feedback. Each team also has an academic supervisor, who will monitor the overall status of the team, and can provide more specialised support and feedback. The project finishes with delivery of their software to their project partner in a usable state with a joint code walkthrough, an assessed presentation and delivery of all associated materials such as legal implications, manufacturing processes and others. The taught material will be directly relevant to, and needed for, the project work. The overall project work will be submitted as an online project portfolio made up of the milestones.

Requisites:

To be eligible to select this module as optional or elective, a student must: (1) be registered on a programme and year of study for which it is a formally available; and (2) have experience of object-oriented programming.

Module deliveries for 2024/25 academic year
Intended teaching term: Terms 1 and 2 Undergraduate (FHEQ Level 5)
Teaching and assessment
Mode of study
In person
Methods of assessment
100% Group activity
Mark scheme
Numeric Marks

SECURITY
Description
Aims:

The module aims at giving students a foundation in computer security, cryptography and human aspects of security. After taking the module students should be able to identify risks and security sensitive aspects of systems, where a system can range from a small and simple solution for a specific task to a full organisation including IT infrastructure and users. They will be taught this from the experience of how systems have failed in the past and by analysing systems from an adversarial viewpoint.

Students should also know core defensive techniques to prevent or mitigate attacks. They should be able to formulate a threat model for a system and reason about whether it is secure or insecure, Students should be able to deliver effective security for real-world environments: how to design and manage security in organisations to effectively protect them.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Understand how to formulate a threat model including the assets to be protected, the capabilities of the adversary, and reason about whether the probability the adversary may succeed.
Know core cryptographic methods to protect communication such as message authentication codes, digital signatures, encryption, and key exchange. Understand the security guarantees cryptographic tools provide and conversely how inappropriate use lead to vulnerabilities.
Understand key principles and concepts related to computer and systems security.
Understand the complexities of building flexile and correct access control and authentication systems, from a technical and HCI perspective. Recognize and exploit common security vulnerabilities on hosts and networks.
Know that security involves people, processes and technology. Understand how security needs to fit with business goals and organisational processes to work effectively rather than disrupt, create friction, and drain resources.
Know the basics of how to influence and manage security behaviours of a range of organisational stakeholders, employees, managers, developers, security specialists.
Indicative content:

The following are indicative of the topics the module will typically cover:

The module gives students a foundation in computer security, cryptography and human aspects of security. Students will study attacks against system and analyse systems from an attacker's perspective. The module will also cover core defensive techniques and develop threat models that can be used to analyse the security of a system. The module studies specific cryptographic systems, core computer security, and security for real-world environments: how to design and manage security in organisations to effectively protect them.

Requisites:

To be eligible to select this module as an optional or elective, a student must be registered on a programme and year of study for which it is a formally available.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 2   Undergraduate (FHEQ Level 5)
Teaching and assessment
Mode of study
In person
Methods of assessment
100% Exam
Mark scheme
Numeric Marks

Year 3
COMPILERS
Description
Aims:

This is a practical module whose primary goal is develop an understanding of the operation of compilers and the development and specification of computer-based languages. The course pulls together threads from underlying theory, most notably from logic and from data structures and algorithms, and builds on these a practical exercise in which students create a compiler of their own using commonly available compiler development tools.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Build lexical analysers and use them in the construction of parsers.
Express the grammar of a programming language.
Build syntax analysers and use them in the construction of parsers.
Perform the operations of semantic analysis; build a code generator.
Discuss the merits of different optimisation schemes.
Indicative content:

The following are indicative of the topics the module will typically cover:

Anatomy of a compiler:

The importance of compilers.
Structure of a compiler.
Analysis (lexical, syntax and semantic analysis).
Synthesis (intermediate code generation, optimisation and code generation).
Compilers vs. interpreters. Lexical analysis (scanning).
Tokens.
Regular expressions.
Finite state automata (deterministic and non-deterministic).
Translating regular expressions into finite state automata.
Automatic lexer generators (JLex/JFlex).
Syntax analysis (parsing):

Context-free grammars.
Derivations and (concrete/abstract) syntax trees.
Handling ambiguous grammars.
Top-down parsing (LL(k) grammars, recursive descent parsers).
Bottom-up parsing (LR(k) grammars, shift-reduce parsers).
Automatic parser generators (CUP).
Syntactic error recovery.
Syntax-directed translation:

Syntax-directed definitions.

