Course/Unit Support Notes

Advanced Higher Computing Science Course/Unit Support Notes

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

May 2016, version 2.1

Contents

Introduction

These support notes are not mandatory. They provide advice and guidance on approaches to delivering and assessing the Advanced Higher Computing Science Course. Although primarily intended for teachers and lecturers who are delivering the Course and its Units, it may be useful to share some aspects with learners.

These support notes cover both the Advanced Higher Course and the Units in it.

The Advanced Higher Course/Unit Support Notes should be read in conjunction with the relevant:

Mandatory information:

- Course Specification
- Course Assessment Specification
- Unit Specifications

Assessment support:

- Specimen and Exemplar Question Papers and Marking Instructions
- Exemplar Question Paper Guidance
- Guidance on the use of past paper questions
- Coursework Information:
  - General assessment information
  - Coursework Assessment Task*
- Unit Assessment Support*

*These documents are for assessors and are confidential. Assessors may access these through the SQA Co-ordinator in their centres.

Related information
Advanced Higher Course Comparison

Further information on the Course/Units for Advanced Higher Computing Science
This information begins on page 13 and both teachers and learners may find it helpful.

Equality and inclusion

It is recognised that centres have their own duties under equality and other legislation and policy initiatives. The guidance given in these *Course/Unit 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.

The greater flexibility and choice in Advanced Higher Courses provide opportunities to meet a range of learners' needs and may remove the need for learners to have assessment arrangements. However, where a disabled learner needs reasonable adjustment/assessment arrangements to be made, you should refer to the guidance given in the above link.

General guidance on the Course/Units

Aims

The aims of the Course are to enable learners to:

- ♦ understand and apply computational thinking skills across a range of computing contexts
- ♦ extend and apply knowledge and understanding of advanced concepts and processes in computing science
- ♦ apply skills and knowledge in analysis, design, development, implementation and evaluation to a range of digital solutions with increasingly complex aspects
- ♦ apply creative problem-solving skills across a range of contexts
- ♦ develop autonomous learning, investigative and research skills
- ♦ communicate advanced computing concepts clearly and concisely, using appropriate terminology
- ♦ develop an informed understanding of the role and impact of computing technologies in transforming and influencing our environment and society

Progression

In order to do this Course, learners should have achieved the Higher Computing Science Course.

Learners who have achieved this Advanced Higher Course may progress to further study, employment and/or training. Opportunities for progression include:

- ♦ Progression to further/higher education
  - — For many learners a key transition point will be to further or higher education, for example to Higher National Certificates (HNCs)/Higher National Diplomas (HNDs) or degree programmes, for example, degree courses in various branches of computing science, information technology and other related areas.
  - — This Course provides a good preparation for learners progressing to further and higher education, as learners doing Advanced Higher Courses must be able to work with more independence and less supervision. This eases their transition to further/higher education. Advanced Higher Courses may also allow ‘advanced standing’ or partial credit towards the first year of study of a degree programme.
  - — Advanced Higher Courses are challenging and testing qualifications — learners who have achieved multiple Advanced Higher Courses are regarded as having a proven level of ability that attests to their readiness for higher education in HEIs in other parts of the UK as well as in Scotland.

— For many learners, progression will be directly to employment or work-based training programmes.

This Advanced Higher Computing Science could be part of the Scottish Baccalaureate in Science. The Scottish Baccalaureates in Expressive Arts, Languages, Science and Social Sciences consist of coherent groups of subjects at Higher and Advanced Higher level. Each award consists of two Advanced Highers, one Higher and an Interdisciplinary Project which adds breadth and value and helps learners to develop generic skills, attitudes and confidence that will help them make the transition into higher education or employment.

Hierarchies

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

This Course is designed in hierarchy with the corresponding Course at Higher and has a similar Course structure and Unit titles as the Higher Course.

Centres should be aware that although the mandatory knowledge and skillset may be similar across the hierarchical Units in Higher and Advanced Higher Courses, there may be differences in the:

- depth of underpinning knowledge and understanding
- complexity and sophistication of the applied skills
- way in which learners will learn: namely, they will take more responsibility for their learning at Advanced Higher and work more autonomously

A table showing the relationship between the mandatory Higher and Advanced Higher knowledge and understanding is included in this document. This table may be useful for:

- ensuring seamless progression between levels
- identifying important prior learning for learners at Advanced Higher

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

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.

