National 5 Course Specification

National 5 Computing Science

Course code:	C816 75
Course assessment code:	X816 75
SCQF:	level 5 (24 SCQF credit points)
Valid from:	session 2017–18

The course specification provides detailed information about the course and course assessment to ensure consistent and transparent assessment year on year. It describes the structure of the course and the course assessment in terms of the skills, knowledge and understanding that are assessed.

This document is for teachers and lecturers and contains all the mandatory information you need to deliver the course.

This edition: May 2017, version 1.1

Contents

Course overview

The course consists of 24 SCQF credit points which includes time for preparation for course assessment. The notional length of time for a candidate to complete the course is 160 hours.

The course assessment has two components.

Component	Marks	Duration
Component 1: question paper	110	2 hours
Component 2: assignment	50	See course assessment section

Recommended entry	Progression
Entry to this course is at the discretion of the centre. Candidates should have achieved the fourth curriculum level or the National 4 Computing Science course or equivalent qualifications and/or experience prior to starting this course.	♦ other qualifications in computing science or related areas ♦ further study, employment and/or training

Conditions of award

The grade awarded is based on the total marks achieved across all course assessment components.

Achievement of this course gives automatic certification of the following Core Skill:

- Information and Communication Technology at SCQF level 5

Course rationale

National Courses reflect Curriculum for Excellence values, purposes and principles. They offer flexibility, provide more time for learning, more focus on skills and applying learning, and scope for personalisation and choice.

Every course provides opportunities for candidates to develop breadth, challenge and application. The focus and balance of assessment is tailored to each subject area.

The National 5 Computing Science course encourages candidates to become successful, responsible and creative in using technologies, and to develop a range of qualities including flexibility, perseverance, confidence, and enterprise.

At this level, the course covers a common core of concepts which underpin the study of computing science and explores the role and impact of contemporary computing technologies. It also includes a range of transferable skills, which opens up a wide range of career and study opportunities.

Purpose and aims

The course helps candidates to understand computational processes and thinking. It covers a number of unifying themes that are used to explore a variety of specialist areas, through practical and investigative tasks.

The course highlights how computing professionals are problem-solvers and designers, and the far-reaching impact of information technology on our environment and society.

It enables candidates to:

- apply computational-thinking skills across a range of contemporary contexts
- apply knowledge and understanding of key concepts and processes in computing science
- apply skills and knowledge in analysis, design, implementation, testing and evaluation to a range of digital solutions
- communicate computing concepts and explain computational behaviour clearly and concisely using appropriate terminology
- develop an understanding of the role and impact of computing science in changing and influencing our environment and society

Who is this course for?

This course is designed for learners who are considering further study or a career in computing science and related disciplines. It provides opportunities to enhance skills in planning and organising, working independently and in teams, critical thinking and decision making, research, communication, and self- and peer-evaluation, in a range of contexts.

Course content

The course has four areas of study:

Software design and development

Candidates develop knowledge, understanding and practical problem-solving skills in software design and development, through a range of practical and investigative tasks using appropriate software development environments. This develops their programming and computational-thinking skills by implementing practical solutions and explaining how these programs work. Tasks involve some complex features (in both familiar and new contexts), that require some interpretation by candidates. They are expected to analyse problems, and design, implement, test and evaluate their solutions.

Computer systems

Candidates develop an understanding of how data and instructions are stored in binary form and basic computer architecture. They gain an awareness of the environmental impact of the energy use of computing systems and security precautions that can be taken to protect computer systems.

Database design and development

Candidates develop knowledge, understanding and practical problem-solving skills in database design and development, through a range of practical and investigative tasks. This allows candidates to apply computational-thinking skills to analyse, design, implement, test, and evaluate practical solutions, using a range of development tools such as SQL. Tasks involve some complex features (in both familiar and new contexts), that require some interpretation by candidates.

