Higher National Unit Specification

General information

Unit title: Computer Programming (SCQF level 7)

Unit code: J0HA 34

Superclass: CA

Publication date: June 2018

Source: Scottish Qualifications Authority

Version: 02

Unit purpose

The purpose of this unit is to **introduce** learners to the **basic** principles and practice of computer programming using contemporary high-level programming languages. No previous programming experience is required.

It is a **non-specialist** unit, intended for a wide range of learners with an interest in coding. It is particularly suitable for learners with an interest in Science, Technology, Engineering or Mathematics (STEM) who want to understand the principles of programming; however, it will be beneficial to any learner who wants to gain an understanding of programming for vocational or personal purposes.

The unit focuses specifically on coding, rather than the wider aspects of software development. Learners will gain foundation knowledge of, and skills in, writing short programs for a variety of purposes, using contemporary high-level programming languages, such as Python or C. They will also be introduced to programming concepts and programming techniques, and develop their logical and computational thinking skills.

Learners can progress to a number of more specialised qualifications, including more advanced units in computer programming such as H173 34 *Developing Software: Introduction* (SCQF level 7) or H171 35 *Software Development: Object Oriented Programming* (SCQF level 8).

Higher National Unit Specification: General information (cont)

Unit title: Computer Programming (SCQF level 7)

Outcomes

On successful completion of the unit, the learner will be able to:

1 Describe the programming process.
2 Explain programming concepts.
3 Write algorithms to solve familiar problems.
4 Write programs to solve familiar problems.

Credit points and level

1 Higher National unit credit at SCQF level 7: (8 SCQF credit points at SCQF level 7)

Recommended entry to the unit

This is an introductory unit in computer programming and, as such, no previous knowledge of programming is required.

Core Skills

Achievement of this Unit gives automatic certification of the following Core Skills component:

Complete Core Skill None

Core Skill component Critical Thinking at SCQF level 5

There are also opportunities to develop aspects of Core Skills which are highlighted in the Support Notes of this Unit specification.

Context for delivery

If this unit is delivered as part of a group award, it is recommended that it should be taught and assessed within the subject area of the group award to which it contributes.

There may be opportunities to combine the assessment of this unit with other units. For example, the knowledge contained within this unit could be combined with the knowledge contained within other units and assessed through an examination-based integrated assessment. The practical competencies could be combined with the practical competences contained within other units (such as J0J9 34 *Machine Learning* and/or J0HK 34 *Ethical Hacking*) and assessed through a project-based integrated assessment.

The Assessment Support Pack (ASP) for this unit provides assessment and marking guidelines that exemplify the national standard for achievement. It is a valid, reliable and practicable assessment. Centres wishing to develop their own assessments should refer to the ASP to ensure a comparable standard. A list of existing ASPs is available to download from SQA's website **(http://www.sqa.org.uk/sqa/46233.2769.html)**.

Equality and inclusion

This unit specification has been designed to ensure that there are no unnecessary barriers to learning or assessment. The individual needs of learners should be taken into account when planning learning experiences, selecting assessment methods or considering alternative evidence.

Further advice can be found on our website **www.sqa.org.uk/assessmentarrangements**.

Higher National unit specification: Statement of standards

Unit title: Computer Programming (SCQF level 7)

Acceptable performance in this unit will be the satisfactory achievement of the standards set out in this part of the unit specification. All sections of the statement of standards are mandatory and cannot be altered without reference to SQA.

Where evidence for outcomes is assessed on a sample basis, the whole of the content listed in the knowledge and/or skills section must be taught and available for assessment. Learners should not know in advance the items on which they will be assessed and different items should be sampled on each assessment occasion.

Outcome 1

Describe the programming process.

Knowledge and/or skills

- High level programming languages
- Types of programming language
- Stages in programming
- Requirements specification
- Editing, translation, testing and debugging
- Program documentation including internal documentation
- Programming tools including editors and translators

Outcome 2

Explain programming concepts.

Knowledge and/or skills

- Syntax and semantics
- Algorithms and data structures
- Variables and data types
- Operators and operator precedence
- Assignment statements
- Program constructs
- Syntax errors and logic errors
- Structured programming
- Program testing

Outcome 3

Write algorithms to solve familiar problems.

Knowledge and/or skills

- Methods of expressing algorithms
- Representation of programming constructs as algorithms
- Functional decomposition of problems (top-down problem solving)
- Sorting and searching algorithms
- Problem solving using algorithms

Higher National unit specification: Statement of standards (cont)

Unit title: Computer Programming (SCQF level 7)

Outcome 4

Write programs to solve familiar problems.