Abstract syntax tree construction.
Semantic analysis:

Symbol table management.
Scoping and type checking.
Basic implementation techniques (Visitor methodology).
Intermediate code generation:

Three address code.
IR instructions.
Translation methodologies.
Code generation and optimisation:

Run-time storage organisation.
A simple code generation algorithm.
Optimisation of intermediate code.
Optimisation of target code (Peephole optimisation).
Requisites:

To be eligible to select this module as optional or elective, a student must be registered on a programme and year of study for which it is a formally available.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 2   Undergraduate (FHEQ Level 6)
Teaching and assessment
Mode of study
In person
Methods of assessment
80% Fixed-time remote activity
10% Coursework
10% Group activity
Mark scheme
Numeric Marks

COMPUTABILITY AND COMPLEXITY THEORY
Description
Aims:

The module addresses the theoretical and practical limitations of computation and provides a mathematical framework for proving properties of programs. The concepts of undecidability and intractability are formally introduced and discussed through a number of examples. The module will convey the proof techniques that are used to classify problems. It is intended that students learn how to apply them to classify unfamiliar problems for themselves.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Analyse the complexity of a variety of problems and algorithms.
Prove that a problem is undecidable.
Prove the equivalence of different models of computation.
Find a polynomial time reduction from one problem to another.
Determine the complexity class of a decidable problem.
Categorise the complexity of a language.
Indicative content:

The following are indicative of the topics the module will typically cover:

Models of Computation:

Deterministic Turing machines.
Alternative models: variations of the concept of Turing machine.
Alternative models: register machines, imperative programming languages.
Languages:

Language acceptance.
Language recognition.
Decidable and Recognisable languages.
Closure properties of languages.
Undecidability:

The Halting Problem and other unsolvable problems.
Problem reduction.
Rice's Theorem.
Cantor's diagonal argument and the cardinality of unrecognisable languages.
Undecidability of the tiling problem.
Undecidability of first-order logic and incompleteness of arithmetic.
Basic Concepts of Complexity Theory:

Tractable and intractable problems and algorithms.
Definition of Time Complexity.

Decision vs Optimisation Problems.
Travelling Salesman and its variants.
Polynomial-time reduction:

Definition and properties.
Lemmas and proofs.
Hamiltonian Graphs and Travelling Salesman.
Complexity classes:

Non-deterministic Turing machines.
P, NP, and NP-complete classes.
Complement and Co-Complete Classes.
Proving NP-hardness:

Cook’s theorem.
Satisfiability and its variants.
Reductions between variants.
Other Complexity Classes:

Space complexity, PSPACE, and PSPACE-Completeness.
Savitch’s theorem.
Logarithmic space.
Time and space hierarchy theorems.
Exponential time and hardness.
Probabilistic classes.
Approximation algorithms and hardness.
Requisites:

To be eligible to select this module as optional or elective, a student must: (1) be registered on a programme and year of study for which it is a formally available; and (2) have passed Theory of Computation (COMP0003), Algorithms (COMP0005), and Logic and Database Theory (COMP0009).

Module deliveries for 2024/25 academic year
Intended teaching term: Term 1 Undergraduate (FHEQ Level 6)
Teaching and assessment
Mode of study
In Person
Methods of assessment
100% Exam
Mark scheme
Numeric Marks

INDIVIDUAL PROJECT FOR YEAR 3 BSC
Description
Aims:

The module aims to give students the experience of undertaking and completing a substantial piece of individual project work, applying knowledge learned throughout the degree programme, including the skills of research, analysis, design, implementation, testing and evaluation. To document the results to a good academic standard via a written report.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Work individually to undertake a major project.
Plan and coordinate the project activities to meet the deadlines.
Make realistic work commitments.
Write a substantial report to document the project results.
Demonstrate good problem solving and evaluation skills.
Indicative content:

The following are indicative of the topics the module will typically cover:

There is no set syllabus; students identify their chosen project topic and find a member of the academic staff who will act as the Project Supervisor.
The project begins in October and runs in parallel with normal teaching during terms 1 and 2. The submission deadline is early in term 3.
Students are responsible for organising and carrying out their project and are expected to attend weekly tutorials with their supervisor during term time. Tutorials allow the supervisor to monitor the progress of the project, providing advice and feedback as necessary.
There are two formative milestones: (i) the Project Plan submitted in mid-November, outlining the project aims and objectives, providing a work plan, and confirming that the project has started successfully, and (ii) the Interim Report submitted in late January outlining what has been achieved so far and what it is required to complete the project. These documents are prepared with advice from the Project Supervisor and must be approved by them to confirm the project is proceeding properly.
Project goals:

A BSc project is expected to have a reasonable level of challenge and depth and may include research-oriented goals as well as the practical application of Computer Science ideas and concepts.
Project report:

The final report documents the results of the project, and is expected to be a substantial, well-written, and properly structured document. Obtaining a higher module mark depends on submitting a good quality report, as well as successfully undertaking the project work. The project guidelines available on Moodle give detailed information about the required content.
Reading list:

As every student undertakes a distinct project, there is no indicative reading list available. The student is expected to locate relevant information sources themselves with advice from their Project Supervisor.
Requisites:

To be eligible to select this module, a student must be registered on a programme and year of study for which it is a formally available.

