Higher
Course
Specification

Higher Computing Science

Course code:	C816 76
Course assessment code:	X816 76
SCQF:	level 6 (24 SCQF credit points)
Valid from:	session 2018–19

This document 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: April 2018 (version 1.0)

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 candidates to complete the course is 160 hours.

The course assessment has two components.

Component	Marks	Duration
Component 1: question paper	110	2 hours and 30 minutes
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 National 5 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.

Course rationale

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

This course highlights the central role of computing professionals as problem-solvers and designers, and the far-reaching impact of information technology on our environment and society.

It provides candidates with an understanding of the technologies and develops a wide range of practical skills that underpin our modern, digital world. The course also builds awareness of the importance of computing in meeting our needs today and for the future, in many fields including science, education, business and industry.

Purpose and aims

The course introduces candidates to an advanced range of computational processes, where they learn to apply a rigorous approach to the design and development process across a variety of contemporary contexts. They also gain an awareness of the important role that computing professionals play in meeting the needs of society today and for the future.

The course enables candidates to:

- develop and apply aspects of computational thinking in a range of contemporary contexts
- apply knowledge and understanding of advanced concepts and processes in computing science
- apply skills and knowledge in analysis, design, implementation, testing and evaluation to a range of digital solutions with some complex aspects
- communicate advanced computing concepts and explain computational behaviour clearly and concisely, using appropriate terminology
- develop awareness of current trends in computing technologies and their impact in transforming and influencing our environment and society

Who is this course for?

The course is suitable for candidates interested in exploring the role and impact of contemporary computing technologies. It provides an insight into the challenge, excitement and rewards found in these areas.

Course content

The course has four areas of study:

Software design and development

Candidates develop knowledge and understanding of advanced concepts and practical problem-solving skills in software design and development. They do this by using appropriate modular software development environments. Candidates develop modular programming and computational-thinking skills by analysing, designing, implementing, testing, and evaluating practical solutions and explaining how these programs work. They use their knowledge of data types and constructs to create efficient programs to solve advanced problems.

Computer systems

Candidates develop their understanding of how data and instructions are stored in binary form and factors affecting system performance. They gain an awareness of the environmental impact of intelligent systems, as well as the security risks, precautions and laws that can protect computer systems.

Database design and development

Candidates develop knowledge, understanding and advanced practical problem-solving skills in database design and development. They do this through a range of practical tasks, using a minimum of three linked tables and implemented in SQL. Candidates apply computational-thinking skills to analyse, design, implement, test, and evaluate practical solutions, using a range of development tools. Candidates apply interpretation skills to tasks involving some complex features in both familiar and new contexts.

Web design and development

Candidates develop knowledge, understanding and advanced practical problem-solving skills in web design and development. They do this through a range of practical and investigative tasks. Candidates apply computational-thinking skills to analyse, design, implement, test, and evaluate practical solutions to web-based problems, using a range of development tools including HTML, Cascading Style Sheets (CSS) and JavaScript. Candidates apply interpretation skills to tasks involving some complex features in both familiar and new contexts.

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 computational thinking to understand problems across a range of contexts
- analysing problems with some complex aspects within computing science across a range of contemporary contexts
- designing, implementing, testing and evaluating digital solutions (including computer programs) to problems with some complex aspects 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 advanced concepts related to software design and development, and information system design and development, clearly and concisely, using appropriate terminology
- understanding and evaluating the legal and environmental impact of contemporary computing 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 compare the development methodologies: ♦ iterative development process ♦ agile methodologies
Analysis	Identify the: ♦ purpose ♦ scope ♦ boundaries ♦ functional requirements of a problem that relates to the design and implementation at this level, in terms of: ♦ inputs ♦ processes ♦ outputs

Software design and development	
Design	Identify the data types and structures required for a problem that relates to the implementation at this level. Read and understand designs of solutions to problems at this level, using the following design techniques: ♦ structure diagrams ♦ pseudocode Exemplify and implement efficient design solutions to a problem, using a recognised design technique, showing: ♦ top level design ♦ the data flow ♦ refinements 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 structures: ♦ parallel 1D arrays ♦ records ♦ arrays of records
Implementation (computational constructs)	Describe, exemplify and implement the appropriate constructs in a procedural high-level (textual) language: ♦ parameter passing (formal and actual) ♦ the scope of local and global variables ♦ sub-programs/routines, defined by their name and arguments (inputs and outputs): — functions — procedures ♦ pre-defined functions (with parameters): — to create substrings — to convert from character to ASCII and vice versa — to convert floating-point numbers to integers — modulus ♦ file handling: — sequential CSV and txt files (open, create, read, write, close) Read and explain code that makes use of the above constructs.