Knowledge and/or skills

- Syntax and semantics of a high level language
- Implementation of variables, operators and expressions in a high level language
- Implementation of programming constructs in a high level language
- Implementation of data structures in a high level language
- Implementation of structured programming in a high level language
- Implementation of algorithms in a high level language
- Programming tools including editors, translators and debuggers

Evidence requirements for this unit

Learners will need to provide evidence to demonstrate the knowledge and/or skills across all outcomes. The evidence requirements for this unit will take **two** forms.

1 Knowledge evidence (Outcome 1 and Outcome 2)
2 Product evidence (Outcome 3 and Outcome 4)

The **knowledge evidence** will relate to Outcome 1 and Outcome 2. Knowledge evidence is required for **all** knowledge and/or skills statements in these outcomes. The evidence may be produced over an extended period of time in lightly controlled conditions. The amount of evidence may be the **minimum** required to infer competence. For example, it is acceptable if learners describe the main languages (only) used for programming (Outcome 1) and explain the most important programming constructs (Outcome 2).

The knowledge evidence may be sampled when testing is used. In this case, the evidence must be produced under controlled conditions in terms of location (supervised), timing (limited) and access to reference materials (not permitted). The sampling frame must include both outcomes (at least partially) and the majority of knowledge/skills statements (in each outcome). The sampling frame must always include the following:

- Stages in programming
- Syntax and semantics
- Algorithms and data structures
- Program constructs

The knowledge evidence may be written or oral or a combination of these. Evidence may be captured, stored and presented in a range of media (including audio and video) and formats (analogue and digital). Particular consideration should be given to digital formats and the use of multimedia.

Higher National unit specification: Statement of standards (cont)

Unit title: Computer Programming (SCQF Level 7)

The **product evidence** will relate to Outcome 3 and Outcome 4. The evidence must demonstrate that the learner can write **at least one** algorithm and **at least one** program to solve a problem. The algorithm and program may relate to the same problem. The problem may be **familiar** and **routine** but must be **non-trivial**. Its solution should involve a **range** of programming constructs and **two or more** data structures **including one dimensional arrays**. The solution should require **modular programming** to demonstrate a structured approach to coding. The program must **compile without syntax errors**. There must be **evidence of testing**, which must indicate that the program **executes correctly** and is **error free** (logically correct).

Product evidence may be produced over the life of the unit, under loosely controlled conditions (including access to reference materials). Authentication will be necessary (see below).

The SCQF level of this unit (level 7) provides additional context on the nature of the required evidence and the associated standards. Appropriate level descriptors should be used when making judgements about the evidence.

When evidence is produced in loosely controlled conditions it must be authenticated. The Guide to Assessment provides further advice on methods of authentication.

The support notes section of this specification provides specific examples of instruments of assessment that will generate the required evidence.

Higher National Unit Support Notes

Unit title: Computer Programming (SCQF level 7)

Unit support notes are offered as guidance and are not mandatory.

While the exact time allocated to this unit is at the discretion of the centre, the notional design length is **40 hours**.

Guidance on the content and context for this unit

This unit aims to introduce learners to computer programming. No previous experience is required, or should be assumed, from learners. The focus of the unit is coding, rather than the broader subject of software development. The unit is suitable for a wide range of learners with a wide range of interests, including learners who wish to learn to program for personal purposes.

All of the outcomes are equally important. Although Outcomes 1–3 lead to Outcome 4 (the coding outcome), the key aspect of the unit is that learners gain an appreciation of the programming process. Subsequent units can deepen this basic knowledge for those learners who require more specialised knowledge and skills.

It is recommended that a contemporary high-level procedural programming language is used, such as Python. It is recommended that the language chosen is accessible to new learners and is typical of high-level programming languages in general.

Learners will require access to a range of resources to undertake this unit, including desktop computers, program translator and programming tools, such as editors and debuggers.

Outcome 1 and Outcome 2 are theoretical. Outcome 3 and Outcome 4 are practical. The first two outcomes provide a theoretical basis to programming; the latter two outcomes apply that knowledge, using a specific programming language (Outcome 4). Given the short time available to learn the basics of coding, the types of problems presented to learners should be familiar and non-complex. The problems should be linked to learners' vocational or personal interests.

Please note that the following guidance, relating to specific outcomes, does not seek to explain each knowledge/skills statement, which is left to the professionalism of the teacher. It seeks to clarify the statement of standards where it is potentially ambiguous. It also focuses on non-apparent teaching and learning issues that may be over-looked, or not emphasised, during unit delivery. As such, it is not representative of the relative importance of each knowledge/skill.

Outcome 1: This outcome is about the programming process. It provides an opportunity for learners to learn how programs are created in the abstract before applying that knowledge in later outcomes.

The level of treatment of each topic should be light. For example, it is sufficient to introduce learners to the most common programming languages and the most common types of language (procedural, event-driven, etc).

