Next Generation Higher National Unit Specification

Computer Science (SCQF level 8)

Unit code: J68M 48

SCQF level: 8 (24 SCQF credit points)

Valid from: session 2022–23

Prototype unit specification for use in pilot delivery only (version 1.0) May 2022

This unit specification provides detailed information about the unit to ensure consistent and transparent assessment year on year.

This unit specification is for teachers and lecturers and contains all the mandatory information required to deliver and assess the unit.

This edition: May 2022 (version 1)

Unit purpose

This specialist unit is designed for learners with a vocational interest in computer science. Its main purpose is to deepen learners' theoretical understanding of computer systems to facilitate progression to higher level studies.

Before starting this unit, learners should possess knowledge of computing and mathematics. They should have previous experience of computer programming, perhaps from Computer Science at SCQF level 7, and have some knowledge of computer science concepts.

This broad-based unit is primarily theoretical, covering a wide range of advanced topics, including architecture, operating system design, cryptography, graphics, artificial intelligence (AI) and ethics. Learners also gain skills in software engineering and programming in a low-level language, as well as discrete mathematics, which is introduced in context.

Unit outcomes

Learners who complete this unit can:

1 explain the architecture of computer systems
2 write software in a low-level programming language
3 demonstrate an understanding of discrete mathematics relating to computer science
4 explain the principles of machine learning
5 create and manipulate data within a normalised database structure
6 evaluate a current ethical dilemma in computer science

Evidence requirements

Learners must provide knowledge and product evidence for this unit, which must collectively demonstrate that they can meet all outcomes.

Knowledge evidence should relate to outcomes 1, 3, 4 and 6. Evidence should be the minimum required to infer competence, and should include at least one evaluation of a contemporary ethical issue. Sampling is allowed when testing is used. For example, knowledge evidence for outcomes 1, 3 and 4 could be produced by a test comprising a number of extended-response questions. When testing is used, knowledge evidence must be produced under controlled (and timed) conditions without access to reference materials.

Product evidence should relate to outcomes 2 and 5. Learners should produce one or more assembly language programs to demonstrate their practical competency in low-level programming. At least one relational, normalised database must be produced.

Evidence can be produced over an extended period of time in lightly controlled conditions, in which case authentication is required, or holistically generated in conjunction with other units within a group award.

The standard of evidence should be consistent with the SCQF level of this unit.

Knowledge and skills

The following table shows the knowledge and skills covered by the unit outcomes:

Knowledge	Skills
Learners should understand: ♦ the functionality, organisation, and implementation of computer systems ♦ microarchitecture design and logic design ♦ systems design, including direct memory access, virtualisation and multiprocessing ♦ CPU instruction sets ♦ the assembly process ♦ low-level programming ♦ discrete mathematics, including logic, set theory and number theory ♦ machine learning, deep learning and neural networks ♦ machine learning methods, including supervised and unsupervised learning ♦ machine learning algorithms ♦ databases, including relational databases and database schemas ♦ normalisation ♦ the ethical implications of computer science	Learners can: ♦ use system software to produce machine code ♦ write programs in a low-level language ♦ use a relational database management system ♦ create relational databases

Meta-skills

Throughout the unit, learners develop meta-skills to enhance their employability in the computing sector.

Self-management

This meta-skill includes:

- focusing: attention, filtering
- adapting: self-learning
- initiative: independent thinking, decision making, responsibility

Social intelligence

This meta-skill includes:

- communicating: receiving information, listening

Innovation

This meta-skill includes:

- creativity: visualising, idea generation
- sense-making: holistic thinking, pattern recognition
- critical thinking: logical thinking, judgement

Literacies

Numeracy

Learners develop numeracy skills, particularly as they discover new mathematical concepts through discrete mathematics and real-world implementations in the programming sections of the unit.

Communication

Learners develop communication skills by participating in oral and textual discussions, collaborating with their peers and presenting their learning.

Digital

Digital literacy is implicit in all areas of this unit.

Delivery of unit

This specialised unit introduces and expands on the principles of computer science. You can deliver it on a standalone basis, or there are opportunities for integration with other units when covering some of the subject matters in more depth.