Software design and development	
Implementation (algorithm specification)	Describe, exemplify and implement standard algorithms using 1D arrays or arrays of records: ♦ linear search ♦ find minimum and maximum ♦ count occurrences
Testing	Describe, exemplify and implement a comprehensive final test plan to show that the functional requirements are met. Identify syntax, execution, and logic errors at this level. Describe and exemplify debugging techniques: ♦ dry runs ♦ trace tables/tools ♦ breakpoints ♦ watchpoints
Evaluation	Describe, identify and exemplify the evaluation of a solution in terms of: ♦ fitness for purpose ♦ efficient use of coding constructs ♦ usability ♦ maintainability ♦ robustness

Computer systems	
Data representation	Describe and exemplify the use of binary to represent positive and negative integers using two's complement, including the range of numbers that can be represented using a fixed number of bits. Conversion of two's complement numbers from binary to denary and vice versa. Describe and exemplify floating-point representation of positive and negative real numbers, using the terms mantissa and exponent. Describe the relationship between the number of bits assigned to the mantissa/exponent, and the range and precision of floating-point numbers. Describe Unicode used to represent characters and its advantage over extended ASCII code (8-bit) in terms of numbers of characters. Describe the relative advantages and disadvantages of bit-mapped graphics versus vector graphics.
Computer structure	Describe the concept of the fetch-execute cycle. Describe the factors affecting computer system performance: ♦ number of processors (cores) ♦ width of data bus ♦ cache memory ♦ clock speed
Environmental impact	Describe the environmental impact of intelligent systems: ♦ heating systems ♦ traffic control ♦ car management systems
Security risks and precautions	Describe and identify the implications for individuals and businesses of the Computer Misuse Act 1990: ♦ unauthorised access to computer material ♦ unauthorised access with intent to commit a further offence ♦ unauthorised modification of programs or data on a computer Describe and identify the security risks of: ♦ tracking cookies ♦ DOS (Denial of Service) attacks: — symptoms o slow performance, inability to access

Computer systems	

	— effects
	  o disruption to users and businesses
	— costs
	  o lost revenue, labour in rectifying fault
	— type of fault
	  o bandwidth consumption, resource starvation, Domain Name Service (DNS)
	— reasons
	  o financial, political, personal
	
	Describe how encryption is used to secure transmission of data:
	
	♦ use of public and private keys
	♦ digital certificates
	♦ digital signatures

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 exemplify entity-relationship diagrams with three or more entities, indicating: ♦ entity name ♦ attributes ♦ name of relationship ♦ cardinality of relationship (one-to-one, one-to-many, many-to-many) Describe and exemplify an instance using an entity-occurrence diagram. Describe and exemplify a compound key. Describe and exemplify a data dictionary with three or more entities: ♦ 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 a query: ♦ tables and queries ♦ fields ♦ search criteria ♦ sort order ♦ calculations ♦ grouping

Software design and development	
Implementation	Describe, exemplify and use SQL operations for pre-populated relational databases, with three or more linked tables: ♦ UPDATE, SELECT, DELETE, INSERT statements making use of: — wildcards — aggregate functions (MIN, MAX, AVG, SUM, COUNT) — computed values, alias — GROUP BY — ORDER BY — WHERE 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 at this level 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 of a multi-level website with a home page and two additional levels, with no more than four pages per level. Describe, exemplify and implement, taking into account end-user requirements and device type, an effective user-interface design (visual layout and readability) using wire-framing: ♦ horizontal navigational bar ♦ relative horizontal and vertical positioning of the media ♦ form inputs ♦ file formats of the media (text, graphics, video, and audio) Describe, exemplify and implement prototyping (low-fidelity) from wireframe design at this level.
Implementation (CSS)	Describe, exemplify and implement efficient inline, internal and external Cascading Style Sheets (CSS) using grouping and descendant selectors to: ♦ control appearance and positioning: — display (block, inline, none) — float (left, right) — clear (both) — margins/padding — sizes (height, width) ♦ create horizontal navigation bars: — list-style-type:none — hover Read and explain code that makes use of the above CSS.
Implementation (HTML)	Describe, exemplify and implement HTML code: ♦ nav ♦ header ♦ footer ♦ section ♦ main ♦ form ♦ id attribute