Web design and development

Candidates develop knowledge, understanding and practical problem-solving skills in web design and development, through a range of practical and investigative tasks. This allows candidates to apply computational-thinking skills to analyse, design, implement, test and evaluate practical solutions to web-based problems, using a range of development tools such as HTML, CSS and Javascript. Tasks involve some complex features (in both familiar and new contexts), that require some interpretation by candidates.

Skills, knowledge and understanding

Skills, knowledge and understanding for the course

The following provides a broad overview of the subject skills, knowledge and understanding developed in the course:

- applying aspects of computational thinking across a range of contexts
- analysing problems within computing science across a range of contemporary contexts

- designing, implementing, testing and evaluating digital solutions (including computer programs) to problems across a range of contemporary contexts
- developing skills in computer programming and the ability to communicate how a program works, by being able to read and interpret code
- communicating understanding of key concepts related to computing science, clearly and concisely, using appropriate terminology
- understanding of legal implications and environmental impact of contemporary technologies
- applying computing science concepts and techniques to create solutions across a range of contexts

Skills, knowledge and understanding for the course assessment

The following provides details of skills, knowledge and understanding sampled in the course assessment:

Software design and development	
Development methodologies	Describe and implement the phases of an iterative development process: analysis, design, implementation, testing, documentation, and evaluation, within general programming problem-solving.
Analysis	Identify the purpose and functional requirements of a problem that relates to the design and implementation at this level, in terms of: ♦ inputs ♦ processes ♦ outputs
Design	Identify the data types and structures required for a problem that relates to the implementation at this level, as listed below. Describe, identify, and be able to read and understand: ♦ structure diagrams ♦ flowcharts ♦ pseudocode Exemplify and implement **one** of the above design techniques to design efficient solutions to a problem. Describe, exemplify, and implement user-interface design, in terms of input and output, using a wireframe.
Implementation (data types and structures)	Describe, exemplify, and implement appropriately the following data types and structures: ♦ character ♦ string ♦ numeric (integer and real)

	♦ Boolean ♦ 1-D arrays
Implementation (computational constructs)	Describe, exemplify, and implement the appropriate constructs in a high-level (textual) language: ♦ expressions to assign values ♦ expressions to return values using arithmetic operations (addition, subtraction, multiplication, division, and exponentiation) ♦ expressions to concatenate strings ♦ selection constructs using simple conditional statements with <, >, ≤, ≥, =, ≠ operators ♦ selection constructs using complex conditional statements ♦ logical operators (AND, OR, NOT) ♦ iteration and repetition using fixed and conditional loops ♦ pre-defined functions (with parameters): — random — round — length Read and explain code that makes use of the above constructs.
Implementation (algorithm specification)	Describe, exemplify, and implement standard algorithms: ♦ input validation ♦ running total within loop ♦ traversing a 1-D array
Testing	Describe, identify, exemplify, and implement normal, extreme, and exceptional test data for a specific problem, using a test table. Describe and identify syntax, execution, and logic errors.
Evaluation	Describe, identify, and exemplify the evaluation of a solution in terms of: ♦ fitness for purpose ♦ efficient use of coding constructs ♦ robustness ♦ readability: — internal commentary — meaningful identifiers — indentation — white space

Computer systems	
Data representation	Describe and exemplify the use of binary to represent positive integers. Describe floating point representation of positive real numbers using the terms mantissa and exponent. Convert from binary to denary and vice-versa. Describe extended ASCII code (8-bit) used to represent characters. Describe the vector graphics method of graphic representation for common objects: ♦ rectangle ♦ ellipse ♦ line ♦ polygon with attributes: ♦ co-ordinates ♦ fill colour ♦ line colour Describe the bit-mapped method of graphics representation.
Computer structure	Describe the purpose of the basic computer architecture components and how they are linked together: ♦ processor (registers, ALU, control unit) ♦ memory locations with unique addresses ♦ buses (data and address) Explain the need for interpreters and compilers to translate high-level program code to binary (machine code instructions).
Environmental impact	Describe the energy use of computer systems, the implications on the environment and how these could be reduced through: ♦ settings on monitors ♦ power down settings ♦ leaving computers on stand-by
Security precautions	Describe the role of firewalls. Describe the use made of encryption in electronic communications.

