

COMPUTING
Advanced Higher

Second edition – published May 2008

COURSE Computing (Advanced Higher)

COURSE CODE C206 13

COURSE STRUCTURE

This Course has two mandatory Units and one optional Unit:

Mandatory Units:

<i>Unit Code</i>	<i>Unit Title</i>	<i>Credit and Duration</i>
<i>DF2Y 13</i>	<i>Software Development (AH)</i>	<i>1 credit (40 hours)</i>
<i>DM43 13</i>	<i>Developing a Software Solution (AH)</i>	<i>1 credit (40 hours)</i>

Optional Units-one selected from:

<i>Unit Code</i>	<i>Unit Title</i>	<i>Credit and Duration</i>
<i>DF31 13</i>	<i>Artificial Intelligence (AH)</i>	<i>1 credit (40 hours)</i>
<i>DF30 13</i>	<i>Computer Networking (AH)</i>	<i>1 credit (40 hours)</i>
<i>DM44 13</i>	<i>Computer Architecture (AH)</i>	<i>1 credit (40 hours)</i>

All Courses include 40 hours over and above the 120 hours for the Units. This may be used for induction, extending the range of learning and teaching approaches, support, consolidation, integration of learning and preparation for external assessment.

RECOMMENDED ENTRY

While entry is at the discretion of the centre, candidates would normally be expected to have attained one of the following, or equivalent:

- ◆ Higher Computing

PROGRESSION

This Course or its Units may provide progression in the following way:

- ◆ exit to Higher Education programme in Computer Science and related subjects

Administrative Information

Publication date: May 2008

Source: Scottish Qualifications Authority

Version: 02

© Scottish Qualifications Authority 2008

This publication 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 this Specification (including Unit Specifications) can be purchased from the Scottish Qualifications Authority for £7.50. **Note:** Unit Specifications can be purchased individually for £2.50 (minimum order £5).

National Course Specification: Course details

COURSE Computing (Advanced Higher)

RATIONALE

The development of computing over the last few decades has been significant in terms of speed and scope. It has had an effect on all aspects of our lives, and its future course remains unpredictable. Computing is both a science and a technology, and has wide-ranging social implications. It encompasses a very wide field of study, merging at its boundaries with many other disciplines. It provides us with many increasingly powerful hardware and software tools. Our society requires more and more individuals who have the skills to use these tools, who understand how they work, and who have the ability to develop new and improved tools.

The Advanced Higher Course in Computing is not only about learning to use current hardware and software. It is designed to provide candidates with both the necessary knowledge and understanding and the practical problem solving skills to enable them to become the ICT tool designers of the future.

The purpose of the Course is to build on the knowledge and understanding and practical skills developed by the candidate in the Higher Computing Course, and provide a useful bridge towards further study of computing in higher education. This bridge is achieved by a Course, which consolidates and extends learning, provides opportunity for independent and investigative work, while encouraging teamwork, and requires candidates to undertake and report on a significant software development project.

The importance of both knowledge and understanding, and related practical skills are reflected in the two Outcomes of each Unit. The ability to combine knowledge and understanding and practical skills to solve practical problems is a key theme of the Course.

AIMS

The aims of the Course are to extend:

- ◆ knowledge and understanding of computing concepts
- ◆ practical skills in the use of computer hardware and software
- ◆ ability to solve problems by applying knowledge, understanding and practical skills
- ◆ awareness of the professional, social, ethical and legal implications of computing
- ◆ ability to communicate computing concepts clearly and concisely using appropriate terminology.

Related to these aims, and underlying the study of computing are a number of **unifying themes** which are developed and exemplified throughout the Units of the Course. These themes are:

- ◆ technological development and progress
- ◆ factors affecting system performance
- ◆ objects and operations
- ◆ syntax and semantics
- ◆ social, professional, ethical and legal implications
- ◆ the relationship between software and hardware
- ◆ computing terminology
- ◆ the development process as it applies to both software and hardware systems.

National Course Specification: Course details (cont)

COURSE Computing (Advanced Higher)

The Course is designed to build on prior learning at Higher level (or equivalent) and to provide progression to degree Courses in Computer Science and related subjects.

COURSE CONTENT

The Course is made up of two mandatory Units, *Software Development*, *Developing a Software Solution*, and a choice of one from three optional Units.

The *Software Development* Unit develops the candidate's knowledge and skills in developing software through the use of a high level programming language. It builds on the learning laid down in the corresponding Unit at Higher level, or other equivalent experience. In the *Developing a Software Solution* Unit, the student will draw on the knowledge and understanding and practical skills developed through previous study, extend these through investigation, and then analyse, design and implement a solution to a significant computing problem. This solution must then be tested and evaluated, and a project report produced. Because computing is such a wide and rapidly developing field of study, a choice of three optional Units is offered, each one allowing the student to extend their learning in a contemporary aspect of applied Computing—*Artificial Intelligence*, *Computer Networking* or *Computer Architecture*.

To ensure consistency of terminology, the meanings of the technical terms used throughout this documentation (including the Unit Specifications) were taken from the British Computer Society's publication *A Glossary of Computing Terms*, 10th edition, pub.. Addison-Wesley, 2002. This glossary of terms will be used as a reference for all internal and external assessments, and its use is encouraged in all teaching and learning activities.

The Unit Specifications have been fully developed and provide detailed support notes to assist assessors in their understanding of Outcomes and Performance Criteria. The detailed content for each Unit is provided in the form of a table in the content/context section of each Unit Specification.

National Course Specification: Course details (cont)

COURSE Computing (Advanced Higher)

Content statements

The following pages of content statements describe in detail the knowledge and understanding which a candidate should be able to demonstrate in the external Course assessments, which will sample across these content statements. In order to achieve a Course award, candidates must also demonstrate an appropriate level of problem solving skills (application, analysis, synthesis and evaluation) based on these content statements.

This table defines the terms ‘knowledge and understanding’ and ‘problem solving’ as used in these arrangements in terms of the terminology used in Bloom’s Taxonomy of Learning:

Arrangements	Bloom’s Classification	Typical skills/words
<i>Knowledge and Understanding</i>	<i>Knowledge</i>	<i>recall of information (list, state, define, label, describe, name, identify)</i>
	<i>Comprehension</i>	<i>interpreting information in own words, grasping meaning (interpret, explain, discuss, predict, summarise, classify)</i>
<i>Problem Solving</i>	<i>Application</i>	<i>application to new situations (apply, demonstrate, show, relate, explain)</i>
	<i>Analysis</i>	<i>identification of patterns, recognising relationships (analyse, arrange, order, explain, connect, infer, compare, categorise)</i>
	<i>Synthesis</i>	<i>generalise, create new ideas, bring together from different sources, draw conclusions, predict (integrate, modify, design, compose, plan, arrange)</i>
	<i>Evaluation</i>	<i>make judgements, assess ideas, compare ideas, evaluate data (judge, evaluate, recommend, justify)</i>

Advanced Higher Computing: *Software Development* (mandatory Unit)

The candidate must demonstrate knowledge and understanding, practical skills and problem solving based on the following content statements:

Software development process

- ◆ Description of the progression through project proposal, feasibility study (economic, legal, technical and time), operational requirements document (ORD) and system specification, detailed design, implementation, component testing, system and acceptance testing, evaluation and maintenance
- ◆ Comparison of different user-interface design styles (menu-driven, textual, graphical)
- ◆ Explanation of module, component and beta (acceptance) testing
- ◆ Description of debugging techniques: dry runs, trace tables (tools), break points

Software development languages and environments

- ◆ Description of object-oriented language
- ◆ Comparison of object-oriented with procedural, declarative, event-driven and low level languages
- ◆ Explanation of the trends in language development (low level to high level, 4th generation)
- ◆ Description of the use and advantages of Computer-Aided Software Engineering (CASE) tools

National Course Specification: Course details (cont)

COURSE Computing (Advanced Higher)

Advanced Higher Computing: *Software Development* (mandatory Unit)

High level programming language constructs

- ◆ Description and exemplification of the following constructs in pseudocode and an appropriate high level language: file handling (write, read, delete item, create new file, open, read and close file)
- ◆ Description and exemplification of the following variable types/data structures: 2-D arrays, records, queues, stacks
- ◆ Description and exemplification of user-defined module libraries

Standard algorithms

- ◆ Description and exemplification of the following standard algorithm in pseudocode and an appropriate high level language: binary search
- ◆ Simple description and comparison of linear and binary search algorithms
- ◆ Simple description and comparison of sort algorithms in terms of number of comparisons and use of memory:
 - selection sort using two lists
 - simple sort
 - bubble sort

Advanced Higher Computing: *Developing a Software Solution* (mandatory Unit)

The candidate must demonstrate knowledge and understanding, practical skills and problem solving based on the following content statements:

- ◆ Description and exemplification of a problem specification
- ◆ Description and exemplification of the elements of the analysis stage:
 - statement of the requirements
 - identification of scope and boundaries of the problem
 - identification of functional requirements
- ◆ Description and exemplification of the elements of a project plan:
 - identification of sub-tasks
 - setting a realistic time-scale
 - application of appropriate project management technique
- ◆ Description of the need to:
 - consider and compare possible strategies using clearly specified criteria
 - select and justify a strategy
- ◆ Description and exemplification of aspects of a good user interface
- ◆ Description and exemplification of the elements of the testing stage of the process:
 - creation of a test plan
 - creation of test data
 - systematic testing
 - user questionnaire
 - summary of results
 - rectifying errors and bugs

National Course Specification: Course details (cont)

COURSE Computing (Advanced Higher)

Advanced Higher Computing: *Artificial Intelligence* (optional Unit)

The candidate must demonstrate knowledge and understanding, practical skills and problem solving based on the following content statements:

Search techniques

- ◆ Application of a problem solving approach to solve problems systematically:
 - problem abstraction (definition of the problem in terms of initial state, goal state, constraints)
 - symbolic representation (representation of transitional states in a state space graph, tree or production rules)
 - search strategy (selection of the best problem-solving technique and apply it to the problem)
- ◆ Representation of a problem as a start state (node), goal state (node), and transitions between states
- ◆ Representation of transitions as production rules
- ◆ Use of an AND-OR graph as a symbolic representation for appropriate problems
- ◆ Definition of a heuristic (or cost/evaluation function)
- ◆ Explanation of advantages and disadvantages of heuristic search techniques
- ◆ Description and use of the following search techniques: hill-climbing, best first search, A*
- ◆ Description of the relative advantages and disadvantages of each technique
- ◆ Brief description of the minimax procedure in the context of game playing