Web design and development	
	Describe, exemplify and implement form elements: ♦ form element: input — text — number — textarea — radio — submit ♦ form element: select Describe, exemplify and implement form data validation: ♦ length ♦ presence ♦ range Read and explain code that makes use of the above HTML.
Implementation (JavaScript)	Describe, exemplify and implement coding of JavaScript functions related to mouse events: ♦ onmouseover ♦ onmouseout ♦ onclick
Testing	Describe, exemplify and implement usability testing using personas, test cases and scenarios based on low-fidelity prototypes. Describe and exemplify testing: ♦ input validation ♦ navigational bar works ♦ media content displays correctly Describe and exemplify compatibility testing: ♦ device type: — tablet, smartphone, desktop ♦ browser
Evaluation	Evaluate solution at this level in terms of: ♦ fitness for purpose ♦ usability

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, and can be found on the SCQF website.

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

You must build these skills into the course at an appropriate level, where there are suitable opportunities.

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 knowledge and skills developed through the course to:

- solve appropriately challenging, practical computing science problems
- answer appropriately challenging questions in computing science contexts

Course assessment structure: question paper

Question paper 110 marks

The question paper gives candidates the opportunity to demonstrate their ability to:

- apply computational thinking to understand problems, across a range of contexts
- analyse computing science problems with some complex aspects, across a range of contemporary contexts
- design, implement, test and evaluate digital solutions (including computer programs) to problems, across a range of contemporary contexts
- communicate how a program works in technical detail
- communicate understanding of advanced concepts related to computing science clearly and concisely, using appropriate terminology
- understand the legal and environmental impact of contemporary computing technologies
- apply computing science concepts and techniques to create solutions, across a range of contexts

The question paper has a total mark allocation of 110 marks. This is 69% of the overall marks for the course assessment.

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%

A proportion of marks are available for more challenging questions that may require candidates to integrate, provide detailed descriptions or explanations, and/or analyse, compare, and evaluate.

The question paper has two sections containing questions that sample from the 'Skills, knowledge and understanding for the course assessment' detailed in this document. Candidates must answer all the questions.

Section 1 is worth 25 marks and consists of short-answer, restricted-response questions from across all four areas of the course.

Section 2 is worth 85 marks and consists of structured, restricted-response and extended-response questions from across all four areas of the course. Questions in this section:

- assess application of understanding (very few questions require 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, providing integration by drawing on understanding from two or more topics

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 Higher Computing Science subject page on SQA's website.

Where candidates need 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, and conducted in centres under conditions specified for external examinations by SQA.

Candidates have 2 hours and 30 minutes to complete the question paper.

Specimen question papers for Higher 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 their ability to:

- apply aspects of computational thinking across a range of contexts
- analyse problems within computing science across a range of contemporary contexts
- design, implement, test and evaluate digital solutions (including computer programs) to problems across a range of contemporary contexts
- demonstrate skills in computer programming
- apply computing science concepts and techniques to create solutions across a range of contexts

The assignment has a total mark allocation of 50 marks. This is 31% of the overall marks for the course assessment.

The assignment has three distinct tasks, with marks distributed across the following areas of study:

♦ software design and development	25 marks
♦ database design and development	10–15 marks
♦ web design and development	10–15 marks

Candidates gain marks for the following skills:

♦ analysis	5 marks
♦ design	5 marks
♦ implementation	30 marks
♦ testing	5 marks
♦ evaluation	5 marks

Note: implementation (including writing code) is assessed in all three tasks every year. The other four skills are sampled in different tasks 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
- submitted to SQA for external marking

All marking is quality assured by SQA.

The specimen assessment for the course is published on SQA's website. This illustrates the standard, structure and requirements of the assessment task candidates complete. The specimen assessment task also includes 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 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, teachers and lecturers 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 or lecturers to provide support to candidates, other than to ensure that they have access to the necessary resources within the centre.

Once the assignment is complete, it must not be returned to the candidate for further work to improve their mark.

Evidence to be gathered

Candidate evidence includes program listings, screenshots or similar, as appropriate.

Volume

There is no word count.

Grading

Candidates' overall grades are 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.

- Higher Computing Science subject page
- Assessment arrangements web page
- Building the Curriculum 3–5
- Guide to Assessment
- Guidance on conditions of assessment for coursework
- 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

The SCQF framework, level descriptors and handbook are available on the SCQF website.

Administrative information

Published: April 2018 (version 1.0)

History of changes

Version	Description of change	Date

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