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 development of subject-specific and generic skills is central to the Course. Learners should be made aware of the skills they are developing and of the transferability of them. It is the transferability that will help learners with further study and enhance their personal effectiveness.

The table below shows where there are likely to be opportunities to develop mandatory skills in or across the Units. However, the delivery mode adopted and the approaches to learning and teaching will determine how and where the opportunities arise.

Mandatory skills and knowledge	**Software Design and Development**	**Information System Design and Development**	**Course assessment**
applying computational thinking to solve complex computing problems	✓		✓
analysing complex problems within computing science, across a range of contemporary contexts	✓	✓	✓
developing and implementing well-structured, complex modular programs, and the ability to communicate how a program works	✓		✓
communicating understanding of complex concepts related to software design and development clearly and concisely, using appropriate terminology	✓		✓
designing, developing and implementing complex information systems		✓	✓
knowledge and understanding of contemporary software and information system project planning and management		✓	✓
knowledge and understanding of the role and impact of contemporary development on the environment and society			✓
knowledge and understanding of contemporary programming paradigms	✓		✓
ability to plan, design, implement, test, evaluate and report on a challenging computing science project			✓

Approaches to learning and teaching

Advanced Higher Courses place more demands on learners, as there will be a higher proportion of independent study and less direct supervision. Some of the approaches to learning and teaching suggested for other levels (in particular, Higher) may also apply at Advanced Higher level but there will be a stronger emphasis on independent learning.

For Advanced Higher Courses, a significant amount of learning may be self-directed and require learners to demonstrate a more mature approach to learning, and the ability to work on their own initiative. This can be very challenging for some learners, who may feel isolated at times, and teachers and lecturers should have strategies for addressing this. These could include, for example, planning time for regular feedback sessions/discussions on a one-to-one basis and on a group basis led by the teacher or lecturer (where appropriate).

Learners should be encouraged to use an enquiring, critical and problem-solving approach to their learning. Learners should also be given the opportunity to practise and develop research and investigation skills, and higher-order evaluation and analytical skills.

Learners will engage in a variety of learning activities as appropriate to the subject, for example:

- researching information for their subject, rather than receiving information from their teacher or lecturer
- using active and open-ended learning activities such as research, case studies and presentation tasks
- making use of the internet to assist research
- recording in a systematic way the results of research and independent investigation from different sources
- presenting findings/conclusions of research and investigation activities in a presentation
- participating in group work with peers and using collaborative learning opportunities to develop team working
- drawing conclusions from complex information
- using appropriate technological resources (eg web-based resources)
- participating in site visits

Teachers/lecturers should support learners by having regular discussions with them and giving regular feedback. Some learning and teaching activities may be carried out on a group basis and, where this applies, learners could also receive feedback from their peers.

Teachers/lecturers should, where possible, provide opportunities to personalise learning for learners, and to enable them to have choices in approaches to learning and teaching. The flexibility in Advanced Higher Courses and the independence with which learners carry out the work lend themselves to this. In particular, in this Course learners have opportunities to:

- research aspects of computing science of interest to them, in both the *Software Design and Development Unit* and *Information System Design and Development* Unit
- design, implement and test a challenging computing science project on a topic they have chosen for their Coursework project

Teachers/lecturers should also create opportunities for, and use, inclusive approaches to learning and teaching. This can be achieved by encouraging the use of a variety of learning and teaching strategies which suit the needs of all learners. Innovative and creative ways of using technology can also be valuable in creating inclusive learning and teaching approaches.