Module deliveries for 2024/25 academic year
Intended teaching term: Terms 1 and 2   Undergraduate (FHEQ Level 6)
Teaching and assessment
Mode of study
In person
Methods of assessment
90% Dissertations, extended projects, and projects
10% Coursework
Mark scheme
Numeric Marks

MACHINE LEARNING FOR VISUAL COMPUTING

Description
Aims:

The module aims to equip students with knowledge of how to apply AI to problems from the creative industry; familiarity with basic ML-based algorithms and data structures to process digital media; ability of dealing with large scale data and training of machine intelligence; knowing rephrasing of existing concepts from digital media with tools from AI; and awareness of the difficulty of computed results and artistic freedom.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Formulate the diversity of image analysis and synthesis tasks as a machine learning process.
Understand theoretical and practical concepts allowing image processing and generation to become learnable and basic understanding of how to execute that learning.
Indicative content:

Creative industries such as print, feature films, music, fabrication or interactive media increasingly make use of multiple machine learning driven tools. This module enables students to contribute to a new shift of paradigm, where such tools become increasingly intelligent of the content being designed and the users designing them. This is enabled by machine learning, a new way of dealing with data and new forms of algorithms. We will cover an applied background of machine learning and focus on data structures particularly relevant for creative content such as images and video and focus on learnable algorithms that allow to machines to process them intelligently, such as convolutional neural networks.

The following are indicative of the topics the module will typically cover:

Basic regression: Ability to explain basic (1D) data with a linear model and to make predictions. Knowledge of how what a linear model entails, also in higher dimensions. The resulting attitude could be both that a surprisingly simple thing can explain complex visual phenomena, yet others it cannot.
Understanding of linearity and non-linearity: Participants will be able to judge if a linear or nonlinear model is adequate and how important non-linearity is for Visual Computing. They should have knowledge of how to represent such models. The key change in perspective is that almost all non-trivial processes are nonlinear.
Classification: Ability to formalize a problem as classification code a simple classifier in. Knowledge that classification is regression over probabilities to be from a class.
Neural networks: Students should be able to code up a simple perceptron from scratch and manipulate its parameters (e.g., number of units and layers), as well as control the non-linearities. They would understand that NN is just matrix multiplications followed by non-linearities.
Audio/2D/3D images and pixel processing: Load and stored images/3D data, audio in the coding environment they use. Understand basic issues of sampling and representations like

colour and luminance. The attitude should be changing such that not only the algorithm matters, but also the right representation is critical.
Tunable image filters; convolutional neural networks: Participants will learn how to execute convolutions on images and how to set up their environment to optimize the parameters of the convolutions. The key knowledge is that learning convolutions allows to share learned parameters across the image.
3D meshes and point clouds: Until now data structures were regular, but now students will learn how to load irregular data (3D graphs, 3D meshes, 3D point clouds) into their environment and also how to process them as well as to learn tunable filters for them. They will deepen their knowledge of convolution by seeing how it extends to unstructured data, a generalization. The resulting attitude should be, that a well-defined convolution works on regular data as well as irregular ones.
Ambiguity and style: Students will be explained, that under some conditions, multiple solutions are valid and that it can be adequate to report any of these. They will see how to change their code from working on paired data matching an output to a reference, it can be sufficient to match only statistics. The required skill is to change the loss and choose the right representation of these statistics.
Generative modelling: We will first introduce the concept of a generative model, which takes simple parameters in low dimensions and maps them to complex objects in millions of dimensions. Using this requires the skill to load many exemplars and encode and decode them, which students can compose from components from before. The change of attitude should be, that a generative model is a very abstract way to “address” objects in a family of natural instances: images of faces, houses, cars, etc.
Levels of supervision; Adversarial training: Students here will acquire the skill to replace the supervision in form of pairs that are mapped to each other by a paradigm where only random samples form a target distribution are given. To this end they have to understand that the loss is replaced by another network.
Requisites:

To be eligible to select this module an optional or elective, a student must be registered on a programme and year of study for which it is a formally available.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 1 Postgraduate (FHEQ Level 7)
Teaching and assessment
Mode of study
In person
Methods of assessment
50% Coursework
50% Viva or oral presentation
Mark scheme
Numeric Marks

ADVANCED MATHEMATICS FOR COMPUTER SCIENCE
Description
Aims:

This module aims to provide students with advanced mathematical tools and to (1) consolidate knowledge in Algebra analysis, Differential Calculus, Integral Calculus, Probabilities and Statistics; (2) introduce advanced methods making use of this knowledge in a practical manner; and (3) connect these notions with subsequent Computer Science modules.

Intended learning outcomes

On successful completion of this module, a student will be able to:

Know and understand key results in the areas of Algebra, Calculus, Probability and Statistics.
Apply problem solving methods which make use of the key results in the areas of Algebra, Calculus, Probability and Statistics.
Apply these methods in modern areas of Computer Science such as: Artificial Intelligence, Computational Finance, and Computer Graphics.
Indicative content:

The following are indicative of the topics the module will typically cover:

Optimisation.
Convexity.
Advanced Linear Algebra and Matrix Reduction.
Number Theory and Arithmetic.
Statistics and Sampling.
Probabilistic Processes.
Requisites:

To be eligible to select this module as optional or elective, a student must be registered on a programme and year of study for which it is formally available.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 2 Undergraduate (FHEQ Level 6)
Teaching and assessment
Mode of study
In person
Methods of assessment
80% Exam
20% In-class activity
Mark scheme
Numeric Marks