Database design and development	
Analysis	Identify the end-user and functional requirements of a database problem that relates to the implementation at this level.
Design	Describe and identify the implications for individuals and businesses of the Data Protection Act 1998: ♦ prior consent of data subject ♦ accuracy of data ♦ data used for limited, specifically stated purposes ♦ data kept safe and secure Describe and exemplify entity relationship diagrams with two entities indicating: ♦ entity name ♦ attributes ♦ relationship (one to many) Describe and exemplify a data dictionary: ♦ entity name ♦ attribute name ♦ primary and foreign key ♦ attribute type: — text — number — date — time — Boolean ♦ attribute size ♦ validation: — presence check — restricted choice — field length — range Exemplify a design of a solution to the query: ♦ multiple tables ♦ fields ♦ search criteria ♦ sort order

Implementation	Implement relational databases with two linked tables, to match the design with referential integrity. Describe, exemplify and implement SQL operations for pre-populated relational databases, with a maximum of two linked tables: ♦ select: — from — where: o AND, OR, <, >, = o order by with a maximum of two fields ♦ insert ♦ update ♦ delete ♦ equi-join between tables Read and explain code that makes use of the above SQL.
Testing	Describe and exemplify testing: ♦ SQL operations work correctly at this level
Evaluation	Evaluate solution in terms of: ♦ fitness for purpose ♦ accuracy of output

Web design and development	
Analysis	Identify the end-user and functional requirements of a website problem that relates to the design and implementation at this level.
Design	Describe and exemplify the website structure with a home page, a maximum of four linked multimedia pages, and any necessary external links. Describe, exemplify and implement, taking into account end-user requirements, effective user-interface design (visual layout and readability) using wire-framing: ♦ navigational links ♦ consistency across multiple pages ♦ relative vertical positioning of the media displayed ♦ file formats of the media (text, graphics, video, and audio) Describe and identify the implications for individuals and businesses of the Copyright, Designs and Patents Act 1988 relating to:

	♦ web content (text, graphics, video, and audio) Compare a range of standard file formats: ♦ audio standard file formats WAV and MP3 in terms of compression, quality, and file size ♦ bit-mapped graphic standard file formats JPEG, GIF, and PNG in terms of compression, animation, transparency, and colour depth Describe the factors affecting file size and quality, relating to resolution, colour depth, and sampling rate. Describe the need for compression. Describe, exemplify and implement prototyping (low fidelity) from wireframe design at this level.
Implementation (CSS)	Describe, exemplify and implement internal and external Cascading Style Sheets (CSS): ♦ selectors, classes and IDs ♦ properties — text: o font (family, size) o color o alignment — background colour Read and explain code that makes use of the above CSS.
Implementation (HTML)	Describe, exemplify and implement HTML code: ♦ HTML ♦ head ♦ title ♦ body ♦ heading ♦ paragraph ♦ DIV ♦ link ♦ anchor ♦ IMG ♦ audio ♦ video ♦ lists — ol, ul and li

	Describe and implement hyperlinks (internal and external), relative and absolute addressing. Read and explain code that makes use of the above HTML.
Implementation (Javascript)	Describe and identify Javascript coding related to mouse events: ♦ Onmouseover ♦ Onmouseout
Testing	Describe and exemplify testing: ♦ matches user-interface design ♦ links and navigation work correctly ♦ media (such as text, graphics, and video) display correctly ♦ consistency
Evaluation	Evaluate solution in terms of: ♦ fitness for purpose

Skills, knowledge and understanding included in the course are appropriate to the SCQF level of the course. The SCQF level descriptors give further information on characteristics and expected performance at each SCQF level (www.scqf.org.uk).

