Course Support Notes

Higher Computing Science Course Support Notes

This document may be reproduced in whole or in part for educational purposes provided that no profit is derived from reproduction and that, if reproduced in part, the source is acknowledged. Additional copies of these *Course Support Notes* can be downloaded from SQA's website: **www.sqa.org.uk**.

Please refer to the note of changes at the end of this document for details of changes from previous version (where applicable).

Contents

Introduction

These support notes are not mandatory. They provide advice and guidance on approaches to delivering and assessing the Higher Computing Science Course. They are intended for teachers and lecturers who are delivering the Course and its Units. They should be read in conjunction with the *Course Specification*, the *Course Assessment Specification* and the *Unit Specification*s for the Units in the Course.

General guidance on the Course

Aims

As stated in the *Course Specification*, the aims of the Course are to enable learners to:

- develop and apply aspects of computational thinking in a range of contemporary contexts
- extend and apply knowledge and understanding of advanced concepts and processes in computing science
- apply skills and knowledge in analysis, design, implementation 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

Related to these aims, and underlying the study of computing science, are a number of unifying themes, including technological progress and trends, the relationship between software, hardware and system performance, and information representation and transfer as a core component of any computation. These are used to explore a variety of specialist areas through practical and investigative tasks.

This Course will also give learners the opportunity to develop thinking skills and skills in numeracy, employability, enterprise and citizenship.

Progression into this Course

Entry to this Course is at the discretion of the centre. However, learners would normally be expected to have attained some relevant skills and knowledge through prior experience.

Skills and knowledge developed through any of the following, while not mandatory, are likely to be helpful as a basis for further learning for this Course.

Other SQA qualifications

- National 5 Computing Science Course or relevant component Units

Other experience

Learners may have also relevant skills and knowledge gained through other prior learning, life and work experiences.

Skills, knowledge and understanding covered in this Course

This section provides further advice and guidance about skills, knowledge and understanding that could be included in the Course.

Note: teachers and lecturers should refer to the *Course Assessment Specification* for mandatory information about the skills, knowledge and understanding to be covered in this Course.

The mandatory skills may be developed throughout the Course. The table below shows where there are significant opportunities to develop these in individual Units.

Mandatory skills and knowledge	Software Design and Development	Information System Design and Development	Course assessment
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 clearly and concisely using appropriate terminology	✓		✓
communicating understanding of advanced concepts related to information systems design and development clearly and concisely using appropriate terminology		✓	✓
investigating and evaluating the legal, environmental, economic and social impact of contemporary computing technologies		✓	✓
applying computing science concepts and techniques to create solutions across a range of contexts	✓	✓	✓

Teachers/lecturers should ensure that learners are fully aware of the wide range of skills, knowledge and understanding that they are developing in the Units and Course as a whole.

It is also important to highlight any transferable learning that is taking place which supports the development of skills for learning, skills for life and skills for work.

Progression from this Course

This Course or its Units may provide progression to:

- Advanced Higher Computing Science Course
- National Certificate Group Awards in Computing, IT and related areas
- employment, apprenticeships and/or training in IT and related fields

and ultimately, for some, to:

- a range of computing-related Higher National Diplomas (HNDs)
- degrees in Computing, IT and related disciplines
- careers in Computing, IT and related disciplines

Hierarchies

Hierarchy is the term used to describe Courses and Units which form a structured progression involving two or more SCQF levels.

It is important that any content in a Course and/or Unit at one particular SCQF level is not repeated if a learner progresses to the next level of the hierarchy. The skills and knowledge should be able to be applied to new content and contexts to enrich the learning experience. This is for centres to manage.

The Course is designed in hierarchy with the corresponding Course at SCQF level 5 (National 5). The Computing Science Courses at both levels have the same structure of two Units with corresponding titles. Both Units — *Software Design and Development* and *Information System Design and Development* — are in hierarchy with the corresponding Units at National 5.

The design of the Units means that teachers may be able to design learning activities that are appropriate for a class with learners working at different levels.

Appendix 2 contains a table showing the relationship between the mandatory National 5 and Higher knowledge and understanding. This table may be useful for:

- designing and planning learning activities for mixed National 5/Higher groups
- ensuring seamless progression between levels
- identifying important prior learning for learners at Higher

Teachers should also refer to the Outcomes and Assessment Standards for each level when planning delivery.

Further advice on delivery to a group including National 5 and Higher learners is given in the next section of these support notes, with additional detailed guidance in the *Unit Support Notes.*

Approaches to learning and teaching

Computing Science, like all new and revised National Courses, has been developed to reflect Curriculum for Excellence values, purposes and principles.

The approach to learning and teaching developed by individual centres should reflect these principles. Learners should be encouraged to participate fully in active learning and practical activities by working together, analysing, investigating, debating and evaluating topics, problems and solutions while the teacher acts increasingly as a facilitator.

An appropriate balance of teaching methodologies should be used in the delivery of the Course and a variety of active learning approaches is encouraged, including the following:

Activity-based learning
Whole-class, direct teaching opportunities should be balanced by activity-based learning on practical tasks. An investigatory approach is encouraged, with learners actively involved in developing their skills, knowledge and understanding by investigating a range of real-life and relevant problems and solutions related to areas of study. Learning should be supported by appropriate practical activities, so that skills are developed simultaneously with knowledge and understanding.

Group work
Practical activities and investigations lend themselves to group work, and this should be encouraged. Learners engaged in collaborative group working strategies can capitalise on one another's knowledge, resources and skills by questioning, investigating, evaluating and presenting ideas to one another. While 'working as a team' is not specifically identified as one of the skills for learning, life and work for this Course and, therefore, not assessed, it is a fundamental aspect of working in the IT and related industries and so should be encouraged and developed by teachers.

Problem-based learning
Problem-based learning (PBL) is another strategy which will support a learner's progress through this Course. This method may be best utilised at the end of an Outcome or a topic where additional challenge is required to ensure learners are secure in their knowledge and understanding and to develop the ability to apply knowledge and skills in less familiar contexts. Learning through PBL develops a learner's problem-solving, decision-making, investigative skills, creative thinking, team working and evaluative skills.

Computational thinking
Computational thinking is recognised as a key skill set for all 21st century learners — whether they intend to continue with computing science or not. It involves a set of problem-solving skills and techniques used by software developers to write programs.

There are various ways of defining computational thinking. One useful structure is to group these problem-solving skills and techniques under five broad headings:

- **Abstraction:** seeing a problem and its solution at many levels of detail and generalising the information that is necessary. Abstraction allows us to represent an idea or a process in general terms (eg variables) so that we can use it to solve other problems that are similar in nature.
- **Algorithms:** the ability to develop a step-by-step strategy for solving a problem. Algorithm design is often based on the decomposition of a problem and the identification of patterns that help to solve the problem. In Computing Science as well as in mathematics, algorithms are often written abstractly, utilising variables in place of specific numbers.
- **Decomposition:** breaking down a task so that we can clearly explain a process to another person — or to a computer. Decomposing a problem frequently leads to pattern recognition and generalisation/abstraction, and thus the ability to design an algorithm.
- **Pattern recognition:** the ability to notice similarities or common differences that will help us make predictions or lead us to shortcuts. Pattern recognition is frequently the basis for solving problems and designing algorithms.
- **Generalisation:** realising that a solution to one problem may be used to solve a whole range of related problems.

Underpinning all of these concepts is the idea that computers are **deterministic**: they do exactly what you tell them to do. The corollary of this, of course, is that they can be understood.

Whilst computational thinking can be a component of many subjects, Computing Science is particularly well placed to deliver it. Teachers are encouraged to emphasise, exemplify and make these aspects of computational thinking explicit (at an appropriate level) wherever there are opportunities to do so throughout the teaching and learning of this Course and its Units.

Using online and outside resources

Throughout the teaching of this Course, the stimulation of learners' interest and curiosity should be a prime objective. Engagement with outside agencies or industry professionals can greatly enhance the learning process. Online resources, such as those listed in the individual *Unit Support Notes* may provide a valuable addition to teaching and learning activities, encouraging research, collation and storage of information and evaluation of these materials. The use of interactive multimedia learning resources, online quizzes, and web-based software can also be used to support teacher-led approaches.

Assessment activities, used to support learning, may usefully be blended with learning activities throughout the Course, for example by:

- sharing learning intentions/success criteria
- using assessment information to set learning targets and next steps
- adapting teaching and learning activities based on assessment information
- boosting learners' confidence by providing supportive feedback

Self- and peer-assessment techniques should be encouraged wherever appropriate.

Working towards Units and Course
Learning and teaching activities should be designed to develop both:

- skills and knowledge to the standard required by **each Unit** and to the level defined by the associated Outcomes and Assessment Standards
- ability to apply the breadth of knowledge, understanding and skills listed in the *Course Assessment Specification*, as required to complete **the Course assessment** successfully

Meeting the needs of all learners
Within any class, each learner will have individual strengths and areas for improvement.

For example, within a class, there may be learners capable of achieving a standard beyond Higher in certain aspects of the Course. Where possible, these learners should be encouraged and given the opportunity to deepen and broaden their computing skills and knowledge.

There may also be learners who are struggling to achieve Higher level in particular aspects of the Course. This may provide opportunities for teachers to consider the use of additional or peer support, with a more able student taking on a tutor-type role and assisting other learners to develop and reinforce their understanding of a particular topic.

It may be too that certain learners will only achieve National 5 in some of the Unit Outcomes. Teachers need to consider both the Outcomes and Assessment Standards, and the content tables in Appendix 2 of these notes, to identify the difference between standards required for National 5 and Higher.

The difference between National 5 and Higher is defined in terms of a higher level of skill. For example, in the *Information System Design and Development* Unit, Outcome 1 requires National 5 learners to 'develop information systems using appropriate development tools', by creating a structure, a user interface, writing or editing code, integrating media and identifying and rectifying errors, while at Higher learners are similarly required to develop information systems, but must also apply contemporary design and development methodologies, write code, and test systems against appropriate criteria.

When delivering this Course to a group of learners, with some learners working towards National 5 and others towards Higher, it may be useful for teachers to identify activities covering common knowledge and skills for all learners, and additional activities required for Higher learners.

Where Higher learners have studied National 5 in a previous year, it is important to provide them with new and different contexts for learning to avoid demotivation. For example, learners could work in a different type of development environment or language at Higher than they did for National 5.
Learning about Scotland and Scottish culture will enrich the learners' learning experience and help them to develop the skills for learning, life and work they will need to prepare them for taking their place in a diverse, inclusive and participative Scotland and beyond. Where there are opportunities to contextualise approaches to learning and teaching to Scottish contexts, teachers and lecturers should consider this.