Knowledge representation

- ◆ Description of the software development process as it applies to declarative language programming
- ◆ Description of the purpose of frames to represent inheritable knowledge:
 - Comparison of frames and semantic networks
 - Distinction between *classes* and *instances*
 - Description and use of slots, current values, default values, inheritance
 - Use of a frame notation to represent a simple hierarchy of domain knowledge
- ◆ Description and exemplification of the following features in Prolog (or similar declarative language): recursion, list processing
- ◆ Explanation of the concepts of goal, sub-goal, instantiation, unification
- ◆ Description and exemplification of multiple inheritance
- ◆ Explanation of the benefit of rules involving inheritance

Rule-based systems

- ◆ Representation of knowledge in terms of IF...THEN rules
- ◆ Explanation and exemplification of the use of certainty factors
- ◆ Identification and explanation of how forward and backward chaining inference may be used or combined to help solve a given problem
- ◆ Description of the main characteristics of a forward chaining system: working memory; conflict set; conflict resolution
- ◆ Explanation of why conflict resolution strategies are required
- ◆ Calculation of the certainty of a conclusion using the formula $CF_{\text{conc}} = CF_{\text{rule}} \times \min(CF_{\text{cond1}}, CF_{\text{cond2}}, \dots)$

National Course Specification: Course details (cont)

COURSE Computing (Advanced Higher)

Applications and uses of artificial intelligence

Computer vision:

- ◆ Application of the Waltz algorithm to a tri-hedral figure to produce a valid labelling
- ◆ Explanation of the role of search in the application of the Waltz algorithm
- ◆ Description of the causes and effects of ambiguities

Natural language processing:

- ◆ Simple description of main stages of natural language understanding (speech recognition, syntactic analysis, semantic analysis, pragmatic analysis)
- ◆ Description of ambiguities which can occur at each stage
- ◆ Definition of grammar rules for a simple subset of English involving noun phrase, verb phrase, determiner, noun, proper noun, pronoun, verb, preposition, adjectives, with a simple vocabulary to reflect the grammar
- ◆ Implementation of a simple parse tree
- ◆ Explanation of the role of search in the parsing process

Robotics:

- ◆ Description of the classical 'blocks world' environment
- ◆ Definition of the actions (stack, unstack, pickup, putdown) and states (on, ontable, clear, holding, empty) in the blocks world
- ◆ Application of rules for solving simple problems in the blocks world
- ◆ Description of the role of planning
- ◆ Explanation of the role of search in the problem solving process

Machine Learning:

- ◆ Description of and distinction between: rote learning, learning from advice, learning from experience, learning from examples (inductive learning); explanation-based learning; learning by discovery; learning by analogy
- ◆ Exemplification of each type of learning
- ◆ Recommendation of a learning method for a given scenario

Advanced Higher Computing: *Computer Networking* (optional Unit)

The candidate must demonstrate knowledge and understanding, practical skills and problem solving based on the following content statements:

Network protocols

- ◆ Explanation of the need for organisations enforcing standards including ISO and IEEE
- ◆ Description of mapping TCP/IP layers to OSI model layers
- ◆ Explanation of the purpose of common protocols (SMTP, POP and MIME)
- ◆ Description of CIDR and binary subnet masks
- ◆ Description of Trace route and Ping in terms of troubleshooting

Network applications

- ◆ Description of the parts of an e-mail message; header (recipients address and other info) and body (containing the message to be sent)
- ◆ Brief description of sending and receiving e-mail including the role of SMTP: connection setup, mail transfer, connection termination

National Course Specification: Course details (cont)

COURSE Computing (Advanced Higher)

- ◆ Description of a web page using HTML tags (start, header, body, title, style, font size, alignment, section headers, colours and hyperlinks)
- ◆ Combination of tags to create a single line of code
- ◆ Description of the process of requesting a web page by a client from a server and its transfer-using HTTP from a server to a client:
 - HTTP (overview, types of connections - end to end TCP connection and not end to end TCP connections, three forms of intermediate - proxy, gateway and tunnel)
 - types of data transmitted
 - messages (request from client to server and response from server to client, fields-request line, response line, general header, request header, response header, entity header, entity body)
- ◆ Description of commonly used plug-ins to enhance browser functionality for portable documents, multimedia elements and streaming audio/video, naming currently used examples
- ◆ Description and uses of Java applets and ActiveX
- ◆ Description of a video telephone call and its technical implications (hardware, software, data transmission and data compression):
 - video telephony in the context of teleconferencing (quality of video-dpi, sample rate)
 - need for compression of video signal

Network security

- ◆ Description of the following methods for network and communication security:
 - conventional encryption (plaintext, encryption algorithm, secret key, ciphertext and decryption algorithm)
 - message authentication (using conventional encryption and without message encryption)
 - public-key encryption and digital signatures (plaintext, encryption algorithm, public and private key, ciphertext and decryption algorithm)
 - Internet architecture security
- ◆ Description of the generic denial of service attack and countermeasures taken:
 - Smurf: (attacker sends spoofed ICMP ECHO packet to broadcast address of a network, amplification of attack, bandwidth consumption; prevention: disable directed broadcast functionality of border router, configure operating system to silently discard broadcast ICMP ECHO packets)
 - SYN Flood: (attacker sends SYN packet from spoofed address, recipient sends SYN/ACK packet to spoofed address, recipient does not receive ACK from spoofed address and connection remains until timed out; consequence: usually a server is taken out; prevention: increase size of connection queue, decrease connection established timeout period, employ vendor specific patches)
 - DNS: (attacker can try to convince a target nameserver to cache information that maps to a nonexistent IP address effectively denying that service; prevention: upgrade to latest version of BIND)
- ◆ Description of a few simple firewall rules used to protect a LAN with an Internet connection from outside attacks
- ◆ Description of types of backup; full, incremental and differential

Data transmission

- ◆ Comparison of bandwidth of different transmission systems; UTP, co-axial, fibre and radio waves
- ◆ Explanation of the advantages, disadvantages and characteristics (data transfer rate, range and frequency) of modern wireless communication methods

National Course Specification: Course details (cont)

COURSE Computing (Advanced Higher)

- ◆ Description of data standards of modern wireless communication methods (802.11a, 802.11b, 802.11g, bluetooth, HiperLAN2 and Ultrawideband)
- ◆ Description of the security and performance issues of modern wireless communication methods (WPAN and wireless LAN)
- ◆ Description of remote access: dial-up protocols (SLIP and PPP), virtual private network protocols (PPTP and L2TP)

Advanced Higher Computing: *Computer Architecture* (optional Unit)

The candidate must demonstrate knowledge and understanding, practical skills and problem solving based on the following content statements:

Computer structure

- ◆ Detailed description of processor and registers including MAR, MDR, IR, PC and general purpose registers
- ◆ Description of relationship between buses and processor registers
- ◆ Description of the fetch-execute cycle in terms of buses and registers
- ◆ Description of the following types of computer memory:
 - internal memory (registers, cache, main memory).
 - external memory (magnetic disk, CD-ROM, CD-RW, DVD-ROM, re-writable DVD, tape)
- ◆ Description of these memory technologies in a hierarchy, using the following factors: decreasing cost per bit, increasing capacity, increasing access time, decreasing frequency of access
- ◆ Explanation of the importance of these factors when designing a system for performance
- ◆ Description of the structured use of cache memory to improve processor performance referring to the use of level 1 and level 2 cache as well as the use of static RAM
- ◆ Description of how memory interleaving operates
- ◆ Description of how memory interleaving can improve system performance by enabling multiple memory accesses simultaneously
- ◆ Description of how direct memory access can improve system performance
- ◆ Description of the effect on system performance of increasing clock speeds and increasing the width of data buses with reference to 8 bit, 16 bit, 32 bit and 64 bit microprocessors
- ◆ Description of PCI and PCI-X buses in terms of the following characteristics: throughput described as bits per second, width, multipoint topology, function
- ◆ Description of the importance of bus throughput for system performance

Processor structure

- ◆ Description and exemplification of the structure of assembly language instructions as op-code and operand
- ◆ Description and exemplification of the classification of assembly language instructions using the following categories: data transfer, arithmetic, logical, shift and rotate, branch
- ◆ Description of the following key features which distinguish RISC from CISC: a limited and simple instruction set, the use of register oriented instructions with limited memory access, the use of limited addressing modes and a large bank of registers
- ◆ Description of the performance gain derived from the use of SIMD (single instruction multiple data) instructions with reference to multimedia processing operations
- ◆ Description of how the following techniques can be used to optimise the instruction and data stream: branch prediction, data flow analysis, speculative loading of data, speculative execution of instructions, predication
- ◆ Description of how pipelining operates and how it can improve system performance

National Course Specification: Course details (cont)

COURSE Computing (Advanced Higher)

- ◆ Description of possible problems with pipelining caused by branch instructions and by instructions of different lengths
- ◆ Description of how superscalar processing operates and how it can improve system performance

Processor development

- ◆ Description of the evolution of the following microprocessor architectures: the Power PC series, the Intel X86 series and the Intel IA-64 in terms, where appropriate, of the following features and techniques: increasing clock speeds, data bus widths, pipelining, superscalar processing, branch prediction, speculative loading of data and executing of instructions, predication, the number and function of registers used, SIMD, RISC, CISC
- ◆ Explanation of the relationship between these developments and system performance
- ◆ Description of how parallel computers function referring to their use of:
 - local (cache) as well as main memory
 - pipelining
 - local pathways and packet switching to achieve communication between CPUs
- ◆ Description of the performance benefits of parallel computers