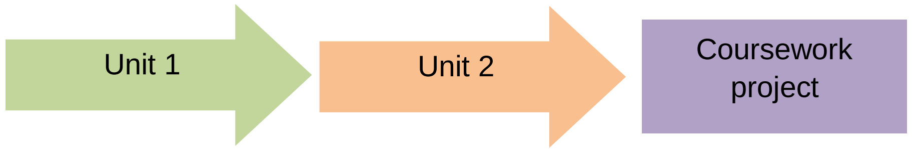
Approaches to structuring the Course

A centre is free to sequence the teaching of the Outcomes, Units and/or Course in any order it wants. For example:

- each Unit could be delivered separately in any sequence
- Units could be delivered concurrently
- Units and Coursework project could be delivered concurrently

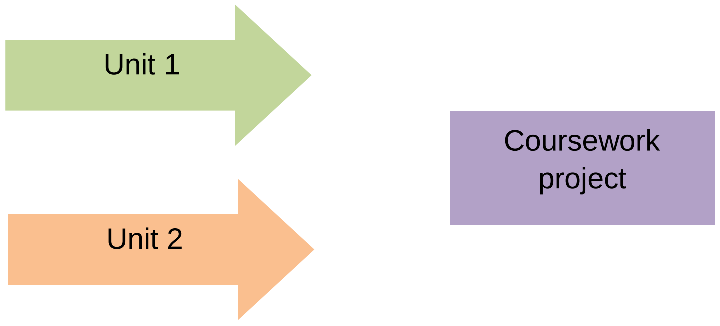
Example 1: Sequential delivery of the Units leading on to the Coursework project

In this case the learner would be completing each of the Units in turn, before embarking on the project. It would not matter in which order the Units were completed.



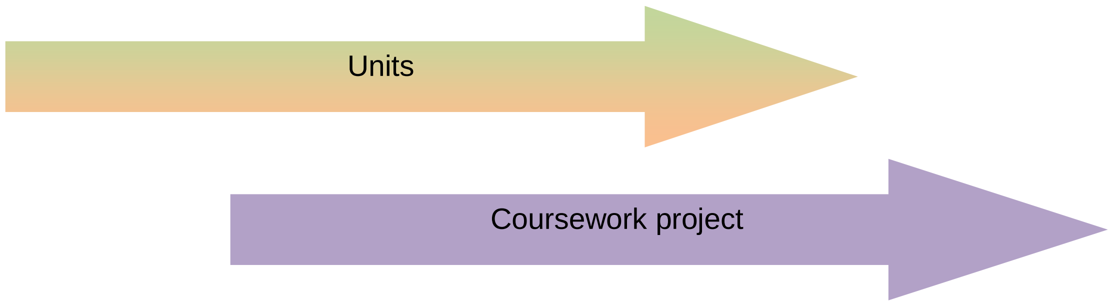
This approach would allow learners to focus on each aspect of the Course, ie information system design and development, and software design and development in turn and, dependent upon resources available, this may assist in resource and classroom management. Centres must ensure that sufficient time is left to undertake the project as this is an extended piece of work.

Example 2: Concurrent delivery of the Units leading on to the Coursework project
Using this approach, the learner would be completing Units concurrently before embarking on the project.



This approach would allow learners to work across the Unit learning themes. Again, as in example 1, centres must ensure that sufficient time is left to undertake the project.

Example 3: Concurrent delivery of the Units and Coursework project



Using this approach, learners would be able to progress to some aspects of the project which relate directly to learning from the Units. More information on this approach is given in the '*Further information on the Course/Units'* section of this document. This concurrent approach would allow a longer period over which the Coursework project could be tackled, whilst at the same time developing skills, knowledge and understanding from the Units.

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

Details of these skills can be found in the *Course Specification* and the *Unit Specifications* for this Course.

Approaches to assessment

Assessment in Advanced Higher Courses should reflect the investigative nature of the Course at this level, together with high-level problem-solving and critical thinking skills and skills of analysis and synthesis.

This emphasis on higher-order skills, together with the more independent learning approaches that learners will use, distinguishes the added value at Advanced Higher level from the added value at other levels.

There are different approaches to assessment, and teachers/lecturers should use their professional judgement, subject knowledge and experience, as well as their understanding of their learners and their varying needs, to determine the most appropriate approaches and, where necessary, to consider workable alternatives.

Assessments must be fit for purpose and should allow for consistent judgements to be made by all teachers/lecturers. They should also be conducted in a supervised manner to ensure that the evidence provided is valid and reliable.