COMPUTER SYSTEMS
Description
Aims:

The aim for this module is to have our advanced undergraduates develop a “whole-stack” understanding of the complex interactions among modern computer system hardware (specifically, a modern, multi-core CPU and its memory hierarchy), a modern operating system, and the applications that run atop them. These interactions dictate application correctness and performance. These fundamental principles will be taught in the concrete framework of the x86 CPU architecture and the Linux/UNIX operating system. Students should not only understand these fundamental principles, but be able to put them into practice in real C programs.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Understand the C and Linux/UNIX programming environment, from the hardware (memory hierarchy, memory model between cores) to low-level operating system functionality (file and network I/O, process management, virtual memory system, program linking and loading).
Understand the interactions between hardware and the C and Linux/UNIX programming environment that affect application correctness and performance.
Design correct applications that perform well in the C and Linux/UNIX programming environment.
Indicative content:

The module teaches students topics in computer systems design and systems programming, with an emphasis on the C programming language, Intel CPU architecture, and UNIX/Linux operating system architecture. The following are indicative of the topics the module will typically cover:

Machine-level representation of code and data (assembly language; how C constructs are compiled into x86 assembly language; machine-level representations of data types and machine and C arithmetic; stack frame structure in C on the x86; array layout in memory; pointer arithmetic).
Use of the gdb debugger on C code in the Linux/UNIX environment.
Undefined behaviour in the C language specification and real C programs, and its consequences.
The memory hierarchy: registers, L1 caches, L2 caches, L3 caches, DRAM, SSD, disk; cache architectures; performance implications of caches on program performance.
Linking: static linking, relocatable object code, symbol resolution, loading executables, dynamic linking and shared libraries, position-independent code, interposition.
Processes, signals, context switches, exceptions, system call error handling, job control, process creation and management.
Virtual memory: address translation, memory mapping, dynamic memory allocation, memory-related bugs in C;
System I/O: filesystem I/O, system call interface and semantics.

Concurrent systems programming: threads in C on Linux/UNIX, x86 memory consistency model, races, deadlocks.
Requisites:

To be eligible to select this module as optional or elective, a student must: (1) be registered on a programme and year of study for which it is a formally available; (2) have basic proficiency in C programming (including dynamic memory allocation); (3) have basic knowledge of computer arithmetic (e.g., two's complement integer representation); and (4) have prior exposure to assembly language for some CPU.

UCL Computer Science undergraduates will normally have met these prerequisites by having passed the relevant compulsory modules in prior years.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 2   Undergraduate (FHEQ Level 6)
Teaching and assessment
Mode of study
In Person
Methods of assessment
60% Exam
40% Coursework
Mark scheme
Numeric Marks

FUNCTIONAL PROGRAMMING
Description
Aims:

This module explores the functional programming paradigm and the implementation technology for functional programming languages. It aims to develop a broad understanding of the functional programming style and recursive programming techniques using the language Miranda, together with an understanding of implementation issues that are relevant not only to functional languages but also to other systems that require automatic dynamic memory management.

The module explores the underlying mechanics of strict and lazy functional languages; it does not use Haskell or F# or OOCAML and does not aim to provide training in such languages, though an introduction to Miranda is provided and students are expected to improve their functional programming skills through independent study. In the second half of the module students are expected to use independent study to read extensively about implementation issues which are then discussed in the lectures.

Intended learning outcomes:

On successful completion of the module, a student will be able to (at a level commensurate with FHEQ level 6):

Understand the basics of the lambda calculus and combinators and how they are used in the implementation of functional languages.
Understand the main features of a lazy functional language.
Understand type checking, type-inference and the operation of the Miranda (Hindley-Milner) type system.
Write and understand non-trivial functional programs in Miranda.
Understand the computation and memory management issues affecting the sequential implementation of lazy functional languages.
Solve problems relating to any aspect of the module's syllabus, under examination conditions.
Indicative content:

The following are indicative of the topics the module will typically cover:

Classification of programming languages:
Distinctive features of Functional Programming Languages.
The Lambda Calculus and Combinators:
Versions of the Lambda Calculus.
Syntax and semantics.
Reduction orders, strong normalisation.
Combinators and computationally complete sets.
Introduction to Miranda:
Programming environment.
Types and simple polymorphic types.
Recursion.

Pattern-matching.
Lists.
Higher-Order functions.
User-defined types.
Type polymorphism and type systems.
Recursive programming techniques.
Introduction to implementation techniques:
Strict evaluation and lazy evaluation.
The need for automatic memory management.
Automatic memory management:
Garbage collection techniques.
Requisites:

To be eligible to select the module delivery for Undergraduate (FHEQ Level 6) as optional or elective, a student must be registered on a programme and year of study for which it is formally available.