Operating systems

- ◆ Description of the following techniques used by operating systems to manage memory: variable partitioning of memory, use of best-fit, worst fit and first fit algorithms for the allocation of memory
- ◆ Comparison of the operation of best fit, worst fit and first fit algorithms in terms of efficient use of memory
- ◆ Description of the relationship between the size of data blocks used in memory allocation and access speed
- ◆ Description of the need for operating systems to schedule programs in a multitasking system
- ◆ Comparison of the following types of pre-emptive scheduling in terms of their effect on system performance: round robin scheduling, multi-level feedback queue
- ◆ Description of the use of direct memory access to manage input/output data transfers in order to improve system performance
- ◆ Description of the key function of the file management system as mapping between the logical view of files and their physical location
- ◆ Description of contiguous and non-contiguous methods of allocation of files to available storage space
- ◆ Description of the way in which the trend towards designing GUI based on the design principle of 'convenience for the user' leads to: increasing software complexity, more demands on system resources (processor and memory) and a consequent burden on system performance
- ◆ Illustration of the demands a GUI makes on the system resources by describing the steps involved in processing a simple operation such as a mouse click
- ◆ Comparison with the demands a CLI makes on the system
- ◆ Descriptions of the trend to expand the role of operating systems to include services and provide capabilities that were formerly within applications themselves
- ◆ Description of those services as:
 - providing a standard look and feel for applications
 - simplifying and extending graphic capabilities of application programs
 - improving the capability of programs to communicate and pass data
 - the capability of launching one application inside another
- ◆ Description of the use of libraries of objects that applications call as required

National Course Specification: Course details (cont)

COURSE Computing (Advanced Higher)

- ◆ Description of techniques used by OS to support communication between applications and document embedding
- ◆ Description of procedures used to control user access in both multi-user and single user systems
- ◆ Description of the purpose of file attributes
- ◆ Description of backup facilities
- ◆ Exemplification and comparison of the features of **two** specific operating systems

National Course Specification: Course details (cont)

COURSE Computing (Advanced Higher)

ASSESSMENT

To achieve the Course award the candidate must pass the Units as well as the Course assessment. The candidate's grade is based on the Course assessment.

DETAILS OF THE INSTRUMENTS FOR COURSE ASSESSMENT

Course assessment should provide opportunities to demonstrate:

- ◆ retention of knowledge, understanding and skills over a longer period of time
- ◆ integration of knowledge, understanding and skills acquired in the Units
- ◆ application of knowledge, understanding and skills in more complex contexts
- ◆ application of knowledge, understanding and skills in less familiar contexts

The Course assessment for Computing at Advanced Higher level will consist of two components:

	Time Allocation	Mark Allocation
Coursework Project	Completed during Course	80
Question Paper	2 hours 30 minutes	120

The Coursework Project provides candidates with the opportunity to demonstrate and integrate the practical skills, knowledge and understanding from the Units, and apply these in a more complex practical context.

The purpose of the Question Paper is to assess the candidate's competence to integrate and retain knowledge and understanding and demonstrate higher order cognitive abilities across the contents of all the Units, and in varied contexts, and to demonstrate the ability to communicate computing concepts clearly.

Coursework Project

The candidate should select a project:

- ◆ at an appropriate level for the Advanced Higher Course
- ◆ which builds on learning from the core Units
- ◆ which is achievable with 40 hours.

Advice on appropriate projects is included in the support notes for the *Developing a Software Solution* Unit.

The project will be undertaken under 'open-book' conditions. Collaborative projects are to be encouraged, but the assessor should ensure that each candidate's individual contribution to the project can be clearly identified and assessed.

For Unit assessment, the candidate should provide a record showing evidence of:

- ◆ analysis
- ◆ design
- ◆ implementation
- ◆ testing.

National Course Specification: Course details (cont)

COURSE Computing (Advanced Higher)

This evidence need not be a formal report. It could consist of a ‘log book’, notes, references to sources, annotated hard copy of implementation at various stages during the development, hard copy of screen shots, notes on testing. It is expected that the production of this evidence should be achieved during the 40 hours of the *Developing a Software Solution* Unit.

To pass the Unit, there must be evidence that the candidate has:

- ◆ analysed a practical problem at an appropriate level
- ◆ designed, implemented and tested a software solution based on the problem analysis.

Note: the implementation (which may be incomplete) may be developed using any appropriate software development environment, not necessarily a high level programming language.

For Course assessment, the candidate should provide:

- ◆ the problem specification
- ◆ evidence of project planning
- ◆ evidence of a completed solution (preferably files on a CD plus hard copy of coding/data files, screen shots)
- ◆ user documentation and technical documentation
- ◆ an evaluation report.

The project will be marked internally, based on this evidence and observation of the candidate at work, using the marking scheme provided by SQA, but be subject to moderation. The marking scheme will provide a mark out of 80, which will be submitted directly to SQA.

The marks available for each aspect will be:

◆ specification and plan	15
◆ implemented product	28
◆ process skills	11
◆ user documentation	6
◆ technical documentation	4
◆ evaluation report	16

It is expected that the production of the user and technical documentation, and evaluation report, should be achieved using up to 10 hours of the ‘additional’ time provided during the Course.

Question Paper

The Question Paper will comprise a single paper of 2 hours and 30 minutes duration. The total marks available will be 120. The examination will be set and marked by SQA. The paper will be composed of two sections:

SECTION 1 (60 marks)

This will consist of questions requiring extended responses requiring structuring and reasoning. These questions will test both knowledge and understanding and problem solving. Approximately 1/3 of the marks will be for knowledge and understanding and 2/3 for problem solving. The questions will be set in a range of familiar and less familiar contexts.

National Course Specification: Course details (cont)

COURSE Computing (Advanced Higher)

The questions will sample across the content statements associated with the mandatory Units, and will require integration of knowledge across the two Units. Candidates will be expected to tackle all questions.

SECTION 2 (60 marks)

This will have three sub-sections, one for each of the optional Units. Candidates will be expected to tackle all the questions within any one sub-section. The questions will require extended responses. Approximately 1/3 of the marks will be for knowledge and understanding and 2/3 for problem solving as in section 1, and the questions, which will sample across the content statements for the optional Unit, will also require some integration of knowledge from the mandatory Units.

Further details about assessment for this Course can be found in NAB materials, the Course Assessment Specification and the Specimen Question Paper.

NB: refer to the table on page 6 of these arrangements for guidance on the meaning of the terms 'knowledge and understanding' and 'problem solving' in this context.

National Course Specification: Course details (cont)

COURSE Computing (Advanced Higher)

GRADE DESCRIPTIONS AT GRADE A AND GRADE C

The candidate's grade will be based on the total score obtained by adding the marks from the Coursework Project and the Question Paper. The descriptions below indicate the nature of achievement for the award at Grade C and Grade A in the Course.

GRADE C	GRADE A
<ul style="list-style-type: none"> retention of knowledge, understanding and skills over a longer period of time 	
Candidates are able to describe and explain some of the facts and concepts to the standard defined by the Performance Criteria.	Candidates are able to describe and explain most of the facts and concepts to the standard defined by the Performance Criteria.
Candidates are able to demonstrate some practical skills to the standards defined by the Performance Criteria.	Candidates are able to demonstrate practical skills, which exceed the standards defined by the Performance Criteria.
<ul style="list-style-type: none"> integration of knowledge, understanding and skills acquired in Units 	
Candidates are able to demonstrate their knowledge and understanding in the context of specific Units.	Candidates are able to demonstrate the integration of knowledge and understanding from more than one Unit.
Candidates are able to demonstrate their practical skills in the context of specific Units.	Candidates are able to demonstrate the integration of practical skills from more than one Unit.
<ul style="list-style-type: none"> application of knowledge, understanding and skills in more complex contexts 	
Candidates are able to apply knowledge and understanding to solve problems in straightforward contexts relating to a single Unit.	Candidates are able to apply knowledge and understanding to solve problems in more complex contexts relating to more than one Unit.
Candidates are able to apply practical skills to solve problems in straightforward contexts relating to a single Unit.	Candidates are able to apply practical skills to solve problems in more complex contexts relating to more than one Unit.
<ul style="list-style-type: none"> application of knowledge, understanding and skills in less familiar contexts 	
Candidates are able to apply knowledge, understanding and skills to solve problems in familiar contexts.	Candidates are able to apply and transfer knowledge, understanding and skills to solve problems in less familiar contexts .
Candidates are able to carry out defined tasks to the standards defined in the Performance Criteria.	Candidates are able to resolve non-routine problems that arise during practical activity, by independent research.

National Course Specification: Course details (cont)

COURSE Computing (Advanced Higher)

ESTIMATES AND APPEALS

Estimates

In preparing estimates, evidence of performance should be considered from across the breadth of coverage of the Course and must take account of performance in the whole Course. Evidence used to compile estimates and support appeals must be valid and reliable, must take account of performance in both of the Course components, the Coursework Project and the Question Paper and must relate to the Course Grade Descriptions. The assessment instruments which are used to generate evidence for Estimates must, therefore, allow candidates opportunities to demonstrate attainment against the Course Grade Descriptions.

Further advice on the preparation of estimates is given in the Course Assessment Specification and in the SQA guidance on submitting estimates and appeals.

Appeals

Evidence assembled in support of an Assessment Appeal should cover the content of the Course. Ideally, this will comprise evidence generated by a properly constructed, integrated prelim which reflects the Course assessment in range, balance and depth.

Although a prelim is not mandatory it is desirable. This is because it provides evidence of how well a candidate can perform in conditions which replicate the Course Assessment. The prelim can test retention of knowledge and understanding across all areas of the Course content; can provide opportunities for integration; can allow candidates to demonstrate problem solving and the application of their knowledge in less familiar and more complex contexts. It can also result in evidence which is produced within the same time constraint as that specified by the Course Assessment. Any prelim should replicate the style, level of demand and mark allocation of the Specimen SQA examination.

When developing prelim papers, centres should bear in mind that past papers, including SQA past papers, will not be accepted in their entirety. However, questions selected carefully from a minimum of three past papers, preferably adapted (to ensure the breadth and depth of coverage required to satisfy the Course Grade Descriptions) can be combined to form a valid assessment instrument for a prelim. Centres must also be certain that the question paper used for a prelim is not in the public domain and has not been previously seen by candidates. It is the responsibility of centres to ensure the validity, reliability and security of assessment instruments used for Estimates and Appeals.