Unit assessment

Units will be assessed on a pass/fail basis. All Units are internally assessed against the requirements shown in the *Unit Specification.*

Assessments must ensure that the evidence generated demonstrates, at the least, the minimum level of competence for each Unit. Teachers/lecturers preparing assessment methods should be clear about what that evidence will look like.

Teachers/lecturers should refer to the following documents to ensure that all the requirements of Unit assessment are met:

- *Software Design and Development (Advanced Higher) Unit Specification*
- *Information System Design and Development (Advanced Higher) Unit Specification*
- *Unit Assessment Support package 1: portfolio approach*
- *Unit Assessment Support package 2: Unit-by-Unit approach for each Unit*
- *Unit Assessment Support package 3: Unit-by-Unit approach for each Unit*

Unit Assessment Support packages are available on SQA's secure website (www.sqa.org.uk/sqasecure)

In particular, assessors should refer to the **judging evidence tables** within the Unit Assessment Support packages for guidance on making assessment judgements for each Outcome and Assessment Standard.

The structure of an assessment used by a centre can take a variety of forms, for example:

- individual pieces of work could be collected in a folio as evidence for Outcomes and Assessment Standards
- assessment of each complete Outcome
- assessment that combines the Outcomes of one Unit
- assessment that requires more than the minimum competence, which would allow learners to prepare for the Course assessment

Teachers/lecturers should note that learners' day-to-day work may produce evidence which satisfies assessment requirements of a Unit, or Units, either in full or partially. Such naturally-occurring evidence may be used as a contribution towards Unit assessment. However, this naturally-occurring evidence must still be recorded and evidence such as written reports, recording forms, PowerPoint slides, drawings/graphs, video footage or observational checklists provided.

Added value

Advanced Higher Courses include assessment of added value which is assessed in the Course assessment.

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

In Advanced Higher Courses, added value involves the assessment of higher-order skills such as high-level and more sophisticated investigation and research skills, critical thinking skills and skills of analysis and synthesis. Learners may be required to analyse and reflect upon their assessment activity by commenting on it and/or drawing conclusions with commentary/justification. These skills contribute to the uniqueness of Advanced Higher Courses and to the overall higher level of performance expected at this level.

In this Course, added value will be assessed by means of a project and a question paper.

Preparation for Course assessment

Teachers/lecturers should refer to the following documents to ensure that all the requirements of the Course assessment are met:

- Advanced Higher Computing Science Course Assessment Specification
- Advanced Higher Computing Science General Assessment Information
- Advanced Higher Computing Science Coursework assessment task

Each Course has additional time which may be used at the discretion of the teacher/lecturer to enable learners to prepare for Course assessment. This time may be used near the start of the Course and 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. For this Course, the assessment methods are a project and a question paper.

Preparation for the Coursework project

In relation to preparing for the project, teachers/lecturers should explain the requirements to learners and the amount and nature of the support they can expect. However, at Advanced Higher level, it is expected that learners will work with more independence and less supervision and support.

Learners should be given opportunities to develop and practise the skills required for the project including:

- selecting a topic
- gathering and researching information
- project planning
- testing the solution
- evaluating the solution and development process
- presenting information

Detailed information on the Coursework project Component of Course assessment can be found in the *General Assessment Information* and the *Coursework assessment task* (project).

The **General Assessment Information** includes:

- an overview of the project, what it is for and its intentions
- the conditions for undertaking the project
- possibilities and limitations in relation to ‘reasonable support and guidance’
- the evidence that has to be gathered
- the actual Marking Instructions for each of the project aspects
- suggested computing science project ideas that might be interesting for the candidate to consider, although these are not mandatory

The **Coursework assessment task** includes:

- an overview of the project
- Marking Instructions (identical to those in the General Assessment Information)
- an assessment record
- comprehensive guidance for the candidate for each aspect of the project (this is found in Appendix 1, which can be detached and given to candidates)

Appendix 1 of the **Coursework assessment task** also provides the candidate with:

- guidance on recording progress
- the assessment requirements for each aspect of the project

- what to consider for each aspect
- additional guidance on research ethics if required
- suggested computing science project ideas

Preparation for the question paper

In relation to preparing for the question paper, learners should be given opportunities to develop and practise the skills required by:

- practising question paper techniques
- revising for the question paper

To support this learning, teachers/learners may find it helpful to refer to:

- Advanced Higher Computing Science Specimen Question Paper
- Advanced Higher Computing Science Exemplar Question Paper
- Advanced Higher Computing Science Guidance on the use of past paper questions

Questions assessing understanding and application of programming skills will be presented using SQA standardised reference language. The document outlining these terms at Advanced Higher level is available on SQA's website, alongside the Specimen Question Paper.