Sequence of delivery of Units
The sequence of delivery of the Units within the Higher Computing Science Courses is at the discretion of the centre.

Units could be delivered in sequence or in parallel. One approach might be to alternate delivery of the Units' content to give variety to teaching and learning activities, eg programming in a software development environment alternating with development of an interactive website, and other possibilities exist.

Fitting the assignment into a Course plan
Whether the decision is taken to deliver Units sequentially or in parallel, it would be good practice to complete Units 1 and 2 before attempting the assignment in the Course assessment. This approach will give learners the opportunity to develop the skills and knowledge necessary to enable them to successfully attempt the assignment. However, it may be possible to begin work on the assignment at an earlier stage, but only where it is clear that learners have already gained the required skills and knowledge.

Advice on distribution of time
Learners, especially at Higher, should be expected to contribute their own time in addition to programmed learning time.

The distribution of time between the Units is a matter for professional judgement and is entirely at the discretion of the centre. Each Unit is likely to require an approximately equal time allocation, although this may depend on the learners' prior learning in the different topic areas.

Time should be allocated for preparation for the Course assessment (assignment and question paper). See below for further advice on use of this time.

Resources
Centres may find that existing hardware and software within the Computing Science classroom provides all that is required to deliver the Course. The suggested resources are summarised below:

- internet-enabled computers and a digital projector
- access to software development tools (one or more software development environments, virtual machines and emulators)
- access to application development software and tools (macro editors, applications that support data handling, presentation, group work, animation, video, graphics and text)
- web development tools (script-enabled browsers, wire-framing software, etc)
- digital media devices (scanners, digital cameras, camcorders, etc)

Teaching and learning materials
Centres may also be able to adapt existing activities and resources to support and consolidate learning. For example, existing Higher materials in programming can be adapted for the *Software Design and Development* Unit (eg standard algorithms are the same), and the same is true for practical tasks in the *Information System Design and Development* Unit (eg relational databases, web design, client-side scripting, etc). However, it is important to ensure that, when using any pre-existing materials in this way, the exemplification of the new Assessment Standards is continually referenced.

Developing skills for learning, skills for life and skills for work

Guidance on the development of skills for life, skills for learning and skills for work is to be found in the *Unit Support Notes* for each of the Units.

Approaches to assessment

See the *Unit Support Notes* for guidance on approaches to assessment of the Units of the Course.

Added value

Courses from National 4 to Advanced Higher include assessment of added value. At Higher, the added value will be assessed in the Course assessment.

Information given in the *Course Specification* and the *Course Assessment Specification* about the assessment of added value is mandatory.

Full details of assessment of added value are included in the *Course Assessment Specification.*

The Course assessment (question paper and assignment) will assess the application of skills and knowledge which learners will have developed through the other Units.

Preparation for Course assessment

Each Course has additional time which may be used at the discretion of the teacher or lecturer to enable learners to prepare for Course assessment. This time may be used at various points throughout the Course for consolidation and support. It may also be used for preparation for Unit assessment, and towards the end of the Course, for further integration, revision and preparation and/or gathering evidence for Course assessment.

Information given in the *Course Specification* and the *Course Assessment Specification* about the assessment of added value is mandatory.

Within the time allocated for the Course assessment, the following activities might be useful:

- preparation for the assignment, which could include considering exemplar assignments and practising the application and integration of skills
- carrying out the stages of the assignment, with teacher guidance and support
- assessing the process and completed solution
- consolidation of learning
- development of skills in applying knowledge and understanding
- preparation for the question paper

Equality and inclusion

The requirement to develop practical skills involving the use of equipment and tools may present challenges for learners with physical or visual impairment. In such cases, reasonable adjustments may be appropriate, including (for example) the use of adapted equipment or alternative assistive technologies. This is for both learners and centres to consider.

It is recognised that centres have their own duties under equality and other legislation and policy initiatives. The guidance given in these Course *Support Notes* is designed to sit alongside these duties but is specific to the delivery and assessment of the Course.

It is important that centres are aware of and understand SQA's assessment arrangements for disabled learners, and those with additional support needs, when making requests for adjustments to published assessment arrangements.

Centres will find more guidance on this in the series of publications on Assessment Arrangements on SQA's website: www.sqa.org.uk/sqa/14977.html.

Appendix 1: Reference documents

The following reference documents will provide useful information and background.

- Assessment Arrangements (for disabled learners and/or those with additional support needs) — various publications are available on SQA's website at: www.sqa.org.uk/sqa//14977.html.
- Pseudocode for Higher Computing Science Question Paper: Computing Science Pseudocode
- British Computer Society, Glossary of Computing and IT, 12th edition
- *Building the Curriculum 4: Skills for learning, skills for life and skills for work*
- *Building the Curriculum 5: A framework for assessment*
- Course Specifications
- Design Principles for National Courses
- *Guide to Assessment* (June 2008)
- Principles and practice papers for curriculum areas
- *SCQF Handbook: User Guide* (published 2009) and SCQF level descriptors (reviewed during 2011 to 2012): www.sqa.org.uk/sqa/4595.html
- *SQA Skills Framework: Skills for Learning, Skills for Life and Skills for Work*

Appendix 2: Comparison of National 5 and Higher

This table shows the relationship between the mandatory National 5 and Higher knowledge and understanding. This table may be useful for:

- designing and planning learning activities for multi-level National 5/Higher classes
- ensuring seamless progression between levels
- identifying important prior learning for learners at Higher

Teachers should also refer to the Outcomes and Assessment Standards for each level when planning delivery.

NB: Where similar topics are covered at both levels, the Outcomes, Assessment Standards and Evidence Requirements distinguish the level of treatment.

Software Design and Development		
Topic	**National 5**	**Higher**
Languages and environments		♦ Description of the following language types: — low-level — high-level — procedural — declarative — object-oriented
Computational constructs	♦ Description, exemplification and implementation of the following constructs: — expressions to assign values to variables — expressions to return values using arithmetic operations (+, -, *, /, ^, mod) — expressions to concatenate strings and arrays using the and operator	♦ Description, exemplification and implementation of the following constructs: — parameter passing (value and reference, formal and actual) — the scope of local and global variables — sub-programs/routines, defined by their name and arguments (inputs and outputs), including functions and procedures

	— use of selection constructs including simple and complex conditional statements using logical operators (AND, OR, NOT) — iteration and repetition using fixed and conditional loops — pre-defined functions (with parameters) including Random, Integer and Round	
Data types and structures	♦ Description, implementation and exemplification of the following data types and structures: — character — string — numeric (integer and real) variables — Boolean variables — 1-D arrays	♦ Description, exemplification and implementation of the following data types and structures: — string — numeric (integer and real) variables — Boolean variables — 1-D arrays — records — arrays of records — sequential files (open, create, read, write, close)
Testing and documenting solutions	♦ Description, identification, exemplification and implementation of normal, extreme and exceptional test data. ♦ Description and identification of syntax, execution and logic errors. ♦ Description, identification and exemplification of the readability of code including: — internal commentary — meaningful identifiers — indentation — white space	♦ Description and implementation of constructing a comprehensive test plan for a specific problem. ♦ Description and identification of syntax, execution and logic errors. ♦ Description and exemplification of testing techniques including: — dry-runs — trace tables/tools — breakpoints — watchpoints

Algorithm specification	♦ Description, exemplification and implementation of algorithms, including: — input validation	♦ Analysis, description, exemplification and implementation of standard algorithms including: — linear search — find minimum and maximum — count occurrences ♦ Analysis of other programs of similar complexity.
Low-level operations and computer architecture	♦ Explanation of the need to translate high-level program code to binary (machine code). ♦ Comparison of interpreters and compilers. ♦ Description and exemplification of the use of binary to represent positive integers. ♦ Conversion from Binary to decimal and vice-versa. ♦ Description of floating point representation of real numbers using the terms mantissa and exponent. ♦ Description of ASCII code (7-bit) used to represent characters. ♦ Description of the vector graphics method of graphic representation. ♦ Description of the bit-mapped method of graphics representation. ♦ Describe the purpose of the Basic computer architecture components and how they are linked together including: — processor (registers, ALU, control unit) — memory — buses (data and address) — interfaces	♦ Description of the uses of virtual machines and emulators. ♦ Description and exemplification of the use of binary to represent negative integers using two's complement, including the range of numbers that can be represented using a fixed number of bits. ♦ Description of the relationship between the range and precision of real numbers using floating point representation. ♦ Description of Unicode used to represent characters and its advantage over ASCII. ♦ Description of the advantages and disadvantages of bit-mapped graphics compared to vector graphics. ♦ Understand that sound is represented in binary and described in terms of sample size and sample rate. ♦ Understand that video is represented as a sequence of still frames and described in terms, for each frame, of: — frame rate — resolution — bit depth ♦ Calculation of storage requirements for uncompressed audio and video. ♦ Describe the trends and implications of computer architecture including:

		— multi-core processors — parallel processing ♦ Describe the fetch-execute cycle using the components of computer architecture including: — processor (registers, ALU, control unit) — memory — buses (data, address and control)

Software Design and Development		
The following mandatory generic concepts and vocabulary may be applied to **both** software design and development and information system design and development:		
Topic	**National 5**	**Higher**
Design notations	♦ Description and identification of structure diagrams, flowcharts and pseudocode to solve problems. ♦ Exemplification of pseudocode to solve problems.	♦ Description, exemplification and implementation of pseudocode to solve problems. ♦ Description, exemplification and implementation of entity relationship diagrams. ♦ Exemplification and implementation of data dictionary including name, type, size, required and validation. ♦ Exemplification and implementation of wire-framing.
Development methodologies		♦ Description of the general iterative phases of the development process: analysis, design, implementation, testing, documentation, evaluation, maintenance. ♦ Description identification and benefits of development methodologies including: — rapid application development — top-down/step-wise refinement — Agile technologies
Contemporary developments	♦ Exemplification of trends in the development of: — software development languages — software development environments — their editing features — high-level code translation and execution	♦ Exemplification of trends in the development of: — software development languages — software development environments — intelligent systems — online systems

User interface	♦ Description of requirements for a good user interface including: — visual layout — navigation — selection — consistency — interactivity — readability	♦ Description of problems with accessibility of computer systems and how they can be overcome including: — vision impairments — hearing impairment — motor and dexterity impairments

