Next Generation Higher National Unit Specification

Computer Science (SCQF level 7)

Unit code: J68L 47

SCQF level: 7 (16 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 non-specialist unit is designed for a wide range of learners with an interest in computer science. It is particularly suitable for learners with a vocational interest in STEM, or who wish to progress to study an aligned STEM subject at university.

Learners should possess appropriate numeracy skills before starting this unit. They should also have experience of computer programming, and skills gained from units such as Digital Skills at SCQF level 7 or any other unit with some practical programming.

This unit introduces learners to the foundation principles of computer science, and provides a grounding in computer architecture and software development. Learners also acquire practical skills in computer programming in a high-level language.

The unit is mostly theoretical, and covers the main principles of computer science, including its historical development. Nonetheless, learners also gain experience of structured programming methods and are made aware of the practical applications of what they are learning.

On completion of this unit, learners may progress to the Computer Science unit at SCQF level 8.

Unit outcomes

Learners who complete this unit can:

1 perform calculations and conversions between number and logic systems
2 explain the principles of computer architecture
3 install systems software onto a computer
4 write computer programs in a high-level language using structured programming
5 investigate present-day developments in computer science

Evidence requirements

Learners must provide knowledge and product evidence for this unit.

This evidence must collectively demonstrate that they can:

1 describe measures of performance and storage
2 perform conversions and calculations using number systems
3 describe computer architecture
4 explain basic concepts in computer science
5 describe present-day developments in computer science
6 install systems software
7 write computer programs

The evidence should be the minimum required to infer competence, and should include at least one of each of the following:

- the installation of systems software on at least one computing device
- a description of one contemporary development, including its historical context and ethical implications
- the writing of one computer program — this must include the associated diagrams and algorithms to demonstrate computational thinking, one sort algorithm and at least one search algorithm

Sampling is allowed when testing is used. For example, evidence requirements 1, 2, 3 and 4 could be evidenced by means of a test comprising short-response and extended-response questions.

Evidence may 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: ♦ measures of performance and storage ♦ mathematical foundations including number systems and Boolean logic ♦ historical development of computer systems ♦ present-day developments in computer science including artificial intelligence (AI) and machine learning (ML) ♦ ethics and computer science ♦ computer organisation ♦ systems software including language translators ♦ operating system design including user interface (UI) and user experience (UX) ♦ basic concepts in computer science, including the stored-program concept ♦ computational thinking in terms of decomposition, abstraction, algorithms and pattern recognition ♦ algorithms and data structures ♦ programming paradigms ♦ programming languages and levels of language ♦ syntax and semantics of a high-level language ♦ programming methods, including structured programming and pseudocode ♦ algorithms for sorting and searching ♦ software development processes, including testing	Learners can: ♦ convert between number systems ♦ perform calculations using number systems ♦ install and customise systems software, including operating systems ♦ interpret diagrams ♦ apply computational thinking ♦ create flowcharts and other visualisations ♦ write algorithms ♦ select data structures ♦ program in a high-level language ♦ apply structured programming techniques to code

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: critical reflection
- initiative: independent thinking, responsibility

Social intelligence

This meta-skill includes:

- communicating: receiving information, listening
- collaborating: teamworking and collaboration

Innovation

This meta-skill includes:

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

Literacies

Numeracy

Numeracy is implicit in multiple areas of this unit and is used exclusively in some.

Communication

This unit helps develop communication skills — particularly during the collaborative and presentation elements.

Digital

Digital literacy is implicit in all areas of this unit.

Delivery of unit

This unit provides foundational knowledge and skills in computer science.

We suggest the following allocation of time:

Outcome 1 — Perform calculations and conversions between number and logic systems
(10 hours)

Outcome 2 — Explain the principles of computer architecture
(15 hours)

Outcome 3 — Install systems software onto a computer
(15 hours)

Outcome 4 — Write computer programs in a high-level language using structured programming
(30 hours)

Outcome 5 — Investigate present-day developments in computer science
(10 hours)

Learners require access to a range of digital technologies, such as personal computers and tablets, and a range of software types, including a high-level programming language.

Additional guidance

The guidance in this section is not mandatory.

Content and context for this unit

This unit serves as an introduction to computer science. We do not assume that learners have formal, prior knowledge of computer science. Nonetheless, they do need some prior experience of coding, such as that included in the Digital Skills unit at SCQF level 7.

It is important that knowledge crossover is highlighted throughout the delivery of this unit, and that you provide real-world examples or tasks.

Mathematical foundations

This foundational knowledge is demonstrable throughout the unit, and you should ensure that learners see real-world applications of the knowledge they are gaining.