Centres that submit an integrated test or prelim that only covers the mandatory Units should also submit an additional test covering the chosen optional Unit. Furthermore, this test must integrate some knowledge and understanding from the mandatory Units. As the Coursework Project is expected to represent a candidate's best practical work, there is no need to submit additional evidence of problem solving in practical contexts.

Unit Assessments are designed to allow candidates to demonstrate the knowledge and understanding and practical skills required to pass the Units. They cannot provide evidence of how a candidate can perform against the Course Grade Descriptions. Unit Assessments **do not** provide opportunities for the candidate to demonstrate problem solving skills, integration across Units, retention of knowledge, and application of knowledge in more complex and less familiar contexts, and therefore **do not** provide sufficient evidence for appeals.

National Course Specification: Course details (cont)

COURSE Computing (Advanced Higher)

QUALITY ASSURANCE

All National Courses are subject to external marking and/or moderation. External markers, visiting examiners and moderators are trained by SQA to apply national standards. SQA is currently seeking to assist centres by preparing exemplification of standards materials in a number of subject areas. This will be rolled out to all subjects in due course.

The Units of all Courses are subject to internal moderation and may also be chosen for external moderation. This is to ensure that national standards are being applied across all subjects.

Courses may be assessed by a variety of methods. Where marking is undertaken by a trained marker in their own time, markers meetings are held to ensure that a consistent standard is applied. The work of all markers is subject to scrutiny by the Principal Assessor and a PA report is published for all subjects.

National Course Specification: Course details (cont)

COURSE Computing (Advanced Higher)

APPROACHES TO LEARNING AND TEACHING

The five main aims of the Course are to extend:

- ◆ knowledge and understanding of computing concepts
- ◆ practical skills in the use of computer hardware and software
- ◆ ability to solve problems by applying knowledge, understanding and practical skills
- ◆ appreciation of the professional, social, ethical and legal implications of computing
- ◆ ability to communicate computing concepts clearly and concisely using appropriate terminology

There is no prescriptive ‘best way’ to approach the teaching and learning of this Course. However a holistic approach is recommended which relates each of these aims to the computing facts and concepts being studied. Within each Unit, there is a combination of knowledge and understanding with practical problem solving skills. Teachers and lecturers are encouraged to provide learning experiences which blend together the acquisition of knowledge and understanding, the development of practical skills and opportunities to apply these to solve problems.

Throughout the Course, reference should be made to professional, social, ethical and legal implications where appropriate, and to ‘real world’ applications. Candidates should be encouraged to develop the use of appropriate computing terminology to communicate their understanding.

Related to the Course aims, a number of unifying themes have been identified which should be used to bring a coherence to the Course. Most of these themes can be illustrated and exemplified in each of the Units of the Course. These themes include:

- ◆ technological development and progress
- ◆ factors affecting system performance
- ◆ objects and operations
- ◆ syntax and semantics
- ◆ social, professional, ethical and legal implications
- ◆ the relationship between software and hardware
- ◆ computing terminology
- ◆ the development processes it applies to both software and hardware systems.

Candidates will require individual access to appropriate computer hardware and software throughout the Course. More detailed guidance is given within the support notes for each Unit.

Teachers and lecturers are encouraged to make use of the wide range of teaching and learning materials (both paper-based and electronic) which have been developed to support this Course.

The Units of the Course may be studied sequentially or in parallel (or a combination of these). The candidate’s project is likely to be based on learning from both the *Software Development* and *Developing a Software Solution* Units, and so these would normally precede the project. However, as the project is an extended piece of work, it is advisable to encourage the candidate to begin this as early as possible.

National Course Specification: Course details (cont)

COURSE Computing (Advanced Higher)

APPROACHES TO LEARNING AND TEACHING

A typical Course plan might therefore take the form:

August-December:	Software Development
August-September	Developing a Software Solution (introduction to project)
September-February	work on Coursework Project
January	preparation for prelim examination(s)
February-March	optional Unit
March	writing up project report
April	flexible time

Preliminary examinations, if used, should be timed to allow maximum coverage of the three Course Units. This can be achieved by holding the prelim as late as possible (end of March), or by having an early prelim covering two Units, with a supplementary prelim later covering the third Unit, and integration with the mandatory Units.

The teaching and learning and internal assessment of the three Units of the Course is designed to be completed within 120 hours. This includes practical activities in preparation for the practical coursework task. As centres are advised to allow 160 hours for the delivery of a National Course, this leaves up to 40 hours of flexible time.

Use of the additional 40 hours

Appropriate activities for this time include

- ◆ an introduction to the Course
- ◆ revision of required prior learning
- ◆ consolidation and integration of learning
- ◆ remediation and re-assessment
- ◆ formative assessment (class tests)
- ◆ preliminary examination(s)
- ◆ preparation for external assessment
- ◆ completion of the project report
- ◆ extending the range of study

CANDIDATES WITH ADDITIONAL SUPPORT NEEDS

This Unit Specification is intended to ensure that there are no artificial barriers to learning or assessment. The additional support needs of individual candidates should be taken into account when planning learning experiences, selecting assessment instruments or considering alternative Outcomes for Units. For information on these, please refer to the SQA document *Guidance on Assessment Arrangements for Candidates with Disabilities and/or Additional Support Needs (SQA 2004)*.

National Course Specification: Course details (cont)

COURSE Computing (Advanced Higher)

COMPUTING IN A BROADER CONTEXT

Careful consideration should be given to any sections of the Course or Units which could contribute to current national initiatives, including education for enterprise, citizenship, employeeship and personal and social development. This might be achieved through suitable choice of contexts for problem solving activities within the Course.

The development of communication skills and problem solving skills are important cross-curricular themes, and opportunities should be taken to develop links with other aspect of each candidate's curriculum.

A number of national initiatives and programmes have been designed to promote themes that are important to contemporary society such as citizenship and enterprise. These themes contribute to individual subjects and Courses by making connections beyond the subject boundaries and enrich the learning experience. Similarly, specialist knowledge and skills developed through the study of a particular subject contributes to the understanding of these themes.

There are several opportunities within Computing (Advanced Higher) for assessors to help candidates make links to cross-curricular themes. Some suggestions are given below.

Cross-curricular theme	Course content
Enterprise in Education	Understanding the world of work in relation to: <ul style="list-style-type: none">◆ feasibility studies and ORDS◆ cost implications of errors in SD process◆ efficiency of using CASE tools and macros
Education for Citizenship	Developing citizenship skills: <ul style="list-style-type: none">◆ working independently◆ locating, using and communicating information and ideas using ICT◆ persevering in the face of setback and practical difficulties
Financial Education	Consumer's rights responsibilities and protection in relation to network security. Importance of designing user-friendly interfaces.

National Unit Specification: general information

UNIT **Software Development (Advanced Higher)**

NUMBER DF2Y 13

COURSE Computing (Advanced Higher)

SUMMARY

This Unit is designed to develop knowledge and understanding of software development and to develop practical skills in software development through the use of a high level language within an appropriate software development environment. In particular, it will consolidate and extend candidates' experience of standard language constructs and algorithms, and extend understanding of the software development process.

On completion of the Unit, the candidate should be able to apply this knowledge and understanding, and these skills to solve practical problems.

It is designed for candidates undertaking the Advanced Higher Computing Course, but it is also suitable for anyone wishing to extend and deepen their experience of software development beyond Higher level.

OUTCOMES

1. Demonstrate knowledge and understanding of the principles of software development, software development languages and environments, high level language constructs and standard algorithms.
2. Demonstrate practical skills in the context of software development using contemporary hardware and an appropriate software development environment.

Administrative Information

Superclass: CB

Publication date: April 2005

Source: Scottish Qualifications Authority

Version: 01

© Scottish Qualifications Authority

This publication 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 this unit specification can be purchased from the Scottish Qualifications Authority. The cost for each unit specification is £2.50. (A handling charge of £1.95 will apply to all orders for priced items.)

National Unit Specification: general information (cont)

UNIT Software Development (Advanced Higher)

RECOMMENDED ENTRY

While entry is at the discretion of the centre, candidates would normally be expected to have attained one of the following or equivalent:

- ◆ *Software Development* (Higher) Unit
- ◆ Higher Computing

CREDIT VALUE

1 credit at Advanced Higher (8 SCQF credit points at SCQF level 7*)

**SCQF credit points are used to allocate credit to qualifications in the Scottish Credit and Qualifications Framework (SCQF). Each qualification in the Framework is allocated a number of SCQF credit points at an SCQF level. There are 12 SCQF levels, ranging from Access 1 to Doctorates.*

CORE SKILLS

There is no automatic certification of Core Skills or Core Skills components in this Unit.

National Unit Specification: statement of standards

UNIT Software Development (Advanced Higher)

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

OUTCOME 1

Demonstrate knowledge and understanding of the principles of software development, software development languages and environments, high level language constructs and standard algorithms.

Performance criteria

- (a) A range of advanced computing terminology is used appropriately.
- (b) Technically accurate descriptions and explanations are related to a range of familiar and unfamiliar contexts.
- (c) Descriptions of high level language constructs and standard algorithms are correct.
- (d) Conclusions, predictions and generalisations are made from knowledge and understanding.

Evidence requirements

Written or oral evidence that the candidate can describe and explain the principles of software development accurately and concisely. Evidence should be obtained using questions in a closed-book test, under supervision, lasting no more than 45 minutes. The test must sample across the range of the content (see Computing (Advanced Higher) Course content) in each of the following areas:

- ◆ the software development process
- ◆ software development languages and environments
- ◆ high level programming language constructs
- ◆ standard algorithms.

(The content statements are also reproduced for convenience as a table in the support notes for this Unit).

The standard to be applied is illustrated in the National Assessment Bank items available for this Unit.

If a centre wishes to design its own assessments for this Unit, they should be of a comparable standard.

National Unit Specification: statement of standards (cont)

UNIT Software Development (Advanced Higher)

OUTCOME 2

Demonstrate practical skills in the context of software development using contemporary hardware and an appropriate software development environment.

Performance criteria

- (a) A wide range of appropriate hardware is used effectively and efficiently.
- (b) A wide range of appropriate features of software is used effectively and efficiently.
- (c) Practical tasks are planned and organised independently.
- (d) Practical tasks are undertaken in an appropriate range of familiar and unfamiliar contexts.