To be eligible to select the module delivery for Postgraduate (FHEQ Level 7) as optional or elective, a student must both: (1) be registered on a programme of study for which it is formally available; and (2) have either taken all Term 1 modules of the MSc Computer Science programme at UCL or have studied the following at FHEQ level 6 or higher:

Programming in one high-level programming language and one assembly language.
Formal systems of logic such as Boolean algebra, propositional logic or predicate calculus.
Virtual machines, virtual memory and memory paging.
Compilers, including lexical analysis, parsing and code generation.
Dynamic data structures and abstract data types.
Models of storage in computer systems.
Algorithmic complexity.
Students must be proficient in the English language to UCL's Level 4 standard or better.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 2   Postgraduate (FHEQ Level 7)
Teaching and assessment
Mode of study
In person
Methods of assessment
100% Exam
Mark scheme
Numeric Marks

INTERACTION DESIGN
Description
Aims:

The module covers advanced topics in interaction design, informed by current research in user experience, human-computer interaction and interaction design. A central theme is how to design technologies to meet people's needs.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Have knowledge and understanding of research in interaction design.
Have knowledge and understanding of methods used in interaction design.
Reflect critically on the appropriateness of different interaction design methods.
Conduct basic user research.
Design, prototype and evaluate a novel ubiquitous computing technology.
And will have developed the following transferrable skills:

Information gathering and organising skills.
Critical reading skills.
Argumentation skills and the ability to synthesis information from multiple sources.
Written, visual and video presentation skills.
Indicative content:

The module is separated into two related streams. The following are indicative of the topics the module will typically cover:

Research (weekly): This series of e-lectures will introduce students to core (current and historical) and nascent interaction design methods and technologies, including approaches to conducting user research, designing, prototyping and evaluating user-centred systems reviewing evolving interaction technologies.

Workshops (weekly): These more practical sessions will give students an opportunity to put interaction design methods into practice. They will link closely to the coursework.

Requisites:

To be eligible to select the module delivery Undergraduate (FHEQ Level 6) as optional or elective, a student must be registered on a programme and year of study for which it is a formally available.

To be eligible to select the module delivery for Postgraduate (FHEQ Level 7) as optional or elective, a student must: (1) be registered on a programme and year of study for which it is a formally available; and (2) have some prior knowledge of HCI or interaction design.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 2 Postgraduate (FHEQ Level 7)

Teaching and assessment
Mode of study
In person
Methods of assessment
48% In-class activity
52% Coursework
Mark scheme
Numeric Marks

DATABASE AND INFORMATION MANAGEMENT SYSTEMS

Description
Aims:

Students will learn fundamental theories and methods of database systems: what they are, how they are developed and how they function to achieve their purposes. The module will exemplify these constructs with contemporary database technologies and students will learn how these technologies are exploited to build effective information systems of different scale.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Understand the theories and methods of database systems.
Apply the theories and methods of database systems in creating effective information systems.
Understand current and emerging database technologies and the application settings in which they are used.
Indicative content:

The following are indicative of the topics the module will typically cover:

The Database Management System.
Transaction management and failure management.
Query processing and optimization.
Distributed databases.
Data warehousing.
Data mining and online analytic processing.
Semi-structured data databases.
Emerging database models and technologies.
Database implementation and optimisation.
Requisites:

To be eligible to select this module as optional or elective, a student must be registered on a programme and year of study for which it is a formally available.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 2 Undergraduate (FHEQ Level 6)
Teaching and assessment
Mode of study
In person
Methods of assessment
50% Exam
50% Coursework
Mark scheme
Numeric Marks

NETWORKED SYSTEMS
Description
Aims:

The module offers a rigorous introduction to the problems that arise when networking computer systems, and algorithms and systems design that solve these problems. The architectural principles and protocols that underlie the internet will be explained in detail. Topics to be taught will include the physical layer, widely used link layers (wired and wireless), MAC protocols, internetworking, intra-domain routing, reliable transport, congestion control, wide-area (policy) routing, naming, network security, the end-to-end principle, and network applications.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Understand the nuances of interactions between a network's distributed entities.
This is a vital skill, without which one cannot truly be said to understand networking. To help students develop this skill, the module incorporates significant programming courseworks in Python and Java, in which students build or interact with working implementations of network protocols.

Indicative content:

The following are indicative of the topics the module will typically cover:

Introduction to Networking; Information, Error Control Coding, and Compression.
Medium Access Control: CDMA, ALOHA, and Ethernet; Link Layer Addressing and Forwarding; Spanning Trees.
Achieving Reliability; Stop-and-Wait, Go-Back-N.
Achieving Reliability (cont); Selective Repeat; Intro to Internetworking; Network Address Translation; Inside Internet Routers.
The Domain Name System; Multi-hop Networks and end-to-end Arguments.
Reliable Transport and TCP.
TCP and Congestion Control.
Intra-Domain Routing; Introduction, Distance-Vector, Link-State.
Inter-Domain Routing; BGP.
Wireless Networks; 802.11 MAC.
Security: Firewalls, Worms and IDSes; Content Delivery HTTP, Web Caching, and Content Distribution Networks.
Requisites:

To be eligible to select this module as optional or elective, a student must be registered on a programme and year of study for which it is a formally available.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 1   Undergraduate (FHEQ Level 7)
Teaching and assessment