Authenticity

In terms of authenticity, there are a number of techniques and strategies to ensure that learners present work that is their own. Teachers/lecturers should put in place mechanisms to authenticate learners' evidence.

For more information, please refer to SQA's *Guide to Assessment*.

Further information on Course/Units

Approaches to learning and teaching

Both Units of the Course are designed to provide flexibility, personalisation and choice for both the learner and the teacher/lecturer.

Learning and teaching activities should be designed to stimulate the 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.

Software Design and Development (Advanced Higher) Unit

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

Tasks and activities throughout the Unit should be linked to relevant contexts and real-world applications, where appropriate. The *Unit Specification* defines the skills and knowledge required, but leaves complete freedom to the practitioner and learner to select interesting contexts and environments in which to develop these. 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 Advanced 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 within the Unit is at the discretion of the centre.

Outcome 1 and Outcome 2 simultaneously
The teacher/lecturer may wish to combine practical work with theory, using a series of practical tasks that focus on the links between the understanding of programming constructs, algorithms and data integration, and the development of complex modular programs that embrace these concepts. Implementation could be accompanied by a written or oral explanation of the design and coding of each of these aspects.

For example, when using structured data types to implement a complex standard algorithm, learners will be drawing on criteria from both Outcomes. The resulting development and implementation process allows opportunities for learners to demonstrate practical skills and a clear understanding of the implemented algorithm.

As learners further develop their programming skills by selecting and using advanced programming 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 construct complex programs to meet specific design requirements. This can be further linked to how programs interface with data sources such as files and databases.

Outcome 1 before Outcome 2
In order to deepen learners' understanding of advanced concepts and to enhance their ability to explain how complex modular programs work; they should use a variety of programming constructs, standard algorithms and structured data types. It would constitute good practice to demonstrate, discuss and research each of the items in the range given in the Unit. Examples shared with learners should focus initially on one element from the range, moving from a discussion of the element's use to a demonstration of its implementation. Once learners have developed an understanding of individual elements from the range, then additional tasks may focus on integrating a number of these elements in a single program. This integration of multiple elements demonstrates the concept of complex modular programs appropriate for learners to develop at this level.

For example, learners develop a program to read and display records. The program is extended to read and write the records to and from a stored file. Finally the program is further developed to sort the record structure, using one of the required complex algorithms.

A further exercise could take place using a variety of structured data types. Teachers/lecturers could provide learners with these structured data types embedded in example programs. Through a process of walking through the code and testing, learners would identify the purpose of these constructs and the code required to use them as functional elements in a program.

Once learners have a sound understanding of the role and purpose of structured data types, complex standard algorithms, and methods for interfacing with stored data, they should be well placed to develop their own complex modular programs that make use of these constructs, algorithms and data interface methods in their chosen software development environment. They should be able to apply the knowledge gained in Outcome 1 to select and use the appropriate advanced 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 by developing increasingly complex programs in one or more software development environments. As these increase in complexity, there are opportunities for learners to be exposed to new structured data types, more complex standard algorithms and to link their programs with a number of data sources. These activities would allow learners to fully understand the purpose of these elements within programs. Consequently, when learners are then asked to read and explain how well-structured, complex modular programs work in Outcome 1, they would have the necessary knowledge and skills to do this from their experience in coding in Outcome 2.

Outcome 3
It would be appropriate to equip learners with the necessary skills, knowledge and understanding of the content of Outcomes 1 and 2 before addressing the requirements of Outcome 3.

For example, learners will have experienced different software development languages and environments, and will be familiar with the main features of these, before learning about object-oriented coding. Alternatively, if learners were working in an object-oriented environment to meet Outcome 2, then this could also contribute to meeting Outcome 3.