Evidence requirements

Observation checklist showing that the candidate has carried out practical activities, demonstrating all of the following practical skills, as defined in the content statements (see Computing (Advanced Higher) Course content).

- ◆ analysis and design, with feasibility study
- ◆ design and creation of user-friendly interface
- ◆ comparison of two sorting or searching algorithms
- ◆ implementation of file handling
- ◆ use of module libraries
- ◆ testing of software
- ◆ producing user and technical documentation
- ◆ evaluating software.

Hard copy evidence should be provided of implementation and **two** other of these activities.

These practical skills may all be demonstrated in a single extended software development task, or in a number of smaller tasks.

The candidate will be allowed access to books, notes and on-line help while completing the task(s).

(The content statements are also reproduced for convenience as a table in the support notes for this Unit)

The standard to be applied is illustrated in the National Assessment Bank items available for this Unit.

If a centre wishes to design its own assessments for this Unit, they should be of a comparable standard.

National Unit Specification: support notes

UNIT Software Development (Advanced Higher)

This part of the Unit specification is offered as guidance. The support notes are not mandatory.

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

GUIDANCE ON THE CONTENT AND CONTEXT FOR THIS UNIT

The content for this Unit is detailed below (and also in the National Course Specifications: Course details). Content statements in the left-hand column describe the content covered in the corresponding Unit at Higher level, and are included here to clarify the context for the new learning for this Unit. They indicate the prior learning required by the candidate before undertaking new learning within this Unit. Content in the right-hand column is the new content for this Unit.

Content Statements: Software development process	
<i>Higher</i>	<i>Advanced Higher</i>
<i>Explanation of the iterative nature of the software development process. Description of the purpose of the software specification as a legal contract. Explanation of the importance of each stage (analysis, design, implementation, testing, documentation, evaluation, maintenance) of the development process.</i>	Description of the progression through project proposal, feasibility study (economic, legal, technical and time), operational requirements document (ORD) and system specification, detailed design, implementation, component testing, system and acceptance testing, evaluation and maintenance.
<i>Identification of the personnel involved at each stage (client, system analyst, project manager, programmer, independent test group) and brief descriptions of their roles.</i>	
<i>Description and exemplification of pseudocode and one graphical design notation structure diagram or other suitable) including data flow.</i>	Comparison of different user-interface design styles (menu-driven, textual, graphical).
<i>Description and exemplification of the top-down design and stepwise refinement.</i>	
<i>Explanation of the need for systematic and comprehensive testing.</i>	Explanation of module, component and beta (acceptance) testing. Description of debugging techniques: dry runs, trace tables (tools), break points.
<i>Explanation of the need for documentation at each stage.</i>	
<i>Evaluation of software in terms of robustness, reliability, portability, efficiency and maintainability.</i>	
<i>Description and exemplification of corrective, adaptive and perfective maintenance.</i>	

National Unit Specification: support notes (cont)

UNIT Software Development (Advanced Higher)

Content Statements: Software development languages and environments	
<i>Higher</i>	Advanced Higher
<i>Description and comparison of procedural, declarative and event-driven languages.</i>	Description of object-oriented language. Comparison of object-oriented with procedural, declarative, event-driven and low level languages. Explanation of the trends in language development (low level to high level, 4 th generation).
<i>Comparison of the functions, uses and efficiency of compilers and interpreters.</i>	
<i>Description of the features and uses of scripting language (including creating and editing a macro). Explanation of the need for and benefits of scripting languages.</i>	
<i>Description of the use of module libraries.</i>	Description of the use and advantages of Computer-Aided Software Engineering (CASE) tools

Content Statements: High level programming language constructs	
<i>Higher</i>	Advanced Higher
<i>Description and exemplification of the following constructs in pseudocode and an appropriate high level language:</i> <ul style="list-style-type: none"> ◆ <i>string operations (concatenation and substrings)</i> ◆ <i>formatting of I/O</i> ◆ <i>CASE (or equivalent multiple outcome selection)</i> 	Description and exemplification of the following constructs in pseudocode and an appropriate high level language: <ul style="list-style-type: none"> ◆ <i>file handling: write, read, delete item, create new file, open, read, and close file</i>
<i>Description and exemplification of real, integer and boolean variables and 1-D arrays</i>	Description and exemplification of the following variable types/data structures: 2-D arrays, records, queues, stacks.
<i>Description and exemplification of procedures/subroutines, user-defined functions, modularity, parameter passing (in, out, in/out), callpass by reference/value, local and global variables, scope.</i>	Description and exemplification of <ul style="list-style-type: none"> ◆ <i>user-defined module libraries</i>

National Unit Specification: support notes (cont)

UNIT Software Development (Advanced Higher)

Content Statements: Standard algorithms	
<i>Higher</i>	Advanced Higher
Description and exemplification of the following standard algorithms in pseudocode and an appropriate high level language: <ul style="list-style-type: none"> ◆ <i>linear search</i> ◆ <i>counting occurrences</i> ◆ <i>finding min/max</i> 	Description and exemplification of the following standard algorithm in pseudocode and an appropriate high level language: <ul style="list-style-type: none"> ◆ binary search Simple description and comparison of sort algorithms in terms of number of comparisons and use of memory: <ul style="list-style-type: none"> ◆ selection sort using two lists ◆ simple sort ◆ bubble sort
	Simple description and comparison of linear and binary search algorithms

National Unit Specification: support notes (cont)

UNIT Software Development (Advanced Higher)

Sort algorithms

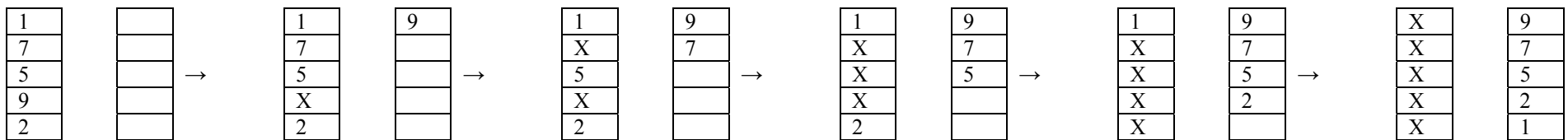
The three sort algorithms required for this Unit are as follows:

Selection sort using two lists

Set up new empty list
for number of items in list
Find max in unsorted list
Transfer it to first empty element of new list
Replace with dummy data in old list

Next

Two list sort



National Unit Specification: support notes (cont)

UNIT Software Development (Advanced Higher)

Simple Sort

Start with item 1

If item 2 > item 1, swap

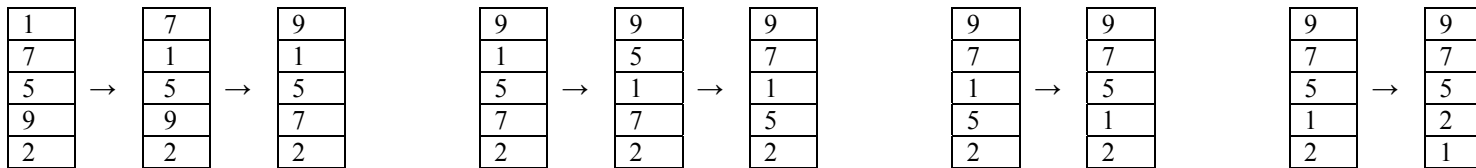
If item 3 > item 1, swap

And so on to the end of the list

Repeat process with item 2

And so on to the end of the list

Simple sort



National Unit Specification: support notes (cont)

UNIT Software Development (Advanced Higher)

Bubble Sort

Start with item 1

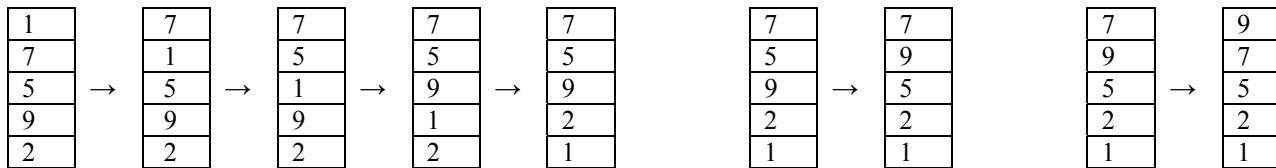
If item 2 > item 1, swap

If item 3 > item 2, swap

And so on to the end of the list

Repeat process until no more swaps

Bubble sort



Animation web site

Animations of the simple sort and bubble sort can be found at:

<http://www.cs.brockport.edu/cs/javasort.html>

National Unit Specification: support notes (cont)

UNIT Software Development (Advanced Higher)

GUIDANCE ON LEARNING AND TEACHING APPROACHES FOR THIS UNIT

Candidates will require individual access to appropriate computer hardware and an appropriate software development environment throughout this Unit. While the learning will usually be achieved in the context of a single software development environment, students will benefit from having some experience of alternative software development environments.

The two Outcomes should be delivered in an integrated way rather than sequentially. For Outcome 2, the practical activities should be taught and used to illustrate and exemplify the knowledge and understanding required for Outcome 1. These practical activities can be used to generate evidence for Outcome 2.

Candidates who have completed the *Software Development* Unit at Higher level should already have covered the content listed in the left-hand column of the content statement grids, but may need to revise this material before progressing to the right hand column.

The main purpose of this Unit is to consolidate and extend candidates' experience of standard language constructs and algorithms, and to extend an understanding of the software development process. For those candidates undertaking the Computing Course, this will provide a suitable basis for undertaking the '*Developing a Software Solution*' Unit and the Coursework Project.

The amount of time spent on each content area will vary depending on the teaching methodology used and the ability and prior experience of the candidates. However, the following times are suggested as a rough guide:

software development process	4 hours
languages and environments	5 hours
language constructs	12 hours
standard algorithms	15 hours

1½ hours should be set aside to:

- ◆ administer the Outcome 1 test
- ◆ gather evidence for Outcome 2

A further 2 ½ hours is allowed for remediation and re-assessment if required.

If the Unit is delivered as part of a Course, the Course documentation will provide further information on teaching and learning in a Course context, including the identification of a number of 'themes' to facilitate holistic learning across the Course.