Information System Design and Development		
Topic	**National 5**	**Higher**
The following mandatory generic concepts and vocabulary may be applicable to a range of information systems:		
Structures and links (database)	♦ Implementation of a relational database with two linked data tables. ♦ Advantages of relational database over flat-file databases. ♦ Description, exemplification and implementation of primary keys and foreign keys. ♦ Description and exemplification of field types (text, number, date, time, graphic, object, link, Boolean). ♦ Description and exemplification of validation including: — presence check — restricted choice — field length — range ♦ Description and exemplification of database operations search, sort (on multiple fields) and calculations. ♦ Description, exemplification and implementation of good design to avoid data duplication and modification errors (insert, delete, update).	♦ Implementation of relational databases with a minimum of three linked data tables. ♦ Description, implementation and exemplification of compound keys and surrogate keys. ♦ Description, exemplification and identification of entity relationships (one-to-one, one-to-many, many-to-many). ♦ Description and implementation of complex database operations including: — input (forms) — searching/sorting/calculations (queries) — outputs (reports)
Structures and links	♦ Description of website, page, URL in relation to a web-based information system.	♦ Description, exemplification and implementation of the site structure of multi-level web-based information

(web-based)	♦ Description and implementation of hyperlinks (internal, external), relative and absolute addressing. ♦ Understand the need for, and exemplify, simple navigation within web-based information systems. ♦ Description and features of web browsers and search engines.	system. ♦ Description and implementation of the page structure of web-based information system including head, title and body. ♦ Description, exemplification and implementation of cascading style sheets with rules for: — element formatting and placement — classes and ID's — inline rules, internal and external stylesheets ♦ Understand the composition of meta tags and how they are used in search engine optimisation. ♦ Description and advantages/disadvantages of dynamic web pages and database-driven website. ♦ Description and exemplification of interactive web pages.
Media types	♦ Description of standard file formats and their benefits. ♦ Know a range of standard file formats for different media types including: — the text standard file formats txt, rtf — the audio standard file formats wav, mp3 — the graphic standard file formats jpeg, gif, png, svg — the video standard file formats mp4, avi ♦ Describe and exemplify the factors affecting file size and quality, including resolution, colour depth, sampling rate. ♦ Calculation of file size for colour bitmap. ♦ Description of the need for compression.	♦ Description of difference between lossy and lossless compression. ♦ Description and identification of a number of compression techniques including: — perceptual coding — audio lossy compression technique — Free Lossless Audio Codec — lossless compression technique — RLE — graphic lossless compression technique — LZW encoding — graphic lossless compression technique — DCT encoding — graphic lossy compression technique — interframe and intraframe video compression techniques

Coding	♦ Description and identification of coding to create and modify information systems including: — JavaScript mouse events ♦ Description, exemplification and implementation of coding to create and modify information systems including the use of HTML with the tags for: — document — links — graphics	♦ Description, exemplification and implementation of coding to create and modify information systems including the use of: — scripting (database/web pages) — client-side scripting using JavaScript mouse events ♦ Description of the role of server-side scripting in the generation of dynamic web pages and database-driven websites including: — receiving user input/selection from a client device — validation of form data — connecting to database server — page generation
Testing	♦ Description and exemplification of testing information systems including: — links and navigation work correctly — matches user interface design — media such as text, graphics and video display correctly	♦ Understand the process and benefits of beta testing. ♦ Describe the process and benefits of usability testing. ♦ Understand that compatibility issues may occur within information systems including: — sufficient memory and storage requirements — compatibility with the operating system
Purpose, features, functionality, users	♦ Description of purpose of an information system. ♦ Description of the features and functions of an information system. ♦ Description of types of users of information systems including: — expert — novice ♦ Description of age-range of users of information systems.	♦ Descriptions of purpose, functions, features and appropriate users of a specific information system. ♦ Description of the interaction of information systems with search engines.

Technical implementation (hardware requirements)	♦ Description and exemplification of the appropriate type of hardware required for a specific information system including: — input and output devices — processor type and speed (Hz) — memory capacity (RAM)	♦ Description and exemplification of the appropriate hardware required for a specified information system including: — input and output devices — processor type, number and speed (Hz) — memory (RAM, cache)
Technical Implementation (software requirements)	♦ Describe the purpose of an operating system including: — controlling peripherals — running software — HCI ♦ Understand the features of web browsers including: — OS support — privacy modes — Ad filtering — page zooming ♦ Description and exemplification of the appropriate type of software required for a specific information system including: — type of application — operating system	♦ Description of the main functions of an operating system including: — interpreting user commands — file management — memory management — input/output management — resource allocation ♦ Description and comparisons of proprietary versus open-source software licenses. ♦ Understand the benefits of portability for computer programs and information systems. ♦ Description and exemplification of current trends in operating system design. ♦ Description and exemplification of the appropriate type of software required for a specific information system including: — type of application — operating system
Technical implementation (storage)	♦ Comparison of local versus cloud storage. ♦ Comparison of built-in versus portable storage. ♦ Comparison of different interface types and their data transfer speeds including:	♦ Description and benefits of distributed storage. ♦ Description and benefits of offline storage. ♦ Description of the advantages/disadvantages of cloud systems compared to local server provision including: — cost

	— Firewire — USB — bandwidth ♦ Description of different types of storage devices and their media in terms of functionality and capacity (in appropriate units) including: — magnetic — optical — solid state ♦ Description and exemplification of the appropriate type of storage required for a specific information system including: — type of device — capacity — interface type	— accessibility — maintenance ♦ Description and comparison between public, private and hybrid cloud systems. ♦ Description of backup systems and strategy including: — schedule — frequency, differential, incremental — media — DAT, DTL, optical — location — on-site, off-site repository, cloud — mirroring (RAID) ♦ Description and exemplification of the appropriate type of storage required for a specific information system including: — type of device — capacity — interface type and data transfer speed ♦ Description and exemplification of current trends in storage systems.
Technical implementation (networking/ connectivity)	♦ Description and comparison of the following transmission media in relation to data speeds and ease of use: — wired — optical — wireless ♦ Description and exemplification of hardware required for network connectivity including: — Network Interface Card — router — hub — switch	♦ Description and exemplification of cloud-based services including: — data storage — mail services — software updates ♦ Description and exemplification of web hosting. ♦ Description and exemplification of current trends in networking and connectivity including: — bandwidth — transmission media — hardware such as hubs, switches and routers ♦ Description and exemplification of the appropriate type of network connection required for a specific information

	♦ Description and exemplification of the appropriate type of network connection required for a specific information system including: — hardware — transmission media	system including: — hardware — transmission media — bandwidth
Security risks	♦ Description and identification of the following security risks: — phishing — keylogging (software and hardware) — virus — online fraud — identity theft	♦ Description, identification and exemplification of spyware including: — Trojans — Adware — tracking cookies ♦ Description and exemplification of DOS (Denial of Service) attacks including: — symptoms — slow performance, inability to access — effects — disruption to users — costs — lost revenue, labour in rectifying fault — type of fault — bandwidth consumption, resource starvation, routing, Domain Name Service(DNS) — reasons — financial, political, personal
Security precautions	♦ Description and exemplification of anti-virus software. ♦ Description and exemplification of good practice in passwords settings. ♦ Description and exemplification of biometrics including: — retina scanning — finger prints — palm prints — face recognition ♦ Description and exemplification of firewalls.	♦ Description and exemplification of encryption used to secure transmission of data including use of public and private keys. ♦ Description and exemplification of digital certificates and signatures.

Legal implications	♦ Description, identification and implications for individuals and businesses of the Computer Misuse Act including: — use of software and hardware to access data unlawfully — impairing of operation of computer systems ♦ Description, identification and implications for individuals and businesses of the Data Protection Act including: — data in electronic transmission — prior consent of data subject — export of data ♦ Description, identification and implications for individuals and businesses of the Copyright, Designs and Patents Act (plagiarism) including: — copyright of computer software — software piracy — web content — text, graphics, video, audio ♦ Description, identification and implications for individuals and businesses of the Communication Acts including: — post of offensive information on social network sites — send offensive, indecent or threatening messages on a public electronic communications network — use networks without permission	♦ Description, identification and implications for individuals, businesses and ISP's of the Regulation of Investigatory Powers Act including: — intercepting and monitoring of electronic communications by government bodies — monitoring of employees communications — equipment and services used for surveillance

Environmental impact/ implications	♦ Description and implications of the impact of the energy use of computer systems and how it could be reduced including: — settings on monitors — power down settings — leaving computers on stand-by ♦ Description and exemplification of the correct ways to dispose of IT equipment including: — recycle individual components appropriately — extraction of dangerous elements — re-use of systems for other uses	♦ Description and implications of the lifetime carbon footprint including: — manufacture of computer systems and peripherals — electricity use during a computer system's lifetime — disposal including re-cycling and extraction of dangerous elements ♦ Description and implications of environmental benefits of computer systems including: — reduction in paper use in offices, etc — reduction in manufacturing/transportation due to increased downloading of music and books — reduction in travelling through working from home — intelligent control of heating systems
Economic and social impact		**Economic impact:** ♦ Description and exemplification of the competitive advantage computer systems give businesses. ♦ Implications of the global marketplace for business and customers. ♦ Description and exemplification of business costs involved in the maintainability and scalability of information systems including: — training — hardware — software — storage — connectivity **Social impact:** ♦ Comparison between censorship and freedom of speech in relation to the internet.

		♦ Exemplification of the safeguards required to ensure privacy when using information systems such as social media sites. ♦ Understand the advantages of global citizenship. ♦ Exemplification of advantages and disadvantages of online communities.

A similar table in the National 5 Computing Science *Course Support Notes* shows the relationship between the mandatory National 5 and National 4 knowledge and understanding.