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 complex modular programs work, to the point where they have the ability to read and explain code, describe the purpose of a range of structured data types, describe how a range of complex standard algorithms work, and describe The purpose of a range of programming constructs and how they work.

Teachers/lecturers 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 one or more programming languages, in order to learn the purpose of a range of structured data types and how they work within the program. Teachers/lecturers would explain these structured data types and demonstrate how they function, and the relative advantages of using these data structures. This would develop the knowledge and understanding required in the learner to be able to explain in part how well-structured, complex modular programs work.
- Learners could be provided with, or asked to locate, working programs that demonstrate reading data from a file or integrating a program with a data source, such as a relational database. Learners 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 binary search or a variety of sort algorithms. Teachers/assessors 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.

Notes on delivery of Outcome 2

It is envisaged that learners will develop a number of different complex 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
Web development environments which enable both server-side and browse-based scripting and the integration of database elements such as: PHP/JavaScript/MySQL, .NET/JavaScript/MSSQL, Java/JavaDB or Python/MySQLdb
Software development environments which are specifically suited to games development such as: C#, Greenfoot and Dark Basic
App development environments which are suited to the production of applications for handheld devices and smartphones such as: Xcode/Objective C for iOS, Eclipse/Java for Android or Corona SDK (Android and iOS support)
Desktop application programming environments such as: VB.Net and C/C++
Client-side scripting using JavaScript and popular libraries such as: JQuery
Server-side languages such as: Perl or Ruby for web applications

Further details and guidance on the assessment criteria to meet this Unit is available in the Unit specifications and the Unit Assessment Support packs.

Notes on delivery of Outcome 3

This Outcome involves learners in investigating simple object-oriented programs.

Object orientation is a key feature of modern software development and the object-oriented paradigm is common in many of the world's most popular programming languages and tools.

Learners should define classes, create objects and develop an understanding of instances. By exploring the concepts of encapsulation and inheritance, learners will develop an understanding of the use of methods and properties when working with classes and objects. It is important to keep the programs relatively simple, both those demonstrated to learners and those developed by learners, so that the focus is on the aspects of the object orientation.

Information System Design and Development (Advanced Higher) Unit

An investigatory approach is encouraged, with learners actively involved in developing their skills, knowledge and understanding advanced concepts and processes relating to information system design and development.

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

When delivering the Unit as part of the Advanced 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.

Notes on delivery of Outcome 1

Learners are required to develop at least one complex information system, such as a database-driven website, to write code to create a searchable structure with a user interface, and to implement a query language.

Opportunities for learning and teaching activities might include the following:

- developing a mobile application with database back-end
- creating a database-driven website
- creating a small social media website
- creating a small-scale information system for sporting or competition results

Notes on delivery of Outcome 2

It is envisaged that learners might engage with a number of case studies of example information systems projects. These would illustrate the key areas of study and provide the basis for learners to develop the broad knowledge and understanding required for both Unit and Course assessment.

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

- Applying project planning and management techniques, such as the development of a Gantt chart or the execution of a feasibility study in relation to a specific development.
- Carrying out usability testing on real systems including using tools to conduct accessibility testing.

Notes on delivery of Outcome 3

Learners are required to investigate a contemporary development which may be large or small-scale, but must be either a current or recent development. Their investigations could be based on any of the following (or other) areas:

- computer architecture (multi-processor computing systems, smart devices)
- artificial intelligence (robotics, intelligent systems, vision systems, speech systems)
- networking (cloud computing, security)
- interactive systems (social media, transactional systems, games)

Teachers/lecturers can provide some background information relating to aspects of the development to be investigated. Learners have to demonstrate that they have an understanding of the main purpose, features and applications of the development, a related technical challenge or area of current research, and be able to explain its legal and/or ethical implications, with an evaluation of its environmental, economic and/or social impact.

The investigation could involve site visits; it could also be based on printed or online sources of information.

Opportunities for different learning and teaching activities might include:

- Researching the energy use of data centres and the measures taken by companies to reduce this use and the environmental impact of their services.
- Researching the system architecture of contemporary information systems such as Wikipedia or Wordpress.com or large-scale cloud-infrastructure providers such as Amazon Web Services or Rackspace.
- Examining how the use of social media systems is changing the legal and ethical thinking relating to information publication and sharing.
- Studying the ethical issues raised by cloud computing and the presence of private personal data in data centres around the world.