National Unit Specification: support notes (cont)

UNIT Software Development (Advanced Higher)

GUIDANCE ON APPROACHES TO ASSESSMENT FOR THIS UNIT

National Assessment Bank tests have been created specifically to assess Outcome 1 of the Unit. This assessment consists of a closed book test, and must be conducted under supervision. In order to gain success in this Outcome, the candidate must achieve at least the cut-off score for the test. If a centre wishes to design its own assessments for this Unit, they should be of a comparable standard.

Outcome 2 requires the candidate to demonstrate practical skills while developing software using an appropriate high-level language environment. These practical skills may all be demonstrated in a single extended software development task, or in a number of smaller tasks. The skills will normally be demonstrated by the candidate during the teaching and learning activities in the Unit, rather than as separate formal assessment activities. The candidate will be allowed access to books, notes and on-line help while completing the task(s).

To gain success in this Outcome, the candidate must demonstrate practical skills in the following contexts and at an appropriate level as defined by the content statements (see Computing (Advanced Higher) Course content):

- ◆ analysis and design (including feasibility study)
- ◆ design and creation of a user-friendly interface
- ◆ comparison of two sorting or searching algorithms
- ◆ implementation of file handling
- ◆ use of module libraries
- ◆ testing of software
- ◆ producing user and technical documentation
- ◆ evaluating software

A pro-forma observation checklist for Outcome 2 is provided in the National Assessment Bank materials.

Hard copy evidence is required of implementation and **two** other of these skills; this need not be formal documentation - it could include hand-written notes on design, hard copy of coding, or screen shots demonstrating implementation and/or testing.

All evidence must be retained by the centre. The assessment of this Unit is subject to moderation by SQA.

National Unit Specification: support notes (cont)

UNIT Software Development (Advanced Higher)

CANDIDATES WITH ADDITIONAL SUPPORT NEEDS

This Unit Specification is intended to ensure that there are no artificial barriers to learning or assessment. The additional support needs of individual candidates should be taken into account when planning learning experiences, selecting assessment instruments or considering alternative Outcomes for Units. For information on these, please refer to the document *Guidance on Assessment Arrangements for Candidates with Disabilities and/or Additional Support Needs* (SQA, 2004).

National Unit Specification: general information

UNIT **Developing a Software Solution (Advanced Higher)**

NUMBER DM43 13

COURSE Computing (Advanced Higher)

SUMMARY

This Unit is designed to develop the ability to analyse a complex computing problem and to design, implement and test a software based solution.

It is suitable for candidates who have a background in software development up to Advanced Higher level, and is designed as a preparation for undertaking the Coursework Project for the Advanced Higher Computing Course.

OUTCOMES

1. Demonstrate knowledge and understanding of the analytical approach to the development of a software solution to a computing problem.
2. Demonstrate practical skills by analysing, designing, implementing and testing a software solution to a computing problem.

RECOMMENDED ENTRY

While entry is at the discretion of the centre, candidates would normally be expected to have attained one of the following, or equivalent:

- ◆ Higher Computing
- ◆ *Software Development* (Advanced Higher) Unit

Administrative Information

Superclass: CB

Publication date: April 2005

Source: Scottish Qualifications Authority

Version: 01

© Scottish Qualifications Authority

This publication 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 this unit specification can be purchased from the Scottish Qualifications Authority. The cost for each unit specification is £2.50. (A handling charge of £1.95 will apply to all orders for priced items.)

National Unit Specification: general information (cont)

UNIT Developing a Software Solution (Advanced Higher)

CREDIT VALUE

1 credit at Advanced Higher (8 SCQF credit points at SCQF level 7).*

**SCQF credit points are used to allocate credit to qualifications in the Scottish Credit and Qualifications Framework (SCQF). Each qualification in the Framework is allocated a number of SCQF credit points at an SCQF level. There are 12 SCQF levels, ranging from Access 1 to Doctorates.*

CORE SKILLS

Core Skills Components for this Unit

Critical Thinking (Higher)
Planning and Organising (Higher).

National Unit Specification: statement of standards

UNIT Developing a Software Solution (Advanced Higher)

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

OUTCOME 1

Demonstrate knowledge and understanding of the analytical approach to the development of a software solution to a computing problem.

Performance criteria

- (a) The terminology of the analytical approach is used appropriately.
- (b) Descriptions and explanations are technically accurate and concise.
- (c) The stages of the software development process are exemplified clearly.

Evidence requirements

Written or oral evidence that the candidate can describe and explain the analytical approach to the development of a software solution to a computing problem. Evidence should be obtained using questions in a closed-book test, under supervision lasting no more than 20 minutes. The test must sample across the range of the content (see Computing (Advanced Higher) Course content) relevant to this Unit.

(The content statements are also reproduced for convenience as a table in the support notes for this Unit)

The standard to be applied is illustrated in the National Assessment Bank items available for this Unit.

If a centre wishes to design its own assessments for this Unit, they should be of a comparable standard.

National Unit Specification: statement of standards (cont)

UNIT Developing a Software Solution (Advanced Higher)

OUTCOME 2

Demonstrate practical skills by analysing, designing, implementing and testing a software solution to a computing problem.

Performance criteria

- (a) Analysis of the problem is clear and concise.
- (b) Appropriate information resources are used effectively.
- (c) Design of solution is clearly expressed and complete.
- (d) The solution demonstrates use of complex programming techniques.
- (e) Hardware is used effectively and efficiently to implement a solution.
- (f) Software is used effectively and efficiently to implement a solution.
- (g) Test plan is clear and appropriate, and testing is carried out systematically.

Evidence requirements

Written or oral evidence that the candidate can analyse a complex computing problem, and design, implement and test a software solution to this problem. The evidence should be generated by the candidate during the development of a software solution to a real computing problem. This evidence will take the form of a record of work.

The record of work need not comprise a formal report but must provide evidence that the candidate has completed each of the following stages of the development process to the level defined by the Performance Criteria and the content statements (see Computing (Advanced Higher) Course content):

- ◆ analysis
- ◆ design
- ◆ implementation
- ◆ testing.

The candidate will be expected to use appropriate books, notes and on-line help whilst completing the record of work.

(The content statements are also reproduced for convenience as a table in the support notes for this Unit)

The standard to be applied is illustrated in the National Assessment Bank items available for this Unit.

If a centre wishes to design its own assessments for this Unit, they should be of a comparable standard.

National Unit Specification: support notes

UNIT Developing a Software Solution (Advanced Higher)

This part of the Unit specification is offered as guidance. The support notes are not mandatory.

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

GUIDANCE ON THE CONTENT AND CONTEXT FOR THIS UNIT

Candidates are expected to be able to conduct independent research at this level and it is anticipated that there will be limited teacher/lecturer exposition of the underlying principles and concepts. Candidates will use a wide range of sources of information including paper based and electronic.

It is expected that the candidate will apply and extend their knowledge of software development by solving a problem which offers an appropriate depth of complexity.

The content for this Unit is detailed below (and also in the National Course Specifications: Course details.)

Content Statements: Developing a software solution
Description and exemplification of a problem specification. Description and exemplification of the elements of the analysis stage: <ul style="list-style-type: none">◆ statement of the requirements◆ identification of scope and boundaries of the problem◆ identification of functional requirements
Description and exemplification of the elements of a project plan: <ul style="list-style-type: none">◆ identification of sub-tasks◆ setting a realistic time-scale◆ application of appropriate project management technique
Description of the need to: <ul style="list-style-type: none">◆ consider and compare possible strategies using clearly specified criteria◆ select and justify a strategy
Description and exemplification of aspects of a good user interface.
Description and exemplification of the elements of the testing stage of the process: <ul style="list-style-type: none">◆ creation of a test plan◆ creation of test data◆ systematic testing◆ user questionnaire◆ summary of results◆ rectifying errors and bugs

National Unit Specification: support notes (cont)

UNIT Developing a Software Solution (Advanced Higher)

GUIDANCE ON LEARNING AND TEACHING APPROACHES FOR THIS UNIT

Candidates will require individual access to appropriate computer hardware and software throughout this Unit.

The two Outcomes should be delivered in an integrated way rather than sequentially. The stages of the software development process should be taught and then applied to an appropriate practical task.

The amount of time spent on each area of content will vary, depending on the teaching methodology used and the ability and prior experience of the candidates. However, the following times are suggested as a rough guide:

problem specification	2 hours
project planning	2 hours
analysis	4 hours
design	4 hours
implementation	20 hours
testing	4 hours

1½ hours should be set aside to:

- ◆ administer the Outcome 1 test
- ◆ gather evidence for Outcome 2

A further 2½ hours is allowed for remediation and re-assessment if required.

If the Unit is delivered as part of the Advanced Higher Computing Course, candidates will extend the work of this Unit into the Coursework element of external assessment, by producing a formal report on the problem and its solution.

The following guidance is offered to help centres to determine whether potential computing problems are at an appropriate level for this Unit.

Appropriate problems for this Unit should:

- ◆ allow the candidate to demonstrate knowledge and understanding gained during completion of the *Software Development Systems* Unit at Advanced Higher level
- ◆ provide a suitable level of difficulty and complexity, appropriate to Advanced Higher level and allowing the candidate to develop a software solution to a computing problem that requires the development of complex algorithms
- ◆ allow sufficient scope for development, enabling candidates to demonstrate all knowledge and skills required for assessment purposes
- ◆ be possible using hardware and software resources available in the teaching centre
- ◆ be possible within the 40 hours available for completion of this Unit.

Suitable problems might include development of:

- ◆ a simulation of a card game with a graphical user interface and an element of artificial intelligence developed using an appropriate high level programming language
- ◆ an encryption/decryption program involving file handling and complex key algorithms
- ◆ an implementation of a natural language interpreter using a declarative language and based on a recognised grammar and parsing technique

National Unit Specification: support notes (cont)

UNIT Developing a Software Solution (Advanced Higher)

- ◆ a computer aided learning package which demonstrates an assembly language emulator for teaching the *Computer Systems* Unit and incorporating a number of commonly used op-codes and addressing modes
- ◆ a network application to allow peer-to-peer chat facilities
- ◆ a website which incorporates advanced techniques such as the development of user interactivity, form filling or a front end to a database system making extensive use of PHP, Java or PERL
- ◆ an expert system developed without the aid of an expert system shell.