Administrative information

Published: May 2016 (version 2.1)

History of changes to Course Support Notes

Version	Description of change	Authorised by	Date
1.1	Changes and clarification of mandatory content to correspond with National 5 and Higher Course Assessment Specifications.	Qualifications Development Manager	June 2014
2.0	Changes made to 'Approaches to learning and teaching' section. The 'Data Protection Act' has been added to the requirements for Course coverage for the Information System Design and Development Unit in Appendix 2.	Qualifications Manager	June 2015
2.1	Adjustments made 'Appendix 2: Comparison of National 5 and Higher' to reflect the changes already made to both Course Assessment Specifications.	Qualifications Manager	May 2016

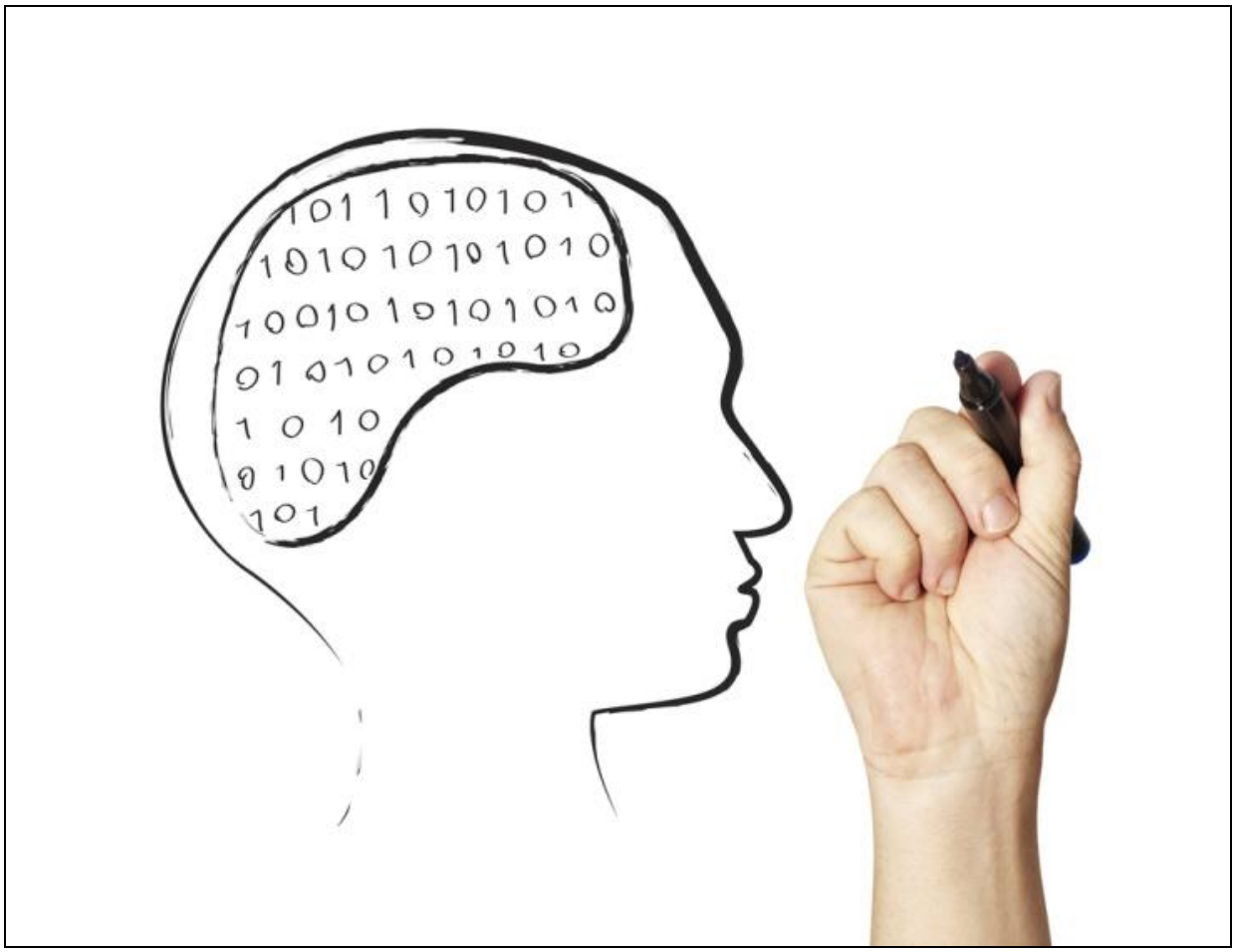
This document may be reproduced in whole or in part for educational purposes provided that no profit is derived from reproduction and that, if reproduced in part, the source is acknowledged. Additional copies can be downloaded from SQA's website at **www.sqa.org.uk**.

Note: You are advised to check SQA's website (**www.sqa.org.uk**) to ensure you are using the most up-to-date version.

© Scottish Qualifications Authority 2016

Unit Support Notes

Unit Support Notes — Software Design and Development (Higher)



This document may be reproduced in whole or in part for educational purposes provided that no profit is derived from reproduction and that, if reproduced in part, the source is acknowledged. Additional copies of these *Unit Support Notes* can be downloaded from SQA's website: **www.sqa.org.uk**.

Please refer to the note of changes at the end of this document for details of changes from previous version (where applicable).

Introduction

These support notes are not mandatory. They provide advice and guidance on approaches to delivering and assessing the *Software Design and Development* (Higher) Unit. They are intended for teachers and lecturers who are delivering this Unit. They should be read in conjunction with:

- the Unit Specification
- the Course Specification
- the Course Assessment Specification
- the Course Support Notes
- appropriate assessment support materials

General guidance on the Unit

Aims

The general aim of this Unit, as stated in the *Unit Specification*, is to develop knowledge and understanding of advanced concepts and practical problem-solving skills in software development through appropriate software development environments. Learners will develop their programming and computational thinking skills by designing, implementing, testing and evaluating practical solutions and explaining how these programs work. They will also develop an understanding of computer architecture and the concepts that underpin how programs work. Through investigative work, learners will gain an awareness of the impact of contemporary computing technologies.

This Unit will give learners the opportunity to develop their thinking skills and skills in numeracy, employability, enterprise and citizenship.

This Unit can be delivered:

- as a stand-alone Unit
- as part of the Higher Computing Science Course

Progression into this Unit

Entry to this Unit is at the discretion of the centre. However, learners would normally be expected to have attained some useful skills and knowledge from prior learning, such as:

- National 5 Computing Science Course or relevant component Units

Learners may also have gained relevant skills and knowledge through other education systems or from their own interests and informal learning.

Skills, knowledge and understanding covered in this Unit

Information about skills, knowledge and understanding is given in the Higher Computing Science *Course Support Notes.*

If the Unit is being delivered as part of the Higher Computing Science Course, the teacher should refer to the 'Further mandatory information on Course coverage' section within the *Course Assessment Specification* for detailed content.

If this Unit is being delivered on a free-standing basis, teachers and lecturers are free to select the skills, knowledge, understanding and contexts which are most appropriate for delivery in their centres.

Progression from this Unit

On successful completion of this Unit, the following Units and Courses provide appropriate progression pathways for learners:

- related Units in Advanced Higher Computing Science Course
- National Certificate Group Awards in a range of Computing, IT and related disciplines
- National Progression Awards in Digital Media
- employment, apprenticeships and/or training in Computing, IT and related fields

Approaches to learning, teaching and assessment

Learning and teaching
The Unit is designed to provide flexibility and choice for both the learner and the teacher.

The Higher Computing Science *Course Support Notes* provide further broad guidance on approaches to learning and teaching which may apply to all of the component Units of the Course and should be read before delivering this Unit.

Learning and teaching activities should be designed to stimulate learners' interest, and to develop skills and knowledge to the standard required by the Outcomes and to the level defined by the associated Assessment Standards. Learning should be supported by appropriate practical activities so that skills may be developed simultaneously with knowledge and understanding.

An investigatory approach is encouraged, with learners actively involved in developing their skills, knowledge and understanding of a range of real-life and relevant software development problems and solutions.

The *Unit Specification* defines the skills and knowledge required, but leaves complete freedom to the teacher and learner to select interesting contexts and environments in which to develop these. Tasks and activities throughout the Unit should be linked to relevant contexts and real-world applications, where appropriate. This provides scope for personalisation and choice, as relevant and motivating environments can be used. Aspects of existing software development solutions to real-world problems can be analysed to aid understanding.

When delivering the Unit as part of the Higher Computing Science Course, reference should be made to the appropriate content statements within the 'Further mandatory information on Course coverage' section of the *Course Assessment Specification* to ensure the required breadth of knowledge is covered.

Sequence of delivery of Outcomes
The sequence of delivery of the Outcomes is a matter of professional judgement and is entirely at the discretion of the centre. Some suggested approaches are outlined below.

Outcome 1 and Outcome 2 simultaneously
Teachers may wish to combine practical work with theory using a series of practical tasks that focus on the implementation of one or more programming constructs. Implementation could be accompanied by a written or oral explanation of how a construct is being used and of how it works.

For example, when learners are selecting and using appropriate constructs when developing their modular programs, they will need to understand the purpose and function of these constructs and how programs relate to low-level structures and operations.

As learners develop their programming skills by selecting and using appropriate constructs, they will also develop a clear understanding of how these constructs

work and what they can be used for. This will enable them to read, interpret and explain the code in their programs. This can be further extrapolated into how high-level code is translated to low-level, machine code instructions in binary and how binary is used to represent and store data types.

Outcome 1 before Outcome 2
In order to deepen the learners' understanding of advanced concepts and to enhance their ability to explain how programs work and how they relate to low-level structures and operations, it would constitute good practice to discuss the purpose of the range of constructs and algorithms used in this Unit and for learners to practice reading and interpreting code before progressing through a series of practical examples and tasks.

For example, learners could be provided with a range of working programs in a variety of software development environments illustrating particular programming constructs. They could then be asked to interpret and explain what is happening to certain sections of code. They would gain familiarity with a range of programming constructs and be able to describe the purpose of these constructs within the program and how these programs represent and store data, such as characters, integers and real numbers.

A similar exercise could take place using a range of standard algorithms. Teachers and lecturers could provide the learners with these algorithms and the learners would need to be able to describe how they work. Once the learners have a sound understanding of the purpose of a range of programming constructs and standard algorithms, and know how they function within a program, they should be well placed to develop their own modular programs that make use of these constructs and standard algorithms in their chosen software development environment. They should be able to apply their knowledge gained in Outcome 1 to select and use the appropriate programming constructs to produce a complete, working program.

Outcome 2 before Outcome 1
Alternatively, it may be preferable to first establish the necessary practical skills, to learn by doing, and meet the requirements of Outcome 2 before moving to formally develop the knowledge and understanding embedded in Outcome 1.

For example, learners could develop their programming skills in one or more software development environments by learning how to program certain constructs using a range of simple data types and 1-D arrays. These activities would allow the learners to fully understand the purpose and function of certain constructs and data types within the programs. Consequently, when learners are then asked to read and explain sections of code within a program in Outcome 1, they would have the necessary knowledge and skills to do this from their experience in coding in Outcome 2.