Resources

Centres may find that existing hardware and software within the computing science classroom will provide all that is required to deliver the Course. The resources required 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, testing and development server environments)
- 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 (HTML5 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. However, it is important to ensure that, when using any pre-existing materials in this way, exemplification of the Computing Science Assessment Standards is continually referenced.

A number of online resources to support software development, and information systems design and development are available such as:

- codeacademy.com (Tutorials for JavaScript, Python, Ruby and other languages)
- programmr.com (Tutorials for Java and a number of other languages)
- hackety.com (Ruby related tutorials)
- learnpython.org (Python related tutorials)
- sqlzoo.net (SQL related tutorials)
- w3schools.com (general reference and tutorials for a number of languages)
- Wikipedia (including the infrastructure it runs onhttp://meta.wikimedia.org/wiki/Wikimedia_servers)
- Wordpress.com (and services offered http://vip.wordpress.com/our-services/)
- Amazon Web Services (Case Studies http://aws.amazon.com/solutions/case-studies/)
- High Scalability — architecture details of information systems such as YouTube, Twitter, Amazon and many others (http://highscalability.com/)

Other resources

- Computing at School Scotland – hosts an annual conference for Scottish computing science teachers and signposts courses, workshops and activities for continuing professional development.
- CompEdNet is an online site for computing science teachers that also provide materials and links to other teaching and learning resources.
- Scholar materials for Advanced Higher are available on Glow or at http://scholar.hw.ac.uk
- A parser for checking the validity of code written in the SQA standardised reference language is available at: http://haggis4sqa.appspot.com/haggisParser.html?variant=higher

Comparison of skills, knowledge and understanding for Higher and Advanced Higher

The following table shows the relationship between the mandatory Higher and Advanced Higher knowledge and understanding. Although some topics are carried through from Higher, particularly within the Software Design and Development Unit, there are substantial differences in the Advanced Higher Course. However, this table may be useful for:

- ensuring seamless progression between levels
- identifying important prior learning for learners at Advanced Higher

Teachers/lecturers 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.

The following mandatory generic concepts and vocabulary may be applied to both software design and development and information system design and development.		
Topic	**Higher**	**Advanced Higher**
Project planning and management		Description and exemplification of iterative project lifecycle: ♦ research: feasibility study, user surveys ♦ planning: scheduling, resources, Gantt chart ♦ analysis of: — user and business requirements — scope, constraints — functional and operational requirements ♦ requirement specifications (for end-users and technical team) ♦ design: — system modelling (data, process) — human computer interaction (user-centred) ♦ implementation: build, integration, deployment ♦ testing: — component, integrative, beta (acceptance), final — usability — accessibility — de-bugging techniques ♦ evaluation: usability, efficiency, effectiveness (goal-tracking), reliability, robustness ♦ maintenance: perfective, corrective, adaptive

Design notations and development methodologies	♦ 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. ♦ 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 methodologies	♦ UML (including class diagrams, use case) ♦ pseudocode ♦ wire-framing ♦ data dictionary ♦ other appropriate design notations ♦ problem decomposition ♦ iterative prototyping ♦ other contemporary methodologies
Languages and environments/ programming paradigms	♦ Description of the following language types: — low-level — high-level — procedural — declarative — object-oriented	♦ object-oriented (object, method, property, class, sub-class, encapsulation, inheritance, instantiation) ♦ imperative (variables, sequence, selection, iteration, modularity) ♦ concurrent (multiple threads, coordination)
Data types and structures	♦ Description, exemplification and implementation of the following data types and structures: — string — numeric (integer and real) variables — Boolean variables — 1-D arrays	♦ simple data types ♦ structured data types, including: — arrays of records — arrays of objects — 2-D arrays — linked lists — queues — stacks