It is important that the problem chosen allows scope for the candidate to demonstrate appropriate complex programming techniques. The following examples, while similar to those listed above, are unlikely to provide sufficient depth and challenge to be suitable for this Unit:

- ◆ a simulation of a card game using only code modules easily available in the public domain
- ◆ an encryption/decryption program using only a simple scrambling algorithm such as one which would be within the reach of a Higher candidate
- ◆ an implementation of a natural language interpreter based on a very limited vocabulary and simplistic approach to grammar rules
- ◆ a computer aided learning package using only simple multimedia authoring tools
- ◆ a website which does not include advanced techniques such as scripting
- ◆ an expert system for a careers database which does not include advanced rules or other constructs available within an expert system shell.

GUIDANCE ON APPROACHES TO ASSESSMENT FOR THIS UNIT

National Assessment Bank tests have been created specifically to assess Outcome 1 of the Unit. This assessment consists of a closed book test, and must be conducted under supervision. In order to gain success in this Outcome, the candidate must achieve at least the cut-off score for the test. If a centre wishes to design its own assessments for this Unit, they must be of a comparable standard.

Outcome 2 requires the candidate to demonstrate practical skills while using contemporary hardware and software to implement and test the software solution to a computing problem. The skills will normally be demonstrated by the candidate during the development of the software solution, rather than as separate formal assessment activities. The candidate will be allowed access to books, notes and on-line help while completing the tasks. The practical skills should be demonstrated in the context and at a level defined by the content statements (see Advanced Higher Course content).

The assessment of Outcome 2 is based on a record of work produced by the candidate. The record of work will normally be generated by the candidate during the development of a software solution, rather than as separate formal assessment activities. The evidence will be generated by the candidate as a series of notes, annotated diagrams, screen shots and notes on testing. A formal report is not required for Unit assessment.

The candidate will be allowed access to books, notes and on-line help while completing the record of work.

To gain success in Outcome 2, the candidate must demonstrate practical skills in each of the following stages of the software development process:

- ◆ analysis
- ◆ design
- ◆ implementation
- ◆ testing

National Unit Specification: support notes (cont)

UNIT Developing a Software Solution (Advanced Higher)

All evidence must be retained by the centre. The assessment of this Unit is subject to central moderation by SQA.

A pro-forma observation checklist for Outcome 2 is provided in the National Assessment Bank materials.

CANDIDATES WITH ADDITIONAL SUPPORT NEEDS

This Unit Specification is intended to ensure that there are no artificial barriers to learning or assessment. The additional support needs of individual candidates should be taken into account when planning learning experiences, selecting assessment instruments or considering alternative Outcomes for Units. For information on these, please refer to the document *Guidance on Assessment Arrangements for Candidates with Disabilities and/or Additional Support Needs* (SQA, 2004).

National Unit Specification: general information

UNIT Artificial Intelligence (Advanced Higher)

NUMBER DF31 13

COURSE Computing (Advanced Higher)

SUMMARY

This Unit is designed to develop knowledge and understanding of the principles of artificial intelligence and practical skills related to artificial intelligence through the use of contemporary hardware and software. This knowledge and understanding, and these practical skills, may then be applied by the candidate to solve practical problems related to artificial intelligence.

It is designed as an option for candidates undertaking the Advanced Higher Computing Course, but is also suitable for anyone wishing to extend and deepen their experience of artificial intelligence beyond Higher level.

OUTCOMES

1. Demonstrate knowledge and understanding of the principles, techniques and applications of artificial intelligence.
2. Demonstrate practical skills in the context of artificial intelligence using contemporary hardware and software.

RECOMMENDED ENTRY

While entry is at the discretion of the centre, candidates would normally be expected to have attained one of the following, or equivalent:

- ◆ *Artificial Intelligence* (Higher) Unit
- ◆ Higher Computing

Administrative Information

Superclass: CB

Publication date: April 2005

Source: Scottish Qualifications Authority

Version: 01

© Scottish Qualifications Authority 2005

This publication 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 this unit specification can be purchased from the Scottish Qualifications Authority. The cost for each unit specification is £2.50. (A handling charge of £1.95 will apply to all orders for priced items.)

National Unit Specification: general information (cont)

UNIT Artificial Intelligence (Advanced Higher)

CREDIT VALUE

1 credit at Advanced Higher (8 SCQF credit points at SCQF level 7).*

**SCQF credit points are used to allocate credit to qualifications in the Scottish Credit and Qualifications Framework (SCQF). Each qualification in the Framework is allocated a number of SCQF credit points at an SCQF level. There are 12 SCQF levels, ranging from Access 1 to Doctorates.*

CORE SKILLS

There is no automatic certification of Core Skills or Core Skills components in this Unit.

National Unit Specification: statement of standards

UNIT Artificial Intelligence (Advanced Higher)

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

OUTCOME 1

Demonstrate knowledge and understanding of the principles, techniques and applications of artificial intelligence.

Performance criteria

- (a) A wide range of advanced computing terminology is used appropriately.
- (b) Technically accurate descriptions and explanations are related to familiar and unfamiliar contexts.
- (c) Conclusions, predictions and generalisations are made from knowledge and understanding.

Evidence requirements

Written or oral evidence that the candidate can describe and explain the principles, techniques and applications of artificial intelligence accurately and concisely. Evidence should be obtained using questions in a closed-book test, under supervision, lasting no more than 45 minutes. The test must sample content (see Artificial Intelligence (Advanced Higher) Course content) in each of the following areas:

- ◆ search techniques
- ◆ knowledge representation
 - semantic networks/frames
 - declarative language programming
 - rule-based systems
- ◆ applications of artificial intelligence
 - computer vision
 - natural language processing
 - robotics
 - machine learning

(The content statements are also reproduced for convenience as a table in the support notes for this Unit).

The standard to be applied is illustrated in the National Assessment Bank items available for this Unit.

If a centre wishes to design its own assessments for this Unit, they should be of a comparable standard.

National Unit Specification: statement of standards (cont)

UNIT Artificial Intelligence (Advanced Higher)

OUTCOME 2

Demonstrate practical skills in the context of artificial intelligence using contemporary hardware and software.

Performance criteria

- (a) Hardware and software is used independently, effectively and efficiently.
- (b) Practical tasks are planned and organised independently.
- (c) Practical tasks are undertaken in an appropriate range of familiar and unfamiliar contexts.

Evidence requirements

Observational checklist showing that the candidate has demonstrated the following skills in the context and at a level defined by the content statements (see Computing (Advanced Higher) Course content):

- ◆ construction of a knowledge base of facts and rules in a declarative language
- ◆ implementation of list processing, involving recursion
- ◆ creation of queries to elicit information from a knowledge base
- ◆ testing a knowledge base
- ◆ creation of a rule-based system.

Hard copy evidence should be provided of both the knowledge base and the rule-based system constructed.

These practical skills may be demonstrated in a number of individual, focused tasks, or in a single extended task.

The candidate will be allowed access to books, notes and on-line help while completing the task(s).

(The content statements are also reproduced for convenience as a table in the support notes for this Unit.)

The standard to be applied is illustrated in the National Assessment Bank items available for this Unit.

If a centre wishes to design its own assessments for this Unit, they should be of a comparable standard.

National Unit Specification: support notes

UNIT Artificial Intelligence (Advanced Higher)

This part of the Unit specification is offered as guidance. The support notes are not mandatory.

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

GUIDANCE ON THE CONTENT AND CONTEXT FOR THIS UNIT

The content for this Unit is detailed overleaf (and also in the National Course Specifications: Course details.)

Content statements in the left-hand column describe the relevant content covered in the corresponding Unit at Higher level, and are included here to clarify the context for the new learning for this Unit. They indicate the prior learning required by the candidate before undertaking new learning within this Unit. Content in the right-hand column is the new content for this Unit.

Content Statements: Search techniques	
<i>Higher</i>	Advanced Higher
	Application of a problem solving approach to solve problems systematically: <ul style="list-style-type: none"> ◆ Problem abstraction: definition of the problem in terms of initial state, goal state, constraints. ◆ Symbolic representation: representation of transitional states in a state space graph, tree or production rules. ◆ Search strategy: selection of the best problem-solving technique and apply it to the problem.
	Representation of a problem as a start state (node), goal state (node), and transitions between states. Representation of transitions as production rules. Use of an AND-OR graph as a symbolic representation for appropriate problems.
<i>Comparison of depth-first and breadth-first search (order of visiting nodes, memory implications, advantages and disadvantages, need for backtracking), and exemplification on a search tree. Description and exemplification of combinatorial explosion. Description and exemplification of the use of heuristics to reduce search time/space.</i>	Definition of a heuristic (or cost/evaluation function). Explanation of advantages and disadvantages of heuristic search techniques. Description and use of the following search techniques: <ul style="list-style-type: none"> ◆ Hill-climbing ◆ Best first search ◆ A* Description of the relative advantages and disadvantages of each technique. Brief description of the minimax procedure in the context of game playing.

National Unit Specification: support notes (cont)

UNIT Artificial Intelligence (Advanced Higher)

Content Statements: Knowledge Representation	
<i>Higher</i>	<i>Advanced Higher</i>
<i>Description of the software development process as it applies to declarative language programming.</i>	Description of the software development process as it applies to declarative language programming
<i>Creation of a semantic net from given problem statement.</i>	Description of the purpose of frames to represent inheritable knowledge: <ul style="list-style-type: none"> ◆ comparison of frames and semantic networks ◆ distinction between <i>classes</i> and <i>instances</i> ◆ description and use of slots, current values, default values, inheritance ◆ use of a frame notation to represent a simple hierarchy of domain knowledge
<i>Description and exemplification of the following features in Prolog (or similar declarative language):</i> <ul style="list-style-type: none"> ◆ multi-argument clauses ◆ recursive and non recursive rules ◆ complex queries: (multiple variable, conjunction of queries) ◆ negation ◆ inheritance 	Description and exemplification of the following features in Prolog (or similar declarative language): <ul style="list-style-type: none"> ◆ recursion ◆ list processing
<i>Explanation of the concepts of goal, sub-goal, instantiation, matching.</i>	Explanation of the concepts of goal, sub-goal, instantiation, unification.
<i>Explanation of complex manual trace: multiple level including backtracking.</i>	
	Description and exemplification of multiple inheritance.
<i>Explanation of the importance of the order of rules.</i>	Explanation of the benefit of rules involving inheritance.