We suggest the following allocation of time:

Outcome 1 — Explain the architecture of computer systems
(20 hours)

Outcome 2 — Write software in a low-level programming language
(30 hours)

Outcome 3 — Demonstrate an understanding of discrete mathematics relating to computer science
(20 hours)

Outcome 4 — Explain the principles of machine learning
(10 hours)

Outcome 5 — Create and manipulate data withing a normalised database structure
(30 hours)

Outcome 6 — Evaluate a current ethical dilemma in computer
(10 hours)

Additional guidance

The guidance in this section is not mandatory.

Databases

You should introduce basic database concepts, including relational databases, and spend some time teaching normalisation and/or optimisation and methods of questioning datasets.

You could capture evidence for this topic through small group projects, though there is the possibility of incorporating it into a broader project.

Computer architecture

You should choose topics that expand and build on learners' existing knowledge, and ensure that you also cover operating system architecture and hardware, and incorporate UX and UI design principles. There is scope for you to cross-pollinate knowledge of low-level programming with practical experience of the stored-program theory.

Discrete mathematics

As discrete mathematics is considered to be the foundation language of computer science, you should take time to establish a basic understanding of number theory, graph theory, set theory, and mathematical logic. You can then expand into exploring cryptography and additional avenues of computer science.

Low-level programming

You could approach low-level programming from a classical standpoint of assembly and machine language, covering the advantages and disadvantages of low-level programming, especially in embedded architectures and the Internet of Things (IoT). You could also introduce and discuss software engineering methods.

Approaches to assessment

Learners can produce evidence using a variety of approaches. For example, an extended-response test for knowledge evidence covering outcomes 1, 3 and 4, and an investigation into a contemporary ethical issue in computer science (outcome 6). The test could be 10 questions relating to computer architecture, discrete mathematics and machine learning, marked out of 100 marks, with a pass mark of 50 marks. The test should be carried out in timed and controlled conditions without access to reference materials. The investigation would relate to one of a number of contemporary ethical issues relating to computer science. For example, learners could investigate the threat to humanity that could be posed by the emergence of artificial general intelligence.

Product evidence could be produced through two assignments that involve learners producing assembly language code and creating a relational database.

Equality and inclusion

This unit is designed to be as fair and as accessible as possible with no unnecessary barriers to learning or assessment.

You should take into account the needs of individual learners when planning learning experiences, selecting assessment methods or considering alternative evidence.

Guidance on assessment arrangements for disabled learners and/or those with additional support needs is available on the assessment arrangements web page: www.sqa.org.uk/assessmentarrangements.

Information for learners

Computer Science (SCQF level 8)

This section explains:

- what the unit is about
- what you should know or be able to do before you start
- what you need to do during the unit
- opportunities for further learning and employment

Unit information

This is a specialist unit designed for those interested in the key concepts of computer science. It is particularly suitable if you wish to progress to further studies in computer science or a related field.

You should have knowledge of computing and mathematics before commencing this unit. You also need some experience of computer programming, perhaps acquired while studying Computer Science at SCQF level 7 or a specialist unit in programming.

This unit covers computer architecture — both hardware and software — and you explore UX and UI design concepts within this arena. You learn about database systems, including relational databases, and gain a practical understanding of the use of database systems in computer science. You put programming knowledge into practice in a low-level language, and examine the advantages and disadvantages of this style of programming. This unit introduces you to software engineering techniques. You also learn about the direct relationship between mathematics and computer science, and the practical implementations of mathematical techniques.

You are assessed both on your knowledge and your practical skills. Assessment for this unit is a mixture of formal assessments to test knowledge and at least one research-based report or essay. We also expect you to produce practical-based evidence, which can be in the form of a short portfolio or group-based tasks.

You develop your meta-skills naturally as you progress through this unit. These skills are not subject specific, and can be applied in many different contexts. During the unit you develop skills in self-management, such as focusing, adapting and initiative; in social intelligence, such as communicating; and in innovation, such as creativity, sense-making and critical thinking.

Administrative information

Published: May 2022 (version 1.0)

Superclass: CB

History of changes

Version	Description of change	Date

Note: please check SQA's website to ensure you are using the most up-to-date version of this document.