	— records — arrays of records — sequential files (open, create, read, write, close)	
Algorithm specification/ standard algorithms	♦ Analysis, description, exemplification and implementation of standard algorithms including: — linear search — find minimum and maximum — count occurrences ♦ Analysis of other algorithms of similar complexity.	♦ binary search ♦ sort algorithms (selection using two lists, insertion, bubble) ♦ analysis of other algorithms of similar complexity ♦ benefits of other sort algorithms (including quicksort)
Computational constructs and principles	♦ 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	♦ reading and writing data to and from existing files and databases, including inserting, amending and deleting data ♦ form structure using HTML, form processing using server-side script ♦ recursion ♦ SQL operations, including: — DDL > create statement — DML > insert, update and delete statements > select statement (from, where, order by and group by clauses), aggregate functions (count, sum, min, max, avg) and equi-joins between tables
Testing and documenting solutions	♦ 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	
Low-level operations and computer architecture	♦ 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)	
Contemporary developments	♦ Exemplification of trends in the development of: — software development languages — software development environments — intelligent systems — online systems	
Structures and links (database)	♦ 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) — output (reports)	
Structures and links (web-based)	♦ Description, exemplification and implementation of the site structure of multi-level web-based information 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 websites. ♦ Description and exemplification of interactive web pages.	
User interface	♦ Description of problems with accessibility of computer systems and how they can be overcome including: — vision impairments — hearing impairment — motor and dexterity impairments	
Media types	♦ Description of the 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, 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 client device — validation of form data — connecting to database server — page generation	
Testing	♦ 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, functionality, users	♦ 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 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)	♦ 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)	♦ 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 — 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 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 system including: — hardware — transmission media — bandwidth	
Security risks	♦ 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 encryption used to secure transmission of data including use of public and private keys. ♦ Description and exemplification of digital certificates and signatures.	
Implications: legal, ethical, environmental	♦ Description, identification and implications for individuals, businesses and ISP's of the Regulation of	♦ intellectual property rights (including patent) ♦ storage of user data ♦ increasing use and power

economic and social	Investigatory Powers Act including: — intercepting and monitoring of electronic communications by government bodies — monitoring of employees communications — equipment and services used for surveillance ♦ Description and implications of the lifetime carbon footprint including: — manufacture of computer systems and peripherals — electricity use during a computer systems 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 impact:** ♦ Description and exemplification of the competitive advantage computer systems give businesses. ♦ Implications of the global marketplace for business	of intelligent systems ♦ energy (data centres, low-carbon equipment) ♦ online marketing (web, e-mail, text) ♦ analytics ♦ cyber security risks and precautions ♦ tracking, privacy, online safety ♦ social media ♦ implications of ‘big data’

	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.	

Appendix 1: Reference documents

The following reference documents will provide useful information and background.

- Assessment Arrangements (for disabled candidates and/or those with additional support needs) — various publications are available on SQA's website at: www.sqa.org.uk/sqa//14977.html.
- *Building the Curriculum 4: Skills for learning, skills for life and skills for work*
- *Building the Curriculum 5: A framework for assessment*
- Course Specification
- Design Principles for National Courses
- *Guide to Assessment*
- Principles and practice papers for curriculum areas
- *SCQF Handbook: User Guide* and SCQF level descriptors
- *SQA Skills Framework: Skills for Learning, Skills for Life and Skills for Work*
- *Skills for Learning, Skills for Life and Skills for Work: Using the Curriculum Tool*
- *Coursework Authenticity: A Guide for Teachers and Lecturers*

Administrative information

Published: May 2016 (version 2.1)

History of changes to Advanced Higher Course/Unit Support Notes

Version	Description of change	Authorised by	Date
2.0	Amended to reflect changes to Unit Specifications and Coursework project guidance materials. Adjustments made to the comparative tables in the 'Comparison of skills, knowledge and understanding for Higher and Advanced Higher' section, for consistency with Course Assessment Specification. General restructuring of document to signpost information available within other documents, and improve overall readability.	Qualifications Development Manager	May 2015
2.1	Amendments made to the 'Comparison of skills, knowledge and understanding for Higher and Advanced Higher'' to reflect the changes already made to Higher Course Assessment Specification (changes to Assessment Standards 1.1 and 1.3 and the deletion of Assessment Standard 3.2). For the SDD Unit, amendments have been made to information within the 'Approaches to learning and teaching' section, around the delivery of Outcome 1 and Outcome 3.	Qualifications Manager	May 2016