Meeting the needs of all learners
When teaching this Unit to a class, with some learners working towards National 5 and others towards Higher, it may be useful for teachers to identify activities covering common knowledge and skills for all learners, and additional activities required for Higher learners. For example:

Computational constructs
When learners are developing their skills and knowledge in computational constructs, both National 5 and Higher learners could undertake tasks where they have to:

- code expressions to return values using arithmetic operations
- code complex conditional statements and use logical operators
- code conditional loops
- use mathematical pre-defined functions

The Higher learners could be given additional tasks where they will have to:

- use sub-programs
- pass parameters
- use sequential file operations

Data types and structures
For data types and arrays, both National 5 and Higher learners will cover the same content, ie string, numeric and Boolean variables and 1-D arrays. However, Higher learners will use these data types in unfamiliar and more complex contexts, such as the study of standard algorithms. There is scope for the planning of both common and additional activities here.

Design notations
When National 5 and Higher learners are designing their programs, they may use pseudocode to exemplify program algorithms. The Higher learners could then look at other design notations, such as structure diagrams.

Testing and documenting solutions
Both sets of learners will be expected to document their programs using internal commentary but Higher learners will add meaningful identifiers and indentation. The Higher learners could then be set an extension task where they will test their solutions using their own test plan. They would also be required to explain the use of dry-runs, trace tables and break points.

Different contexts
Where Higher learners have studied National 5 in a previous year, it is important to provide them with new and different contexts for learning to avoid demotivation. It is particularly important that learners do not feel that they are simply doing the same work over again, albeit at a deeper level.

For example, when the Higher learners are designing their programs they will investigate a range of design notations that will be new to them. They will be introduced to a completely new set of computational programming constructs such as sub-programs, user-defined functions, parameter passing, sequential file operations and scope, local and global variables. They will test their programs in a different way, constructing test plans and using dry-runs and trace tables to locate errors and debug. Higher learners will also need to describe how a range of standard algorithms work which they will not have covered at National 5.

In line with the underlying principles of Curriculum for Excellence, learners should be encouraged, and expected, to take an active role in their own learning. Where Course activities and materials allow them to progress in an independent manner, this will allow teaching of the two groups to happen most effectively.

Useful resources
Online resources (websites, microsites, wikis, newsfeeds, databases, etc) can provide a valuable source of easily accessible and up-to-date information on a wide range of software design and development topics. In addition, the internet can act as a rich source of information for research into current trends in software development languages, environments, intelligent and online systems, etc, and their potential impact.

Although not a definitive list, the following resources may support the delivery of the Software Design and Development (Higher) Unit.

Some suggested general online resources:
- Technology Student
- Teaching Education Scotland Education Scotland
- STEM Central on Education Scotland website
- Khan Academy
- CompEdNet — Scottish Computing Science Teachers' Forum
- Computing at School Scotland

Some suggested specific online resources:
- Education Scotland Consolarium — there are a number of resources available here for games-based learning, programming and software development activities
- Scratch — Scratch on the MIT website — within this site you can also search for Scratch projects by going to the featured projects area of the Channel section
- BYOB (Build Your Own Blocks) website
- Java website
- MIT Centre for Mobile Learning website
- App Inventor software and tutorials
- Visual Basic 5 Control Creation Edition — search for tutorials on Google sites
- Dynamic Learning website
- How Stuff Works website
- Raspberry Pi website

Approaches to delivering and assessing each Outcome
The learner must demonstrate attainment of both of the Outcomes and their associated Assessment Standards. Assessment must be valid, reliable and fit for purpose.

SQA does not specify the methods of assessment to be used; teachers should determine the most appropriate method for their learners. In many cases, evidence will be gathered during normal classroom activities, rather than through formal assessment instruments.

Centres are expected to maintain a detailed record of evidence, including oral or observational evidence. Evidence in written or presentation format should be retained by the centre.

All evidence should be gathered under supervised conditions.

In order to ensure that the learner's work is their own, the following strategies are recommended:

- personal interviews with learners where teachers can ask additional questions about the completed work
- asking learners to do an oral presentation on their work
- ensuring learners are clear about acknowledging sources
- using checklists to record the authentication activity

Preparing learners for their assessment activities
In order to ensure that learners are prepared in advance for their assessment activities, it is good practice for teachers to make learners aware of the Assessment Standards required and to provide a range of feedback designed to improve learners' knowledge and skills as they progress through the Unit.

It is accepted as good practice that the evidence to meet the Assessment Standards for Outcome 1 and Outcome 2 is generated throughout the Unit as an integral part of classroom activities.

Outcome 1

1 Explain how programs work, drawing on an understanding of advanced concepts in software development and computer architecture, by:

1.1 Reading and explaining code
1.2 Describing the purpose of a range of programming constructs and how they work
1.3 Describing how a range of standard algorithms work
1.4 Describing how programs relate to low-level structures and operations

The range of programming constructs should include sub-programs, parameters, user-defined functions and sequential file operations. The range of standard algorithms should include input validation, linear search, finding minimum and maximum and counting occurrences.

Notes on delivery of Outcome 1
In order to meet Outcome 1, learners are expected to develop their knowledge and understanding of how a number of different programs work to the point where they have the ability to read and explain code, describe the purpose of a range of programming constructs and how they work, and are able to describe how a range of standard algorithms work. They will also need to know how these programs relate to low-level structures and operations.

Teachers can provide some background information relating to aspects of the software to be investigated. Opportunities for learning and teaching activities might include the following.

- Learners could be provided with working programs in several software development environments in order to learn the purpose of a range of programming constructs and how they work within the program. Teachers would explain these constructs and demonstrate how they work inside a program. This would develop the knowledge and understanding required in the learner to be able to read and explain sections of code.
- Learners could be provided with or asked to locate working programs that demonstrate sub-programs, user-defined functions, parameter passing, sequential file operations and scope, local and global variables from any software development environment. They could then be asked to identify and explain sections of code from within these programs.
- A similar exercise could be carried out with standard algorithms, whereby learners are provided with algorithms that exemplify input validation, linear search, finding minimum and maximum values and counting occurrences. Teachers could then demonstrate how these algorithms work by explaining each step in the sequence, which would give learners a sound understanding on how these algorithms actually work in practice.
- Within the programs provided to the learners, they will have to understand how data in the form of characters, integers and real numbers is stored and represented. Learners could be provided with programs that contain these data types and then shown how they are represented in binary. For example, positive integers represented using a set number of places; negative integers represented using the signed bit or two's complement; real numbers represented using floating point representation and characters represented using ASCII.
- When learners are creating and running their programs they will realise that their high-level code is being translated into machine code so that the computer can understand the instructions it has to perform. It would be useful to provide the learners with some examples and demonstrations of how this is carried out using an interpreter and a compiler. This would highlight the differences between these two types of translators and help the learners understand how coding and syntax errors are generated in their own programs.

Notes on assessment of Outcome 1
Evidence of the Assessment Standards for Outcome 1 may be derived from a single, extended software development task, or from a number of shorter tasks. Evidence can be generated throughout the Unit as an integral part of classroom activities.

- Learners can decide on the type of software programs to investigate.
- Learners can have access to books, the internet, pre-written code and other materials during the assessment.
- Learners should receive accurate and regular feedback from teachers and be actively involved in the assessment process.

Learners will use understanding of advanced concepts in software development to explain how programs work. Written or oral evidence might take the form of responses to a series of assignments, or a short test. Teachers will select appropriate evidence from the learner that exemplifies:

1.1 Evidence of ability to read and explain advanced program code
1.2 Detailed description of the purpose of a range of high-level programming language constructs and how they work
1.3 Detailed technical description of a range of standard algorithms including input validation, linear search, finding minimum and maximum and count occurrences
1.4 Detailed technical description of how programs relate to low-level structures and operations, describing the role of translator, processor and memory in the execution of high-level language instructions

The learner would be expected to produce an accurate series of descriptions in the form of extended responses (written or oral).

Evidence of the Assessment Standards for Outcome 1 may be oral or written. Evidence in written form should be retained by the centre. Where learners' responses have been in oral form, centres should keep a recording of their performance as evidence and/or an observation checklist.

All evidence should be gathered in supervised conditions.

Outcome 2

2 Develop modular programs using one or more software development environments by:

2.1 Applying contemporary design and development methodologies
2.2 Selecting and using combinations of appropriate constructs
2.3 Selecting and using appropriate simple and structured data types, including 1-D arrays
2.4 Testing digital solutions systematically
2.5 Applying aspects of good programming technique — meaningful variable names, internal commentary, indentation

The range of programming constructs should include sub-programs, parameters and sequential file operations. Programs should include at least two data types, including 1-D arrays.

Notes on delivery of Outcome 2
It is envisaged that learners will develop a number of different modular programs; these can be drawn from different software development languages and/or environments. The choice is entirely at the discretion of the centre and should be based on the suitability of the chosen environment to support the delivery of the mandatory content of the Unit. Evidence can be gathered from any of these throughout the duration of the Unit.

Below is a non-restrictive list of possible examples of software development environments which might be suitable.

Non-restrictive examples of current software development environments
Graphical environments in which code is assembled by combining graphical objects which represent instructions, variables and constructs
Software development environments which are specifically suited to games development such as C++, Gamemaker, Greenfoot and Dark Basic
Apps development environments which are suited to producing applications for handheld devices and smartphones, such as Java development tools, Apple SDK, Android SDK and Livecode
Integrated development environments, such as Eclipse
Text-based programming environments, such as Visual Basic, Truebasic and Python

Opportunities for a mixed variety of learning and teaching activities might include:

- exploring different contemporary design and development methodologies
- learning the main constructs and data types of a second or additional software development environment or language by working through a supported self-study resource or online tutorial
- learning the main constructs and data types of a particular software development environment or language by working through teacher-led instruction
- producing coded solutions to a number of programming tasks that combine certain constructs and data types
- systematically testing each module or sub-program once it has been completed before moving on to the next one
- constructing a test plan for the completed solution

Extending problem-solving opportunities
A useful strategy to extend problem-solving opportunities is to set tasks in a progression of unfamiliar contexts. For example, learners could experiment with less familiar design and development methodologies, such as wire-framing and the Agile method and apply these to their own program developments. They might also try to find solutions to a programming task in more than one software development environment with which they are not familiar, eg Greenfoot.

Testing digital solutions
Learners should develop skills in constructing their own test plan and complete comprehensive testing

Using their test data and any error reporting features of their development environment, learners should develop the skills required to identify and rectify errors in programs, such as syntax, logic and execution errors.

Notes on assessment of Outcome 2
Evidence for Outcome 2 can be generated throughout the Unit as an integral part of classroom activities, and may be derived from a single, extended software development task or from a number of shorter tasks. Formal documentation is not expected or required.

- Learners can decide which type of software development environments in which to develop their modular programs.
- Learners can have access to books, the internet, pre-written code and other materials during the assessment.
- Learners should receive accurate and regular feedback from teachers and lecturers and be actively involved in the assessment process.