Skills for learning, skills for life and skills for work

This course helps candidates to develop broad, generic skills. These skills are based on SQA's Skills Framework: Skills for Learning, Skills for Life and Skills for Work and draw from the following main skills areas:

2 Numeracy

2.1 Number processes
2.3 Information handling

4 Employability, enterprise and citizenship

4.2 Information and communication technology (ICT)

5 Thinking skills

5.3 Applying
5.4 Analysing and evaluating

These skills must be built into the course where there are appropriate opportunities and the level should be appropriate to the level of the course.

Further information on building in skills for learning, skills for life and skills for work is given in the course support notes.

Course assessment

Course assessment is based on the information provided in this document.

The course assessment meets the key purposes and aims of the course by addressing:

- breadth — drawing on knowledge and skills from across the course
- challenge — requiring greater depth or extension of knowledge and/or skills
- application — requiring application of knowledge and/or skills in practical or theoretical contexts as appropriate

This enables candidates to apply:

- breadth of knowledge from across the course and depth of understanding, to answer appropriately challenging questions in computing science contexts
- knowledge and skills developed through the course, to solve appropriately challenging computing science problems in both practical and theoretical contexts

Course assessment structure: question paper

Question paper 110 marks

The question paper gives candidates an opportunity to demonstrate the following skills, knowledge and understanding:

- applying aspects of computational thinking, across a range of contexts
- analysing problems within computing science, across a range of contemporary contexts
- designing, implementing, testing and evaluating digital solutions (including computer programs) to problems, across a range of contemporary contexts
- communicating how a program works
- communicating key concepts related to computing science clearly and concisely, using appropriate terminology
- understanding the legal implications and environmental impact of contemporary technologies
- applying computing science concepts and techniques to create solutions, across a range of contexts

The question paper has 110 marks, which is 69% of the overall marks for the course assessment (160 marks).

A proportion of marks are available for more challenging questions and may require integration, detailed descriptions or explanations, and/or analysis, comparisons, and evaluations.

Marks are distributed across all four areas of study:

- Software design and development (approximately 40%)
- Computer systems (approximately 10%)
- Database design and development (approximately 25%)
- Web design and development (approximately 25%)

The question paper has two sections. Candidates are required to answer all the questions in both sections.

Section 1 has 25 marks and consists of short-answer, restricted response questions. This section allows candidates to demonstrate breadth of knowledge from across the four areas of the course.

Section 2 has 85 marks and consists of structured questions consisting of restricted and extended response. This section allows candidates to demonstrate application of knowledge and understanding when answering appropriately challenging context-based questions from across the four areas of the course.

Questions in this section:

- assess application of understanding, with very few questions requiring direct recall of knowledge
- sample across the course in a balanced way
- consist of questions set in meaningful contexts, that require candidates to provide some descriptions and explanations
- include some structured questions that draw on understanding from two or more topics, providing integration

SQA's standardised reference language

Questions assessing understanding and application of programming skills are expressed using SQA's standardised reference language. Further information can be found in the document '*Reference language for Computing Science question papers*' which can be downloaded from the National 5 Computing Science subject page on SQA's website.

Where candidates are required to answer by writing code, answers may be expressed using any programming language. Candidates are not expected to write code in SQA's standardised reference language. Marks are awarded for demonstrating understanding, not for the correct use of syntax.

Setting, conducting and marking the question paper

The question paper is set and marked by SQA.

It is conducted in centres under conditions specified for external examinations by SQA. Candidates complete the paper in 2 hours.

Specimen question papers for National 5 courses are published on SQA's website. These illustrate the standard, structure and requirements of the question papers candidates sit. The specimen papers also include marking instructions.