Mode of study
In person
Methods of assessment
60% Exam
40% Coursework
Mark scheme
Numeric Marks

ARTIFICIAL INTELLIGENCE AND NEURAL COMPUTING

Description
Aims:

This module introduces artificial intelligence and neural computing as both technical subjects and as fields of intellectual activity. The overall aims are to introduce basic concepts of artificial intelligence for reasoning and learning behaviour; and to introduce neural computing as an alternative knowledge acquisition/representation paradigm, to explain its basic principles and to describe a range of neural computing techniques and their application areas.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Identify problems that can be expressed in terms of search problems or logic problems, and translate them into the appropriate form, and know how they could be addressed using an algorithmic approach.
Identify problems that can be expressed in terms of neural networks, and to select an appropriate learning methodology for the problem area.
Indicative content:

The following are indicative of the topics the module will typically cover:

Artificial intelligence:

Nature of artificial intelligence.
Searching state spaces.
Utility theory.
Logic for artificial intelligence.
Argumentation.
Reasoning about concepts.
Reasoning about uncertainty.
Machine learning.
Common-sense reasoning.
Neural Computing:

Overview of network architectures and learning paradigms.
Fully connected networks.
Convolutional Neural Networks.
Networks Dealing with Sequential Data.
(Deep) Reinforcement Learning.
Meta-Learning.
Requisites:

To be eligible to select the module delivery for Undergraduate (FHEQ Level 6) as optional or elective, a student must: (1) be registered on a programme and year of study for which it is a formally available; and (2) have passed a module in propositional and predicate logic at FHEQ level 4 or higher.

To be eligible to select the module delivery for Postgraduate (FHEQ Level 7) as optional or elective, a student must: (1) be registered on a programme and year of study for which it is a formally available; (2) have a bachelor’s degree or higher in mathematics or philosophy; (3) have passed a module in propositional and predicate logic at FHEQ level 4 or higher; and (4) have pre-approval to select the module from the Module leader.

Module deliveries for 2024/25 academic year

Intended teaching term: Term 2	Postgraduate (FHEQ Level 7)

Teaching and assessment
Mode of study
In person
Methods of assessment
90% Exam
10% Coursework
Mark scheme
Numeric Marks

INTRODUCTION TO CRYPTOGRAPHY
Description
Aims:

Cryptography is the practice and study of techniques for secure communication in the presence of adversarial entities. For many centuries the goal of cryptography was the protection of privacy of communications. However, computers, digital communication and in particular the Internet have brought an abundance of new security objectives such as anonymity, authenticity, non-repudiation, authorized wiretapping (called law enforcement), or traceability. These developments have turned cryptography into an indispensable tool to protect the information in computing systems and it is utilized nowadays by billions of people on a daily basis to safeguard their data at rest and in transit.

The goal of this module is to make students familiar with the basic paradigms and principles of modern cryptography. In particular, students will be confronted with a range of security objectives, different levels of security that can be achieved, and some of the available cryptographic techniques that can be used to achieve these goals.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Model security precisely and formally in terms of adversarial objective and system access.
Explain and reason about basic cryptographic tools to protect and authenticate data.
Suggest security parameters that protect against standard attacks.
Read scientific articles and international standards in the field of cryptography.
Indicative content:

The following are indicative of the topics the module will typically cover:

Classical ciphers:

Cryptanalysis of classical ciphers.
Probability theory.
Perfect security; Block ciphers.
DES.
AES.
Block cipher modes of operation.
Private-key encryption:

Chosen plaintext attacks.
Randomised encryption.
Pseudorandomness.
Chosen ciphertext attacks.
Message authentication codes:

Private-key authentication.
CBC-MAC.

Pseudorandom functions.
CCA-secure private-key encryption.
Hash functions:

Integrity.
Pre-image resistance.
Collision-resistance.
SHA-256.
NMAC/HMAC.
Key distribution:

Key distribution centres.
Modular arithmetic and group theory.
Diffie-Hellman key exchange.
Public-key distribution:

ElGamal encryption.
Cramer-Shoup encryption.
Discrete logarithm problem.
Digital signatures:

RSA signatures.
RSA-FDH and RSA-PSS signatures.
DSA signatures.
X.509 certificates.
Certification paths.
Requisites:

To be eligible to select the module delivery for Undergraduate (FHEQ Level 6) as optional or elective, a student must be registered on a programme and year of study for which it is a formally available.

To be eligible to select the module delivery for Undergraduate (FHEQ Level 7) as optional or elective, a student must be registered on a programme and year of study for which it is a formally available.

To be eligible to select the module delivery for Postgraduate (FHEQ Level 7) as optional or elective, a student must: (1) be registered on a programme and year of study for which it is a formally available; and (2) have successfully completed a module in mathematics or probability theory at FHEQ level 6 or higher.