Teachers will select appropriate evidence from the learner, including:

2.1 Evidence of applying contemporary design and development methodologies in high-level planning, showing main modules and data flow when developing modular programs
2.2 Selecting and using combinations of appropriate constructs as appropriate to programs
2.3 Evidence of using a minimum of one simple and one structured data type, including 1-D arrays in modular programs
2.4 Evidence of constructing their own test plan and systematically testing programs against it
2.5 Evidence of providing aspects of good programming techniques consistently — meaningful variable names, internal commentary and indentation

Developing skills for learning, skills for life and skills for work

Learners are expected to develop broad generic skills as an integral part of their learning experience.

The *Unit Specification* lists the skills for learning, skills for life and skills for work that learners should develop through this Unit. These are based on SQA's *Skills Framework: Skills for Learning, Skills for Life and Skills for Work.* The *Software Design and Development* (Higher) Unit holds opportunities to acquire and develop a number of these which will arise for the most part as a natural part of the learning and teaching process. The level of these skills will be appropriate to the level of the Unit.

The table below highlights opportunities to develop these skills during this Unit.

2 Numeracy	
2.1 Number processing	Learners can be given opportunities to develop their number processing skills by gaining practice in problem solving in numerically-based contexts which involve, for example, multiplication, division or calculating percentages. Problem-solving contexts could then be set in which software would take decisions and vary the output based on the results of calculations.
2.3 Information handling	Information handling skills could be developed by setting problem-solving contexts in which learners are required to use data set out in tables or a graphical format as the basis for input to their programs which then process the data to produce required output.

4 Employability, enterprise and citizenship	
4.2 Information and communication technology	Throughout the Unit, learners will be continuously interacting with the technology around them and will be given abundant opportunities to extend their ICT skills. When producing their report on contemporary computing technologies, they will be given opportunities to specifically develop their ICT-based research, evaluation and decision-making skills.

5 Thinking skills	
5.3 Applying	Learners will be given ample opportunities to analyse a wide range of problems, apply the knowledge and skills they have acquired and then test and review their solutions.
5.4 Analysing and evaluating	Learners will develop skills in analysing and evaluating through the process of creating computer programs to solve problems and evaluate through testing.

Combining assessment within the Unit

Holistic assessments could be developed which cover some of the Assessment Standards from Outcomes 1 and 2.

For example: an assessment could be devised that asks the learner to create a modular program with a combination of appropriate constructs, at least two data types (one structured) and internal commentary. Once they have created the program, tested it, and rectified any errors, they could then be asked to describe the purpose of the constructs and how they work within the program. The teacher could then select a segment of code from the program and ask the learner to explain what the code actually does. This would cover the following Assessment Standards:

1.1 Reading and explaining code
1.2 Describing the purpose of a range of programming constructs and how they work
2.2 Selecting and using combinations of appropriate constructs
2.3 Selecting and using appropriate simple and structured data types, including 1-D arrays
2.4 Testing digital solutions systematically
2.5 Applying aspects of good programming technique — meaningful variable names, internal commentary, indentation

Equality and inclusion

The requirement to develop practical skills involving the use of equipment and tools may present challenges for learners with physical or visual impairment. In such cases, reasonable adjustments may be appropriate, including (for example) the use of adapted equipment or alternative assistive technologies. This is for both learners and centres to consider.

It is recognised that centres have their own duties under equality and other legislation and policy initiatives. The guidance given in these *Unit Support Notes* is designed to sit alongside these duties but is specific to the delivery and assessment of the Unit.

Alternative approaches to Unit assessment to take account of the specific needs of learners can be used. However, the centre must be satisfied that the integrity of the assessment is maintained and where the alternative approach to assessment will, in fact, generate the necessary evidence of achievement.

Centres will find more guidance on this in the assessment arrangements section of SQA's website: www.sqa.org.uk/sqa/14977.html.

Appendix 1: Reference documents

The following reference documents will provide useful information and background.

- Assessment Arrangements (for disabled learners and/or those with additional support needs) — various publications on SQA's website: www.sqa.org.uk/sqa//14977.html
- Pseudocode for Higher Computing Science Question Paper: Computing Science Pseudocode
- British Computer Society, Glossary of Computing and IT, 12th edition
- *Building the Curriculum 4: Skills for learning, skills for life and skills for work*
- *Building the Curriculum 5: A framework for assessment*
- Course Specifications
- Design Principles for National Courses
- *Guide to Assessment* (June 2008)
- *Principles and practice papers for curriculum areas*
- *Research Report 4 — Less is More: Good Practice in Reducing Assessment Time*
- Course*work Authenticity — a Guide for Teachers and Lecturers*
- *SCQF Handbook: User Guide (published 2009)* and SCQF level descriptors (reviewed during 2011 to 2012): www.sqa.org.uk/sqa/4595.html
- *SQA Skills Framework: Skills for Learning, Skills for Life and Skills for Work*
- SQA Guidelines on e-assessment for Schools
- SQA Guidelines on Online Assessment for Further Education
- SQA e-assessment web page: www.sqa.org.uk/sqa/5606.html

Administrative information

Published: May 2016 (version 2.1)

History of changes to Unit Support Notes

Version	Description of change	Authorised by	Date
1.1	Amendments to *'Approaches to learning, teaching and assessment'* section, including Outcomes 1 and 2, to reflect changes in the Unit Specification, with revision of associated guidance. Revision of some suggested resources.	Qualifications Development Manager	June 2014
2.0	Changes made throughout to reflect the removal of Outcome 3 and Assessment Standard 2.5 'Identifying and rectifying program errors' in Outcome 2.	Qualifications Manager	June 2015
2.1	Amendments have been made to information within the 'Approaches to learning, teaching and assessment' section, to include the addition of 'Testing'.	Qualifications Manager	May 2016

This document may be reproduced in whole or in part for educational purposes provided that no profit is derived from reproduction and that, if reproduced in part, the source is acknowledged. Additional copies can be downloaded from SQA's website at **www.sqa.org.uk.**

Note: You are advised to check SQA's website (**www.sqa.org.uk)** to ensure you are using the most up-to-date version.

© Scottish Qualifications Authority 2016

Unit Support Notes

Unit Support Notes — Information System Design and Development (Higher)

This document may be reproduced in whole or in part for educational purposes provided that no profit is derived from reproduction and that, if reproduced in part, the source is acknowledged. Additional copies of these *Unit Support Notes* can be downloaded from SQA's website: **www.sqa.org.uk**.

Please refer to the note of changes at the end of this document for details of changes from previous version (where applicable).

Introduction

These support notes are not mandatory. They provide advice and guidance on approaches to delivering and assessing *the Information System Design and Development* (Higher) Unit. They are intended for teachers and lecturers who are delivering this Unit. They should be read in conjunction with:

- the Unit Specification
- the Course Specification
- the Course Assessment Specification
- the Course Support Notes
- appropriate assessment support materials

General guidance on the Unit

Aims

The purpose of this Unit, as stated in the *Unit Specification*, is for the learner to develop knowledge and understanding of advanced concepts and practical problem-solving skills related to the design and development of information systems through a range of practical and investigative tasks. Learners will apply their computational thinking skills to implement practical solutions using a range of development tools and to develop an understanding of the technical, legal, environmental, economic and social issues related to one or more information systems.

This Unit will give learners the opportunity to develop their thinking skills and skills in numeracy, employability, enterprise and citizenship.

This Unit can be delivered:

- as a stand-alone Unit
- as part of the Higher Computing Science Course

Progression into this Unit

Entry to this Unit is at the discretion of the centre. However, learners would normally be expected to have attained some useful skills and knowledge from prior learning, such as:

- National 5 Computing Science Course or relevant component Units

Learners may also have gained relevant skills and knowledge through other education systems or from their own interests and informal learning.

Skills, knowledge and understanding covered in this Unit

Information about skills, knowledge and understanding is given in the Higher Computing Science *Course Support Notes.*

If the Unit is being delivered as part of the Higher Computing Science Course, the teacher should refer to the 'Further mandatory information on Course coverage' section within the *Course Assessment Specification* for detailed content.

If this Unit is being delivered on a free-standing basis, teachers and lecturers are free to select the skills, knowledge, understanding and contexts which are most appropriate for delivery in their centres.

Progression from this Unit

On successful completion of this Unit, the following Units and Courses provide appropriate progression pathways for learners:

- related Units in Advanced Higher Computing Science Course
- National Certificate Group Awards in a range of Computing, IT and related disciplines
- National Progression Awards in Digital Media
- employment, apprenticeships and/or training in Computing, IT and related fields

Approaches to learning, teaching and assessment

Learning and teaching
The Unit is designed to provide flexibility, personalisation and choice for both the learner and the teacher.

The Higher Computing Science *Course Support Notes* provide further broad guidance on approaches to learning and teaching which may apply to all of the component Units of the Course and should be read before delivering this Unit.

Learning and teaching activities should be designed to stimulate learners' interest, and to develop skills and knowledge to the standard required by the Outcomes and to the level defined by the associated Assessment Standards. Learning should be supported by appropriate practical activities so that skills are developed simultaneously with knowledge and understanding.

An investigatory approach is encouraged, with learners actively involved in developing their skills, knowledge and understanding by investigating a range of real-life and relevant information systems, problems and solutions.

The *Unit Specification* defines the skills and knowledge required, but leaves complete freedom to the teacher and learner to select interesting contexts and environments in which to develop these. Tasks and activities throughout the Unit should be linked to relevant contexts and real-world applications, where appropriate. This provides scope for personalisation and choice, as relevant and motivating environments can be used.

Practical activities and investigations lend themselves readily to group work, and this should be encouraged. Individual, paired or group problem-solving tasks could be related to authentic and relevant contexts. For example, learners could be asked to:

- create an interactive website that appeals to the viewer with appropriate use of text, graphics, sound and video similar to the glossy magazine websites
- work in pairs or groups to research the different types of user interface for information systems aimed at different age groups and levels of ability
- work in pairs or groups to compare the hardware and software specifications of particular types of information system, eg office information systems, decision support systems, transaction processing systems, management reporting systems
- work in pairs or groups to research the impact of cloud computing on certain businesses and companies
- work in pairs or groups to research the environmental, economic and societal impact of different information systems

When delivering the Unit as part of the Higher Computing Science Course, reference should be made to the appropriate content statements within the 'Further mandatory information on Course coverage' section of the *Course Assessment Specification* to ensure the required breadth of knowledge is covered.

Sequence of delivery of Outcomes
The sequence of delivery of the Outcomes is a matter of professional judgement and is entirely at the discretion of the centre. Some suggested approaches are listed below.