Course assessment structure: assignment

Assignment 50 marks

The assignment gives candidates an opportunity to demonstrate the following skills, knowledge and understanding:

- applying aspects of computational thinking across a range of contexts
- analysing problems within computing science across a range of contemporary contexts
- designing, implementing, testing and evaluating digital solutions (including computer programs) to problems across a range of contemporary contexts
- developing skills in computer programming
- applying computing science concepts and techniques to create solutions across a range of contexts

The assignment has 50 marks, which is 31% of the overall marks for the course assessment (160 marks).

The assignment is made up of three distinct tasks. Marks are distributed across three areas of study covered by the assignment as follows:

- Software design and development (25 marks)
- Database design and development (10–15 marks)
- Web design and development (10–15 marks)

Marks are distributed across the five skills covered by the assignment as follows:

- Analysis (5 marks)
- Design (5 marks)
- Implementation (30 marks)
- Testing (5 marks)
- Evaluation (5 marks)

Implementation (including writing code) is assessed in all three areas of study covered by the assignment. The other four skills are sampled in different areas each year.

A proportion of marks are available for the more challenging aspects of each task, where candidates are required to demonstrate problem-solving skills.

Setting, conducting and marking the assignment

The assignment is:

- set by SQA, on an annual basis
- conducted under a high degree of supervision and control

Evidence is submitted to SQA for external marking and all marking is quality assured by SQA.

Specimen assessment tasks for National 5 courses are published on SQA's website. These illustrate the standard, structure and requirements of the assessment tasks candidates complete. The specimen assessment tasks also include marking instructions.

Assessment conditions

Time

The assignment must be carried out within 8 hours, starting at an appropriate point in the course and once all content has been delivered. It is not anticipated that this is a continuous 8-hour session but conducted over several shorter sessions.

Supervision, control and authentication

The assignment is conducted under open-book conditions, but supervised to ensure that the work presented is the candidate's own work.

At the end of each session, and upon completion of the assignment, the teacher must ensure that candidate evidence is stored securely.

Resources

Each candidate must have access to a computer system with a high-level (textual) programming language and software that can run SQL, HTML and CSS.

The assignment is conducted under open-book conditions, which means candidates are permitted to access resources such as programming manuals, class notes, textbooks and programs they have written throughout the course.

Reasonable assistance

The assignment consists of three independent tasks. They are designed in a way that does not require teachers to provide support to candidates, other than to ensure that they have access to the necessary resources within the centre.

Once the assignment has been completed, it must not be returned to the candidate for further work to improve their mark.

Evidence to be gathered

All candidate evidence (whether created manually or electronically) must be submitted to SQA in a paper-based format. This includes hard copies of program listings, screenshots or similar, as appropriate.

Volume

There is no word count.

Grading

A candidate's overall grade is determined by their performance across the course assessment. The course assessment is graded A–D on the basis of the total mark for all course assessment components.

Grade description for C

For the award of grade C, candidates will typically have demonstrated successful performance in relation to the skills, knowledge and understanding for the course.

Grade description for A

For the award of grade A, candidates will typically have demonstrated a consistently high level of performance in relation to the skills, knowledge and understanding for the course.

Equality and inclusion

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

For guidance on assessment arrangements for disabled candidates and/or those with additional support needs, please follow the link to the assessment arrangements web page: www.sqa.org.uk/assessmentarrangements.

Further information

The following reference documents provide useful information and background.

- National 5 Computing Science subject page
- Assessment arrangements web page
- Building the Curriculum 3–5
- Design Principles for National Courses
- Guide to Assessment
- SCQF Framework and SCQF level descriptors
- SCQF Handbook
- SQA Skills Framework: Skills for Learning, Skills for Life and Skills for Work
- Coursework Authenticity: A Guide for Teachers and Lecturers
- Educational Research Reports
- SQA Guidelines on e-assessment for Schools
- SQA e-assessment web page

Administrative information

Published: May 2017 (version 1.1)

History of changes to course specification

Version	Description of change	Authorised by	Date
1.1	Included information on the Core Skill 'Information and Communication Technology at SCQF level 5' in the course overview section.	Qualifications Manager	May 2017