The module is theoretically and mathematically demanding.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 1 Undergraduate (FHEQ Level 7)
Teaching and assessment
Mode of study
In person

Methods of assessment
75% Exam
25% Coursework
Mark scheme
Numeric Marks

IMAGE PROCESSING
Description
Aims:

This module focuses on digital image processing. It first introduces the digital image, with a description of how digital images are captured and represented. It then goes on to cover algorithms for image segmentation and feature extraction in direct space. The module then proceeds to cover image filtering techniques with some indication of the role and implications of Fourier space, and more advanced characterisation and feature detection techniques such as edge and corner detection, together with multi-resolution methods, template matching and optical flow techniques.

The module has a strong practical component that allows students to explore a range of practical techniques by implementing their own image processing tools using Matlab or Python.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Understand (i.e., be able to describe, analyse and reason about) how digital images are represented (in the spatial and frequency domain), manipulated, encoded and processed, with emphasis on algorithm design, implementation and performance evaluation.

Implement a variety of image processing algorithms including image manipulation, segmentation, filtering, blending, feature extraction and description, edge detection, template matching and image editing.

Indicative content:

The following are indicative of the topics the module will typically cover:

Introduction to digital image processing:

Digital image capture.

Data types and 2D representation of digital images.

Discrete sampling model.

Noise processes.

Segmentation:

Thresholding and thresholding algorithms.

Performance evaluation and ROC analysis.

Connected components labelling.

Clustering algorithms.

Graph based methods.

Image transformations:

Grey level transformations.

Histogram equalization.

Geometric transformations.

Affine transformations.

Warps.

Image filtering:

Fourier analysis.

Linear and non-linear filtering operations.

Image convolutions

Separable convolutions.

Aliasing, sub-sampling and interpolation.

Laplacian pyramids.

Edge and corner detection:

Edge detection.

Image structure tensor.

Image auto-correlation.

Harris corner detector.

Sift corner detector/descriptor.

Template matching and advanced topics:

Similarity and dissimilarity matching metrics.

Template matching.

Optical flow.

Non-local means filtering.

Poisson image editing.

Requisites:

To be eligible to select the module delivery Undergraduate (FHEQ Level 6) as optional or elective, a student must be registered on a programme and year of study for which it is a formally available.

To be eligible to select the module delivery Postgraduate (FHEQ Level 7) as optional or elective, a student must: (1) be registered on a programme and year of study for which it is a formally available; and (2) have a bachelor’s degree or higher in a physical science or engineering subject with significant mathematical and programming content.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 1 Postgraduate (FHEQ Level 7)
Teaching and assessment
Mode of study
In person
Methods of assessment
40% Coursework
60% Clinical, laboratory or practical activity
Mark scheme
Numeric Marks

COMPUTER GRAPHICS
Description
Aims:

Understanding the theoretical concepts behind computer that generate pictures from virtual worlds and teaching practical skills to implement them.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Perform operations to 3d primitives using linear algebra.
Write a simple ray-tracer to intersect primitives with rays through pixels.
Understand how all steps of rasterization work and being able to code that from scratch.
Understand the concept of Monte Carlo simulation and how it relates to image synthesis.
Implement Monte Carlo simulations.
Use scene graphs, including light, material and textures to describe virtual 3D worlds.
Indicative content:

The following are indicative of the topics the module will typically cover:

Introduction:

The painter's method.
Creating an image using ray tracing:

Ray casting using a simple camera.
Local illumination.
Global illumination with recursive ray tracing.
Specifying a general camera:

World / image coordinates.
Creation of an arbitrary camera.
Ray tracing with an arbitrary camera.
Constructing a scene:

Scene hierarchy.
Transformations of objects / rays.
Other modelling techniques.
Acceleration Techniques:

Bounding volumes.
Space subdivision.
From ray tracing to projecting polygons:

Graphics pipeline.
Transforming the polygons to image space.
Sutherland-Hodgman clipping.

Scan conversion.
Z-buffering.
Interpolated shading.
Texture mapping.
Back-face culling.
Shadows:

Shadow volumes.
Shadow buffer.
Shadow mapping.
Soft shadows.
The nature of light:

Transport theory, Radiance, luminance, radiosity.
The radiance equation.
Photon mapping.
Monte Carlo integration.
Parametric curves and surfaces:

Bezier Curves.
B-Splines Curves.
Parametric surfaces.
Requisites:

To be eligible to select the module delivery Undergraduate (FHEQ Level 6) as optional or elective, a student must: (1) be registered on a programme and year of study for which it is a formally available; and (2) have understanding of linear algebra and basic programming.

To be eligible to select the module delivery Postgraduate (FHEQ Level 7) as optional or elective, a student must: (1) be registered on a programme and year of study for which it is a formally available; and (2) have understanding of linear algebra and basic programming.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 1 Postgraduate (FHEQ Level 7)
Teaching and assessment
Mode of study
In person
Methods of assessment
100% Coursework
Mark scheme
Numeric Marks

TECHNOLOGY ENTREPRENEURSHIP
Description
Aims:

This is UCL Computer Science’s principal entrepreneurship module for students seeking to develop and test a new business idea. Over the past ten years we have taught entrepreneurship to thousands of students, resulting in the launch of several innovative businesses.