Outcome 1 and Outcome 2 simultaneously
Teachers may consider it good practice to combine the development of knowledge and understanding of the factors to consider in the design and development of an information system at the same time as learners progress through the practical tasks involved in creating information systems.

For example, when learners are designing their information system, whether relational databases, dynamic or interactive websites, and/or other information systems, they will have to consider the functionality of the product they are designing as well as the range and type of users. The importance of a well-designed user interface is necessary depending on the range and type of users. Learners may experiment with different types of software applications to create their products and they should be made aware of the hardware and software requirements for these different types of applications. Although learners will probably use software that is available to them on their centre's network, they should also have an idea of the type of similar applications that are available from cloud systems. File sizes and types of storage media should be considered when learners are storing their products on the network, which highlights the importance of storage and capacity for any information system.

Outcome 1 before Outcome 2
It is possible to develop the learners' practical skills in creating information systems and meet the requirements of Outcome 1 before moving to formally develop the knowledge and understanding associated with the factors involved in the design and development of an information system in Outcome 2.

For example, learners could develop the practical skills associated with creating information systems by working with relational databases, websites, and/or other information systems. This would allow them to become familiar with the features of the different software applications used to build these systems; how to create an appropriate user interface; how to integrate different media types; how to write code in the form of scripts to customise the system; how to constantly test their systems against set criteria; and how to identify and rectify errors and bugs that occur during testing.

This type of process would give them a solid understanding of the main factors involved in developing an information system and the practical skills required to actually create it. Learners should then be able to relate this experience to the factors involved in the design and implementation of a real-life information system. By investigating the functionality, the technical specifications, the security risks and the legal, economic and social impact of a real-life information system, learners should be able to relate the factors and concepts to the smaller-scale information systems that they have created.

Learners should be encouraged to make connections between the processes involved in designing and developing their own information systems and the factors that have to be considered in the design and development of more complex large-scale information systems used in the real world.

Outcome 2 before Outcome 1
It is also possible to consider the factors involved in the design and development of an information system and meet the requirements of Outcome 2 before actually carrying out the practical task of creating information systems in Outcome 1.

For example, learners could develop an understanding of the main factors involved in the design and development of a real-life information system before developing their own. They could select a particular information system that is used in a local company or business and then find out:

- What are the main functions of the system?
- Who are the main users and what does each require from the system?
- What is the technical specification of the system in terms of hardware and software?
- What type of storage does it use?
- What is the level of connectivity?
- What are the main security risks with the system?
- What precautions or safeguards are in place to ensure there are no breaches of security for the operator and the end users?
- Are there any legal implications associated with the design, implementation or use of the system?
- Are there any economic or societal implications associated with the system?

With this overview in mind, learners could then proceed to create their own information systems, such as relational databases, dynamic websites, and/or other contemporary information system types.

By doing Outcome 2 first, learners should be able to relate the main issues and factors they discover when researching a 'real-life' information system to their own products, albeit on a much smaller scale. Learners could then develop their practical skills by exploring the functionality of the software they are using to create a structure with links and integrating different media types. They might then learn about compression techniques and their effects on file size. This could be linked to the importance of storage requirements in real-life information systems. Learners might also customise parts of the system through scripting or writing code in order to achieve a specific Outcome, and there may be examples from the real-life system where the client has requested certain customisations or amendments to meet a particular requirement for the business.

Meeting the needs of all learners
When delivering this Unit to a class, with some learners working towards National 5 and others towards Higher, it may be useful for teachers to identify activities covering common knowledge and skills for all learners, and additional activities for Higher learners. For example:

Structures and links
Both National 5 and Higher learners will be creating information systems, such as relational databases, websites, and/or other types of information systems, but the Higher learners will have different specifications. Both sets of learners can decide on the type of information systems they are going to create and the types of software they are going to use to build the systems. Teacher-directed lessons or self-supported materials could be used to guide the learner through the main

functions of the software, with extension exercises built in for the Higher learners dealing with aspects, such as cascading style sheets, dynamic web pages, interactive web pages and multi-level navigation.

User interface
Core lessons in the design of a user interface could be delivered to both sets of learners, with the Higher learners studying additional criteria for good user interface, such as usability and accessibility.

Media types
When looking at different media types to be incorporated into their information systems, both sets of learners could be made aware of the properties of standard file formats, factors affecting file size and quality and the need for compression. The Higher learners might then go on to look at compression techniques in more detail, eg lossy versus lossless compression.

Coding
Both sets of learners should learn the basic constructs of a scripting language, such as JavaScript, with the Higher learners moving on to looking at how these scripts are used in configuring client- and server-side scripting. While the National 5 learners are learning HTML, the Higher learners could be learning to script cascading style sheets (CSS) and studying their effect on the appearance of web pages.

Testing
Both sets of learners should be taught the importance of testing in the development and implementation of information systems but National 5 learners would just focus on links and navigation whereas the Higher learners should test usability and compatibility.

Technical implementation
National 5 and Higher learners should both study the technical specification of real-life information systems covering the same aspects in terms of processor type, speed, memory, operating system, etc, but the Higher learners should be given an extension exercise in looking at additional features of software, such as licensing, transferability and proprietary versus open source. A similar exercise might take place on the topic of storage, with the Higher learners looking at additional aspects, such as backup systems and strategy, and distributed and offline storage.

Technical implementation/networking and connectivity
On the topic of networking and connectivity, National 5 learners might compare peer-to-peer networks with client/server networks and study different types of transmission media. The introduction of the topic of cloud computing could then be taught to both sets of learners but when the National 5 learners are comparing a cloud system with a local area network, the Higher group might look at some of the main features of cloud services, such as SaaS, IaaS, and PaaS.

Security
Both sets of learners should study security precautions but when the National 5 learners are looking at some of the standard precautions, such as encryption and biometrics, the Higher group could be looking at digital certificates and signatures.

Different contexts
Where Higher learners have studied National 5 in a previous year, it is important to provide them with new and different contexts for learning to avoid demotivation. It is particularly important that learners do not feel that they are simply doing the same work over again, albeit at a deeper level.

For example, at National 5, learners might have built databases, websites, and/or other information system types. At Higher they will be able to add more functionality to one or more products or build new systems using the techniques and skills they have acquired throughout the Higher Course. For example, they will be able to apply style sheets to their web pages, make their web pages dynamic and interactive and introduce multi-level navigation.

Higher learners should consider a number of different aspects of user interface design compared to those studied at National 5, and when studying media types Higher learners should focus on compression techniques applied to those media types rather than their properties.

At National 5, learners will have been introduced to scripting and a mark-up language. At Higher they could continue to develop their coding skills in scripting but in the context of database, client-side and server-side scripting. Higher learners might also write code to insert into cascading style sheets and gain an understanding of the effect these have on web pages for example.

In terms of the technical implementation of information systems (hardware, software, storage, networking and connectivity), Higher learners could develop their knowledge and understanding of these aspects through new contexts, such as software licensing, differences between proprietary and open source software, distributed and offline storage and cloud computing. They could also look at the economic and social implications of information systems which they might not have covered before.

Computational thinking
Aspects of computational thinking — abstraction, algorithms, decomposition, pattern recognition and generalisation — are relevant to the design and development of information systems, just as they are in programming, and so should be exemplified within this Unit where appropriate.

In line with the underlying principles of Curriculum for Excellence, learners should be encouraged, and expected, to take an active role in their own learning. Where Course activities and materials allow them to progress in an independent manner, this will allow teaching of the two groups to happen most effectively.

Useful resources
Online resources (websites, microsites, wikis, newsfeeds, databases, etc) can provide a valuable source of easily accessible and up-to-date information on a wide range of information systems design and development topics. In addition, the internet can act as a rich source of information for research into the legal implications and environmental implications involved in the design and implementation of an information system, as well as its economic and social impact.

Although not a definitive list, the following resources may support the delivery of the Information System Design and Development (Higher) Unit.

Some suggested general online resources:

- Technology Student
- Teaching Education Scotland
- Education Scotland
- STEM Central on Education Scotland website
- Khan Academy
- CompEdNet — Scottish Computing Science Teachers' Forum
- Computing at School Scotland

Specific online resources:

- Education Scotland Consolarium — there are a number of resources available here for games-based learning, programming and software development activities
- Scratch — Scratch on the MIT website
- BYOB (Build Your Own Blocks) website
- Java website
- Webmonkey website
- MIT Centre for Mobile Learning website
- App Inventor software and tutorials
- Visual Basic 5 Control Creation Edition — search for tutorials on Google sites

Approaches to delivering and assessing each Outcome

The learner must demonstrate attainment of all of the Outcomes and their associated Assessment Standards. Assessment must be valid, reliable and fit for purpose.

SQA does not specify the methods of assessment to be used; teachers should determine the most appropriate method for their learners. In many cases, evidence will be gathered during normal classroom activities, rather than through formal assessment instruments. Centres are expected to maintain a detailed record of evidence, including oral or observational evidence. Evidence in written or presentation format should be retained by the centre.

All evidence should be gathered under supervised conditions.

In order to ensure that the learner's work is their own, the following strategies are recommended:

- personal interviews with learners where teachers can ask additional questions about the completed work
- asking learners to do an oral presentation on their work
- ensuring learners are clear about acknowledging sources
- using checklists to record the authentication activity

Preparing learners for their assessment activities

In order to ensure that learners are prepared in advance for their assessment activities, it is good practice for teachers to make learners aware of the Assessment Standards required and to provide a range of feedback designed to improve learners' knowledge and skills as they progress through the Unit.

It is accepted as good practice that the evidence to meet the Assessment Standards for Outcome 1 and Outcome 2 be generated throughout the Course as an integral part of classroom activities.

Outcome 1

1 Develop information systems using appropriate development tools, by:

1.1 Applying contemporary design and development methodologies
1.2 Creating a complex structure with links
1.3 Writing code
1.4 Integrating different media types
1.5 Testing against appropriate criteria

Notes on delivery of Outcome 1
It is envisaged that learners will develop a number of different information systems which might include relational databases, websites and/or other contemporary information systems. Developing a range of these would allow the learner to develop the broad knowledge and understanding required for Course assessment.

Tasks and activities throughout the Unit should be linked to relevant contexts, for example:

- using contemporary design and development methodologies such as prototyping and rapid application development
- using JavaScript for client-side scripting
- creating dynamic web pages and database-driven websites using PHP or MYSQL

Learners should be introduced to a number of different types of design and development methodologies, eg prototyping and rapid application development. They would then need to consider the main factors of these methodologies in order to decide which one to use in the design of their own information systems. They could work in pairs or groups to consider and discuss the main aspects of these methodologies before deciding which one they will use.