Number systems overview

You should ensure that mathematical and number systems topics focus on number bases with a historical and current relationship to computing and computer science — particularly base 2 (binary) and base 16 (hexadecimal). As well as introducing the theory, you should demonstrate these number systems via practical implementations throughout the unit; for example, the use of binaries in computer architecture, and the use of hexadecimal throughout the fetch-execute process.

You should teach the conversion between number systems using multiple methods — such as positional notation and two’s complement — to minimise learners’ reliance on conversion tables.

Boolean logic

As with number systems, you should ensure that you teach Boolean logic beyond theoretical knowledge. Introduce practical examples within programming and link the Boolean logic to selection and iteration principles to demonstrate practical, real-world applications.

Programming in high-level language

You should teach learners that, although the unit may focus on a single programming paradigm (such as procedural), there are several others that they should investigate in their own time.

Learners should become familiar with the syntax and semantics of one language, which they can use to develop a single program to meet the criteria for the outcome. The focus should be programming techniques, not software complexity. The quality of the code is the vital aspect, and this must be well designed and well structured.

Principles of computer architecture

This should feature not just the physical architecture, but the full principles, functionality, organisation and implementation of computer systems. You should introduce each sub-section of the architecture with diagrammatic examples, and aim to include examples, such as the von Neumann diagram or other suitable alternatives.

You could explain each system and then integrate and show interaction with the stored-program concept; for example, the fetch-execute cycle.

You should introduce central processing unit (CPU) and architecture performance criteria, and cover concepts such as clock speed, cores and caches, and how they can impact overall system performance.

The evidence requirements could be satisfied in a variety of ways. A traditional approach to assessment might involve the following:

- test of knowledge for outcome 1 and outcome 2 (single test)
- practical assignment for outcome 3
- programming assignment for outcome 4
- research report for outcome 5

Approaches to assessment

A more contemporary approach to assessment might require learners to maintain a web log during the unit. They would record their knowledge and skills as they acquired them during the life of the unit.

If you take this approach, learners also have to provide product evidence (code) for their programming skills (outcome 4).

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 7)

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

During this unit you not only gain theoretical knowledge but also learn how to apply it, and in many cases have the opportunity to do so yourself.

This unit introduces you to the basic principles of computer science, and develops your skills in computer programming. You are not expected to know anything about computer science before commencing the unit, but we do assume you have some experience of computer programming. When you complete the unit, you understand the basic organising principles of computer systems and have an appreciation of present-day programming concepts.

The unit covers a wide range of knowledge and skills including:

- measures of performance and storage
- number systems, such as binary and hexadecimal
- how computers work
- how to install systems software
- how to write well-structured computer programs
- contemporary developments in the field of computer science, such as artificial intelligence (AI)

Mathematical foundations

You have the opportunity to use your knowledge of mathematical foundations in real-world applications throughout the unit.

Number systems overview

You focus on number bases with a historical and current relationship to computing and computer science, particularly base 2 (binary) and base 16 (hexadecimal). You also see practical examples of their application; for example, using binaries in computer architecture, and hexadecimals throughout the fetch-execute process.

You are encouraged to become familiar with multiple methods of conversion between number systems — including positional notation and two’s complement, among others — so that you become less reliant on conversion tables.

Boolean logic

As with number systems, you learn not only the theory of Boolean logic, but also how it is used in different aspects of computer science. In this unit, the practical application of Boolean logic is demonstrated during the programming sections, and as part of the selection and interaction principles.

Programming in high-level language

We expect you to become familiar with the syntax and semantics of one programming language, and to develop a single program to meet the criteria for the outcome.

The unit may focus on a single programming paradigm, such as procedural, but we encourage you to investigate the other paradigms in your own time.

Principles of computer architecture

As well as the physical architecture, this unit takes you through the full principles, functionality, organisation and implementation of computer systems. These are explained using a range of diagrammatic examples, including the von Neumann diagram.

You learn about each system individually and then study how it integrates and interacts with the stored-program concept; for example, in the fetch-execute cycle.

As part of this outcome, you use CPU and architecture performance criteria, and cover concepts such as clock speed, cores, caches, and how they can impact overall system performance.

Your knowledge and skills may be assessed in a variety of ways. For example, you might have a test to assess your knowledge of theory and a practical assignment to assess your programming skills.

You develop your meta-skills throughout this unit; in particular, pattern recognition, problem solving and computational thinking skills.

On completion of this unit, you could progress to:

- Computer Science at SCQF level 8

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.