Higher National Unit Support Notes (cont)

Unit title: Computer Programming (SCQF level 7)

The critical aspect of this outcome is to prepare learners for the subsequent outcomes. This outcome provides learners with an opportunity to learn about the programming process, the sorts of languages that can be used, and the tools that are required to create code.

Although the outcome is theoretical, there is an opportunity to reinforce learning by permitting learners to use programming tools such as editors, compilers and debuggers by practising with pre-prepared programs.

Outcome 2: This outcome is about programming concepts. It is a theoretical outcome, so the context for learning should be abstract and not based on one specific programming language. Learning will be more effective if a range of languages is used to illustrate the concepts.

The critical aspect of this outcome is breadth, not depth. The level of treatment for any topic should be light. For example, it is sufficient for learners to understand the basic distinction between syntax and semantics or the most common operators and programming constructs. The treatment of structured programming and program testing will have to be particularly light, given the inherent complexity of these topics.

Outcome 3: This outcome is about algorithms. Learners will be taught how to express the solution to problems using algorithms.

More than one method of writing algorithms should be introduced. It is recommended that flowcharts is one of them. However, at least one other method should be illustrated (such as pseudo code).

The treatment of sorting and searching should be light, with only the most common algorithms introduced. In the case of sorting, this would be: merge sort, insertion sort, bubble sort and quicksort. In the case of searching, this would be: linear search and binary search. There is an opportunity to discuss algorithmic efficiency when comparing different algorithms that do the same thing.

Outcome 4: This outcome is about coding. Learners will be introduced to the syntax and semantics of a specific high level programming language. This outcome permits learners to apply the knowledge gained in previous outcomes. For example, they will implement algorithmic constructs in a high-level language, and discover how to write, sort and search programs in that language.

The programming problems that learners are required to solve should be familiar and non-complex. Suitable programs include:

- A program to compute the change given to a customer using the least number of coins/notes
- A program to control two (or more) sets of traffic lights
- A guessing game using randomly generated numbers (‘high/low’)
- Credit scoring program
- Battleships game

Higher National Unit Support Notes (cont)

Unit title: Computer Programming (SCQF level 7)

Guidance on approaches to delivery of this unit

It is anticipated that learners will complete the outcomes sequentially. A suggested distribution of time, across the outcomes, is:

Outcome 1: 8 hours
Outcome 2: 8 hours
Outcome 3: 10 hours
Outcome 4: 14 hours

Outcome 1 is a stand-alone outcome that provides important background information for subsequent outcomes. There is an opportunity, when discussing programming tools, to permit learners to gain practical skills in preparation for Outcome 4.

Outcomes 2, 3 and 4 could be delivered discretely or combined into one learning activity. Combined delivery would permit learning to be contextualised. For example, the explanations of syntax and semantics (Outcome 2) could be linked to the syntax and semantics of a specific programming language (Outcome 4); the explanation of algorithms (Outcome 2) could be linked to specific algorithms for sorting data (Outcome 3); problem solving using algorithms (Outcome 3) could be combined with the implementation of algorithms in a high level language (Outcome 4).

There are established pedagogical approaches to learning programming that should be followed. Consolidation of learning is one of them. The pace of learning, particularly in the early stages of learning, should be slow to permit learners to internalise (and practise) fundamental programming concepts. The use of programming examples (both complete programs and parts of programs) should be used throughout the period of learning to illustrate concepts. Group learning, in teams of two or three learners, is recommended as is group discussions of programming solutions.

Guidance on approaches to assessment of this unit

Evidence can be generated using different types of assessment. The following are suggestions only. There may be other methods that would be more suitable to learners.

Centres are reminded that prior verification of centre-devised assessments would help to ensure that the national standard is being met. Where learners experience a range of assessment methods, this helps them to develop different skills that should be transferable to work or further and higher education.

A traditional approach to assessment would involve a test for Outcome 1 and Outcome 2 (to generate knowledge evidence), and a practical assignment for Outcome 3 and Outcome 4 (to generate practical evidence). The test could be a multiple-choice test and the practical assignment could be a programming task.

The test could consist of a number of Selected Response Questions (SRQs) that assess the knowledge and understanding contained in Outcome 1 and Outcome 2. The test would be timed, closed-book and supervised. An appropriate pass mark would be set. Learners who achieve this threshold would achieve both outcomes. The majority of questions would relate to factual recall (of such things as the stages in programming and knowledge of operator precedence); some questions would relate to deeper understanding and would require more complex types of questions.

Alternatively, the evidence could be generated by using a smaller number of short-answer questions, which could be written or oral. These questions could cover multiple concepts in each question and, therefore, would require fewer questions.

The practical assignment could take the form of a programming task requiring learners to solve a familiar problem. The task could require learners to create a flowchart to represent the solution to the problem and then write the corresponding code. The resulting evidence would be the flowchart(s) and source code.