The module covers the new business lifecycle (selecting and testing a moneymaking idea, preparing a business plan, raising finance, the Exit), aspects of new business operation (registering a company, setting up your office, understanding financial statements), and exploiting new e-commerce tools and techniques (doing business electronically, company web sites, online business software and services.)

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Apply first-hand experience of the selection and deployment of tools, techniques and theories for the identification, validation and structuring of a new business venture.
Indicative content:

The following are indicative of the topics the module will typically cover:

Invention and innovation, finding and qualifying new opportunities. Business Model Generation.
Confirming customer needs and testing market demand. Customer development.
Lean Start-ups: what is your minimum viable product? The value of prototyping.
Delivery channels and customer relationships. Business Plan and Preparing a Pitch.
Financial Forecasting, Costing and Pricing. Management accounts. Cash-flow and profit and loss.
Developing sustainable competitive advantage. Intellectual Property Rights.
Corporate form and structure. Founder dilemmas - team, equity, remuneration etc.
Developing your brand.
Defining and testing critical business model uncertainties. Measuring progress.
Common start-up metrics.
Sources of Funding. Presenting to VCs.
Class presentations, conclusions and next steps.
Requisites:

To be eligible to select this module as optional or elective, a student must be registered on a programme and year of study for which it is a formally available.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 2   Undergraduate (FHEQ Level 7)
Teaching and assessment
Mode of study

In person
Methods of assessment
75% Coursework
25% Viva or oral presentation
Mark scheme
Numeric Marks

MACHINE LEARNING FOR DOMAIN SPECIALISTS
Description
Aims:

The module aims to Introduce students to the basics of machine learning while giving class-based examples of applications to areas of domain specialisation.

Intended learning outcomes:

On successful completion of the module a student will be able to:

Understand elements of the fundamental concepts and mathematical basis of machine learning; apply practical machine learning software to perform data analysis tasks.
Indicative content:

General theory and mathematical foundations are presented in lectures while practical applications are presented in classes.

The following are indicative of the topics the module will typically cover:

An introduction to machine learning tasks (unsupervised, supervised, reinforcement).
Mathematical foundations (linear algebra, calculus, probability, statistics).
Supervised Learning: including an exploration of some of the following: linear and polynomial regression, logistic regression, Naive Bayes, kernel methods, SVMs, decision trees, ensemble learning, neural networks, Gaussian processes.
Unsupervised Learning: including an exploration of some of the following: PCA, manifold learning, k-means, Gaussian mixture models, EM algorithm.
Requisites:

To be eligible to select this module as an optional or elective, a student must: (1) be registered on a programme and year of study for which it is a formally available; and (2) should have experience of rudimentary programming and an awareness of standard results in fundamentals of linear algebra (vectors, matrices, eigenvectors /eigenvalues etc.), elements of probability theory (random variables, expectation, variance, conditional probabilities, Bayes rule etc.), elements of statistics (sample statistics, maximum likelihood estimation etc.), and calculus (real-valued functions, derivatives, Taylor series, integrals etc.). Results from these areas will be used, often without proof, throughout the module.

Self-assessment test:

Students should take a self assessment test, to assess their ability against the level of the module.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 2 Postgraduate (FHEQ Level 7)
Teaching and assessment
Mode of study
In person

Methods of assessment
60% Exam
40% Group activity
Mark scheme
Numeric Marks

QUANTUM COMPUTATION
Description
Aims:

The aim of the module is to introduce the students to the principles of quantum mechanics from a computer science perspective, and to the field of quantum computation and algorithms.

Intended learning outcomes:

On successful completion of the module, a student will be able to:

Understand (i.e., be able to describe, analyse and reason about) quantum computation, and how it differs from classical probabilistic computation.
Analyse the behaviour of simple quantum circuits.
Understand how quantum superposition and interference can give quantum speedups over classical algorithms in certain cases and apply these concepts to the design of quantum algorithms.
Derive the Quantum Fourier Transform, and apply it to the design of quantum algorithms
Derive Grover’s unstructured search technique and apply it to the design of quantum algorithms.
Indicative content:

The module provides a first course on quantum computation. The following is indicative of the topics the module will typically cover:

The basic mathematical formalism of quantum mechanics is be introduced from a computer science perspective, assuming no prior knowledge of quantum mechanics or physics. Building on this, the module develops the standard circuit model of quantum computation and basic quantum complexity theory. Most of the module is devoted to quantum algorithms, including rigorous derivations and algorithm analysis. This section of the course starts from simple examples, and builds up to the major quantum algorithms, such as Shor’s quantum factoring algorithm and Grover’s unstructured search algorithm.

Requisites:

To be eligible to select this module as an optional or elective, a student must be registered on a programme and year of study for which it is a formally available.

Module deliveries for 2024/25 academic year
Intended teaching term: Term 1 Undergraduate (FHEQ Level 6)
Teaching and assessment
Mode of study
In person
Methods of assessment
100% Exam
Mark scheme
Numeric Marks