A similar task could be carried out when considering the principles of good user interface design. Discussion in pairs or groups could take place before learners decide on the design of the user interface they are going to implement.

Learners could use different tools to demonstrate high-level planning of their information system such as storyboarding, wire-framing, modelling or a similar technique to design the structure of each page or section of an information system or of a relational database. They should be given instruction or resources to enable them to use software applications to incorporate and integrate different media types into information systems.

Learners might also need to further develop their skills and have support in coding using a scripting language if the activity is the creation of dynamic web content. Practical tutorials or teacher-led instruction may be required to teach the basics of the scripting language, followed by a series of tasks or exercises to reinforce learners' understanding. Learners should then be able to demonstrate their skills by customising part of their information systems to do a particular task or meet a particular requirement.

Learners should be encouraged to identify any errors and seek solutions during the development of their information systems. Teachers and lecturers should give support to learners in helping them identify errors and look for ways to rectify any problems with the information systems.

Learners should be encouraged to test their information systems against appropriate set criteria. Test plans should be drawn up and all results should be documented to provide evidence of the testing process.

Notes on assessment of Outcome 1
Evidence is only required of one successful example for each Assessment Standard and can be generated throughout the Unit as an integral part of classroom activities, using a single problem-solving task or by a series of separate activities.

- Learners can decide which type of information systems to develop, eg databases, a series of web pages, multimedia products, web-based applications or other current information systems.
- Learners can have access to books, the internet and other materials during assessment activities.
- Learners should receive accurate and regular feedback from teachers and lecturers and be actively involved in the assessment process.

NB: Please note that although learners are required for Course assessment to understand the concept of a relational database, there is no requirement within this Unit for them to carry out a full normalisation process.

Teachers will select the appropriate evidence from the learner. Evidence can be observational; however, centres should keep a record, either in digital format or as an observational checklist, recording, video, etc, for verification purposes.

All evidence should be gathered in supervised conditions.

Outcome 2

2 Consider the factors involved in the design and implementation of an information system by describing in detail its:

2.1 Functionality, range and types of users
2.2 Technical implementation (hardware and software requirements)
2.3 Technical implementation (storage and connectivity)
2.4 Security risks and precautions
2.5 Legal and environmental implications
2.6 Economic and social impact

Notes on delivery of Outcome 2
It is envisaged that learners will investigate a complex real-life information system (this might be an office information system, a decision support system, a transaction processing system, management reporting system, interactive website or other current information system). By focusing on the factors that should be considered in the design and development of this information system, they should meet the Assessment Standards for Outcome 2.

Teachers and lecturers can provide some background information relating to aspects of the information system to be investigated.
Learners have to demonstrate that they have an understanding of the underlying technical concepts of this information system, including hardware, software, storage and connectivity requirements.

Opportunities for different learning and teaching activities might include:

- comparing different devices, such as desktops, laptops, tablet PCs and smartphones in terms of processor type, number and processor speed, memory (RAM, ROM, cache) and available operating systems
- examining the different licensing options available with certain software
- using online research, such as websites for Dell, PC World, etc, to compare the range of different storage devices available
- researching the concept and use of cloud computing services by investigating aspects, such as Software as a Service (SaaS), Infrastructure as a Service (IaaS) and Platform as a Service (PaaS)
- researching the news online to look for instances when public bodies have used the Regulation of Investigatory Powers Act (2000) or the Computer Misuse Act
- using the internet to find examples of censorship in eg China and comparing this with other examples of freedom of speech from other countries

Outcome 2 lends itself to more paired and/or group work. Aspects of existing information systems solutions to real-world problems can be analysed in simple terms to aid understanding, and a number of different case studies or types of 'real-life' information systems should be provided to the learners. The tasks and activities here will be more of the research and investigation type, with learners working in pairs or groups.

For example, each group could be asked to analyse the features of the information system in their centre to obtain an overview of the many tasks that it performs, eg:

- student attendance
- student profiles
- tracking and monitoring attainment
- reports
- timetables
- behaviour referral system
- assessment results

Alternatively, each group will be given an information system to investigate and report on, under the following headings:

- functionality, range and types of users (both human and other software)
- detailed hardware and software requirements
- storage and connectivity
- security risks and precautions
- legal, environmental, economic and social impact

Notes on assessment of Outcome 2
Evidence for Outcome 2 can be generated throughout the Unit as an integral part of classroom activities.

- Learners can decide on the information system to investigate.
- Learners can have access to books, the internet and other materials during the assessment.
- Learners should receive accurate and regular feedback from teachers and lecturers and be actively involved in the assessment process.

The learner would be expected to produce an accurate series of descriptions in the form of extended responses. However, the evidence for Outcome 2 need not be written, but may be presented in another format, such as a written document, a series of web pages, a visual presentation, a video, a podcast, a blog or any other appropriate format.

Group work approaches can be used within Units (and across Courses) where it is helpful to simulate real-life situations, share tasks and promote team working skills. However, there must be clear evidence for **each** learner to show that the learner has me the required Assessment Standards for the Unit or Course.

Teachers and lecturers will select the appropriate evidence from the learner on the following aspects of the information system:

2.1 Detailed statement describing the functionality of the information system and the range and types of users it has been designed for
2.2 Detailed technical description of the hardware and software used and how they meet the requirements of an information system
2.3 Detailed technical description of the type and size of storage required and what connectivity would be suitable to meet the requirements of an information system
2.4 Detailed description of one security risk and two related security precautions affecting the design and implementation of an information system
2.5 Detailed description of one legal and one environmental implication affecting the design and implementation of an information
2.6 Detailed description of one economic and one social impact of an information system

Where a learner's responses have been in oral form, centres should keep either a recording of a learner's performance as evidence and/or an observation checklist.

All evidence should be gathered in supervised conditions.

Developing skills for learning, skills for life and skills for work

Learners are expected to develop broad generic skills as an integral part of their learning experience. The *Unit Specification* lists the skills for learning, skills for life and skills for work that learners should develop through this Course. These are based on SQA's *Skills Framework: Skills for Learning, Skills for Life and Skills for Work* and must be built into the Unit where there are appropriate opportunities. The level of these skills will be appropriate to the level of the Unit.

The table below highlights opportunities to develop these skills during this Unit.

2 Numeracy	
2.1 Number processing	Learners can be given opportunities to develop their number processing skills by gaining practice in problem solving in numerically-based contexts, eg calculation of storage requirements in terms of capacity, speed, compression of different media types and distribution, and then making informed decisions based on the results of these calculations.
2.3 Information handling	Information handling skills could be developed by setting problem-solving contexts in which learners are required to model data structures, layout web pages, create multi-tier navigational structures, compare cloud and other storage system requirements and look at security risks and precautions.

4 Employability, enterprise and citizenship	
4.2 Information and communication technology	Throughout the Unit, learners will be continuously interacting with the technology around them and will be given abundant opportunities to extend their ICT skills. When researching an information system related to Outcome 2, they will be given opportunities to specifically develop their ICT-based research, evaluation and decision-making skills.

5 Thinking skills	
5.3 Applying	Learners will be given ample opportunities to analyse a wide range of problems, apply the knowledge and skills they have acquired and then test and review their solutions.
5.4 Analysing and evaluating	Learners will develop skills in analysing and evaluating through the process of researching and completing the investigations for Outcome 2.

Combining assessment within Units

It may be possible to develop learning/assessment activities which provide evidence that learners have achieved the standards for more than one Outcome within the Unit, thereby reducing the assessment burden on learners. Combining assessment of Outcomes (or parts of Outcomes) in this way is perfectly acceptable, but needs to be carefully managed to ensure that all Assessment Standards and Outcomes for the Unit are covered.

Equality and inclusion

The requirement to develop practical skills involving the use of equipment and tools may present challenges for learners with physical or visual impairment. In such cases, reasonable adjustments may be appropriate, including (for example) the use of adapted equipment or alternative assistive technologies. This is for both learners and centres to consider.

It is recognised that centres have their own duties under equality and other legislation and policy initiatives. The guidance given in these *Unit Support Notes* is designed to sit alongside these duties but is specific to the delivery and assessment of the Unit.

Alternative approaches to Unit assessment to take account of the specific needs of learners can be used. However, the centre must be satisfied that the integrity of the assessment is maintained and that the alternative approaches to assessment will, in fact, generate the necessary evidence of achievement.

Centres will find more guidance on this in the assessment arrangements section of SQA's website: www.sqa.org.uk/sqa/14977.html.

Appendix 1: Reference documents

The following reference documents will provide useful information and background.

- Assessment Arrangements (for disabled learners and/or those with additional support needs) — various publications on SQA's website: www.sqa.org.uk/sqa//14977.html
- Pseudocode for Higher Computing Science Question Paper: Computing Science Pseudocode
- British Computer Society, Glossary of Computing and IT, 12th edition
- *Building the Curriculum 4: Skills for learning, skills for life and skills for work*
- *Building the Curriculum 5: A framework for assessment*
- Course Specifications
- Design Principles for National Courses
- *Guide to Assessment* (June 2008)
- *Principles and practice papers for curriculum areas*
- *Research Report 4 — Less is More: Good Practice in Reducing Assessment Time*
- Course*work Authenticity — a Guide for Teachers and Lecturers*
- *SCQF Handbook: User Guide (published 2009)* and SCQF level descriptors (reviewed during 2011 to 2012): www.sqa.org.uk/sqa/4595.html
- *SQA Skills Framework: Skills for Learning, Skills for Life and Skills for Work*
- SQA Guidelines on e-assessment for Schools
- SQA Guidelines on Online Assessment for Further Education
- SQA e-assessment web page: www.sqa.org.uk/sqa/5606.html

Administrative information

Published: May 2016 (version 2.0)

History of changes to Unit Support Notes

Version	Description of change	Authorised by	Date
1.1	Minor edits and clarification made to reflect changes to the content table in the Course Assessment Specification.	Qualifications Development Manager	June 2014
2.0	Changes made to reflect the removal of Assessment Standards 1.3 and 1.6 from Outcome 1.	Qualifications Manager	June 2015

This document may be reproduced in whole or in part for educational purposes provided that no profit is derived from reproduction and that, if reproduced in part, the source is acknowledged. Additional copies can be downloaded from SQA's website at **www.sqa.org.uk.**

Note: You are advised to check SQA's website (**www.sqa.org.uk)** to ensure you are using the most up-to-date version.

© Scottish Qualifications Authority 2016