A more contemporary approach to assessment would involve the use of a web log (blog) to record learning throughout the life of the unit. The blog would provide knowledge evidence (in the descriptions and explanations) and some (or all) product evidence. It is likely (but not essential) that the blog would have to be supplemented with additional evidence (such as source code). The blog should be assessed using defined criteria to permit a correct judgement about the quality of the evidence. In this scenario, every knowledge and skill must be evidenced; sampling would not be appropriate.

There are opportunities to carry out formative assessment at various stages in the unit. For example, formative assessment could be carried out on the completion of each outcome to ensure that learners have grasped the knowledge contained within it. This would provide assessors with an opportunity to diagnose misconceptions, and intervene to remedy them before progressing to the next outcome. If a blog is used for summative assessment, it would also facilitate formative assessment since learning (including misconceptions) would be apparent from the blog, and intervention could take place to correct misunderstandings on an on-going basis.

Authentication may take various forms including, but not limited to, oral questioning and plagiarism checks. Some forms of evidence generation (such as video recordings) have intrinsic authentication and would require no further means of verification. Where evidence is not generated under closely controlled conditions (for example, out of class), then a statement of authenticity could be provided by the learner to verify the work as their own, and state any necessary sources and permissions.

Centres are reminded that prior verification of centre-devised assessments would help to ensure that the national standard is being met. Where learners experience a range of assessment methods, this helps them to develop different skills that should be transferable to work or further and higher education.

Opportunities for e-assessment

E-assessment may be appropriate for some assessments in this unit. By e-assessment we mean assessment which is supported by Information and Communication Technology (ICT), such as e-testing or the use of e-portfolios or social software. Centres which wish to use e-assessment must ensure that the national standard is applied to all learner evidence and that conditions of assessment as specified in the evidence requirements are met, regardless of the mode of gathering evidence. The most up-to-date guidance on the use of e-assessment to support SQA's qualifications is available at **www.sqa.org.uk/e-assessment**.

Higher National Unit Support Notes (cont)

Unit title: Computer Programming (SCQF level 7)

Opportunities for developing Core and other essential skills

This unit will provide opportunities for learners to develop Core Skills in *Problem Solving* and *Information and Communication Technology* at SCQF level 6.

The Providing/Creating Information component of *Information and Communication Technology* could be developed in Outcomes 3 and 4, where the learner has to create a computer program.

All three components of *Problem Solving* (Critical Thinking, Planning and Organising, Reviewing and Evaluating) could be developed in Outcome 2 where the learner has to create and refine algorithms to solve problems. Aspects of *Problem Solving* could be developed further in Outcome 4, where the learner has to implement an algorithm into code and then test and debug it. It is highly likely that once an algorithm is translated into code, problems will appear requiring refinements to the algorithm/approach.

This Unit has the Critical Thinking component of Problem Solving embedded in it. This means that when learners achieve the Unit, their Core Skills profile will also be updated to show they have achieved Critical Thinking at SCQF level 5.

History of changes to unit

Version	Description of change	Date
02	Core Skills Component Critical Thinking at SCQF level 5 embedded.	31/08/18

General information for learners

Unit title: Computer Programming (SCQF level 7)

This section will help you decide whether this is the unit for you by explaining what the unit is about, what you should know or be able to do before you start, what you will need to do during the unit and opportunities for further learning and employment.

The purpose of this unit is to introduce you to computer programming. No previous knowledge or experience of programming is presumed. This unit is suitable for anyone with an interest in coding.

You will learn to program in languages such as Python or JavaScript. The programs that you learn to write will be relatively simple. You will be permitted to write programs that relate to your personal or vocational interests.

You will learn about the programming process, essential programming concepts, how to write algorithms, and how to translate algorithms into code. As well as learning how to program, you will also learn how to use the various programming tools needed to create code such as editors, compilers and debuggers.

You will not only learn the basics of coding, but also how to write code well. Techniques, such as structured programming, and how to test your code will be taught.

You will be assessed in a variety of ways. Outcomes 1 and 2 are theoretical, whereas Outcomes 3 and 4 are practical. Therefore, it is likely that a more traditional approach to assessment will be taken for Outcomes 1 and, such as a test. For Outcomes 3 and 4, you will be required to carry out a programming task, which will allow you to put into practice the knowledge learned in earlier outcomes.

Throughout the unit, you will also have the opportunity to develop Core Skills in problem solving and *Information and Communication Technology (ICT)* at SCQF level 6.

On the completion of this unit, you will know the basics of coding using a contemporary high-level language, such as Python or JavaScript. You could progress to more advanced units in computer programming, such as J0J9 34 *Machine Learning*.

This Unit has the Critical Thinking component of Problem Solving embedded in it. This means that when you achieve the Unit, your Core Skills profile will also be updated to show you have achieved Critical Thinking at SCQF level 5.