National Unit Specification: support notes (cont)

UNIT Artificial Intelligence (Advanced Higher)

Content Statements: Rule-based systems

<i>Higher</i>	Advanced Higher
<p>Expert Systems <i>Description of the components of an expert system (knowledge base, inference engine, user interface with justification/explanation, working memory). Distinction between an expert system and an expert system shell. Description of contemporary applications of expert systems. Description of advantages of expert systems (including permanence, cost effectiveness, consistency, portability). Description of disadvantages of expert systems (including narrow domain, lack of ‘common sense’, need for expertise to set up and maintain, inability to acquire new knowledge, inflexibility). Description of moral issues (including medical implications). Description of legal issues (including responsibility when advice is wrong).</i></p>	<p>Rule-based systems Representation of knowledge in terms of IF..THEN rules. Explanation and exemplification of the use of certainty factors. Identification and explanation of how forward and backward chaining inference may be used or combined to help solve a given problem. Description of the main characteristics of a forward chaining system: working memory; conflict set; conflict resolution. Explanation of why conflict resolution strategies are required. Calculation of the certainty of a conclusion using the formula $CF_{conc} = CF_{rule} \times \min(CF_{cond1}, CF_{cond2}, \dots)$</p>

Content Statements: Applications and uses of artificial intelligence	
<i>Higher</i>	Advanced Higher
<p>Computer vision <i>Description of the problems of interpreting 2D images of 3D objects. Description of the stages of computer vision (image acquisition, signal processing, edge detection, object recognition, image understanding).</i></p>	<p>Computer vision Application of the Waltz algorithm to a tri-hedral figure to produce a valid labelling. Explanation of the role of search in the application of the Waltz algorithm. Description of the causes and effects of ambiguities.</p>
<p>Natural language processing (NLP): <i>Identification of the main stages of NLP (speech recognition, natural language understanding (NLU), natural language generation, speech synthesis). Explanation of some difficulties in NLP (including ambiguity of meaning; similar sounding words; inconsistencies in grammar of human language; changing nature of language) Identification of applications of NLP (including automatic translation, speech driven software, NL search engines, NL database interfaces).</i></p>	<p>Natural language processing Simple description of main stages of natural language understanding (speech recognition, syntactic analysis, semantic analysis, pragmatic analysis). Description of ambiguities which can occur at each stage. Definition of grammar rules for a simple subset of English involving noun phrase, verb phrase, determiner, noun, proper noun, pronoun, verb, preposition, adjectives, with a simple vocabulary to reflect the grammar. Implementation of a simple parse tree Explanation of the role of search in the parsing process.</p>

National Unit Specification: support notes (cont)

UNIT Artificial Intelligence (Advanced Higher)

<p>Intelligent robots: <i>Explanation of the difference between dumb and intelligent robots.</i> <i>Description of contemporary research and developments.</i> <i>Description of social and legal implications of the increasing use of intelligent robots.</i> <i>Descriptions of practical problems (including processor power, power supply, mobility, vision recognition, navigation, path planning, pick and place) and strategies for overcoming these problems.</i></p>	<p>Robotics Description of the classical “blocks world” environment. Definition of the actions (stack, unstack, pickup, putdown) and states (on, ontable, clear, holding, empty) in the blocks world. Application of rules for solving simple problems in the blocks world. Description of the role of planning Explanation of the role of search in the problem solving process.</p>
	<p>Machine learning Description of and distinction between: rote learning, learning from advice, learning from experience, learning from examples (inductive learning); explanation-based learning; learning by discovery; learning by analogy. Exemplification of each type of learning. Recommendation of a learning method for given scenario.</p>

GUIDANCE ON LEARNING AND TEACHING APPROACHES FOR THIS UNIT

Candidates will require individual access to appropriate computer hardware and software throughout this Unit.

In particular, candidates will require access to an implementation of Prolog (or a similar declarative language. Many versions are available either commercially or as freeware. A search on the World Wide Web will lead to sources for these.

The two Outcomes should be delivered in an integrated way rather than sequentially. For Outcome 2, the practical activities, both computer based and non-computer based, should be taught and used to illustrate and exemplify the knowledge and understanding required for Outcome 1, whenever this is possible. At the very least, candidates should carry out practical tasks using Prolog, and have experience of creating a rule-based system using an expert system shell. Practical illustrations of other applications and uses of artificial intelligence should be provided where suitable hardware and software is available.

The amount of time spent on each area of content will vary depending on the teaching methodology used and the ability and prior experience of the candidates. However, the following times are suggested as a rough guide:

Problems and search	12 hours
Knowledge representation	10 hours
Applications and uses of Artificial Intelligence	14 hours

1½ hours would be set aside to:

- ◆ administer the Outcome 1 test
- ◆ gather evidence for Outcome 2

National Unit Specification: support notes (cont)

UNIT Artificial Intelligence (Advanced Higher)

A further 2 ½ hours is allowed for remediation and re-assessment if required.

If the Unit is delivered as part of a Course, the Course documentation will provide further information on teaching and learning in a Course context, including the identification of a number of ‘themes’ to facilitate holistic learning across the Course.

GUIDANCE ON APPROACHES TO ASSESSMENT FOR THIS UNIT

National Assessment Bank tests have been created specifically to assess Outcome 1 of the Unit. This assessment consists of a closed book test, and must be conducted under supervision. In order to gain success in this Outcome, the candidate must achieve at least the cut-off score for the test. If a centre wishes to design its own assessments for this Unit, they should be of a comparable standard.

Outcome 2 requires the candidate to demonstrate practical skills while using contemporary hardware and software. These practical skills will normally be demonstrated in a single extended task or a number of relatively small tasks. The task(s) may be undertaken by the candidate as part of the teaching and learning activities of the Unit, rather than as separate formal assessment activities. The candidate will be allowed access to books, notes and on-line help while completing the task(s). The practical skills should be demonstrated in the context defined in the content statements (see Artificial Intelligence (Advanced Higher) Course content).

To gain success in this Outcome, the candidate must demonstrate practical skills in the following contexts:

- ◆ construction of a knowledge base of facts and rules in a declarative language
- ◆ implementation of list processing, involving recursion
- ◆ creation of queries to elicit information from a knowledge base
- ◆ testing a knowledge base
- ◆ creation of a rule-based system

Hard copy evidence should be provided of both the knowledge base and the rule-based system constructed.

A pro-forma observation checklist for Outcome 2 is provided in the National Assessment Bank materials.

All evidence for Outcome 2 should be gathered under ‘open-book’ conditions and must be retained by the centre. The assessment of this Unit is subject to moderation by SQA.

National Unit Specification: support notes (cont)

UNIT Artificial Intelligence (Advanced Higher)

The assessment of this Unit will require candidates to be familiar with, and able to correctly use in context, the following technical terms:

<p><i>Problem spaces and search</i></p> <ul style="list-style-type: none"> ◆ constraint ◆ goal state ◆ heuristic ◆ initial state ◆ production rule ◆ state space ◆ state space search ◆ transition 	<p><i>Prolog</i></p> <ul style="list-style-type: none"> ◆ backtracking ◆ compound term ◆ instantiation ◆ list ◆ recursive 	<p><i>General terms</i></p> <ul style="list-style-type: none"> ◆ ambiguity ◆ combinatorial explosion
<p><i>Frames</i></p> <ul style="list-style-type: none"> ◆ class ◆ default ◆ inheritance ◆ instance ◆ multiple inheritance ◆ slot 	<p><i>Computer vision</i></p> <ul style="list-style-type: none"> ◆ edge ◆ trihedral ◆ vertex ◆ Waltz algorithm 	<p><i>Natural language understanding</i></p> <ul style="list-style-type: none"> ◆ discourse ◆ grammar ◆ parse ◆ pragmatic ◆ semantic ◆ syntactic
<p><i>Robotics</i></p> <ul style="list-style-type: none"> ◆ pre-condition ◆ post-condition 	<p><i>Machine learning</i></p> <ul style="list-style-type: none"> ◆ decision tree ◆ inductive ◆ rote learning 	

Note that this list is not exhaustive: in particular, there are some terms in everyday use which also are applied to aspects of artificial intelligence, and which are not reproduced here, eg ‘analogy’ (as a learning method in machine learning).

CANDIDATES WITH ADDITIONAL SUPPORT NEEDS

This Unit Specification is intended to ensure that there are no artificial barriers to learning or assessment. The additional support needs of individual candidates should be taken into account when planning learning experiences, selecting assessment instruments or considering alternative Outcomes for Units. For information on these, please refer to the document *Guidance on Assessment Arrangements for Candidates with Disabilities and/or Additional Support Needs* (SQA, 2004).

National Unit Specification: general information

UNIT **Computer Networking (Advanced Higher)**

NUMBER DF30 13

COURSE Computing (Advanced Higher)

SUMMARY

This Unit is designed to develop knowledge and understanding of the principles of networking and practical skills related to networking through the use of contemporary hardware and software. This knowledge, understanding and practical skill may then be applied by the candidate to solve practical problems related to networking.

It is designed as an option for candidates undertaking the Advanced Higher Computing Course, but it is also suitable for anyone wishing to extend and deepen their experience of computer networking beyond Higher level.

OUTCOMES

1. Demonstrate knowledge and understanding of a range of facts, ideas and terminology relevant to the principles, features and purposes of networking.
2. Demonstrate practical skills in the context of networking using contemporary hardware and software.

RECOMMENDED ENTRY

While entry is at the discretion of the centre, candidates would normally be expected to have attained one of the following, or equivalent:

- ◆ *Computer Networking* (Higher)Unit
- ◆ Higher Computing

Administrative Information

Superclass: CB

Publication date: April 2005

Source: Scottish Qualifications Authority

Version: 01

© Scottish Qualifications Authority 2005

This publication 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 this unit specification can be purchased from the Scottish Qualifications Authority. The cost for each unit specification is £2.50. (A handling charge of £1.95 will apply to all orders for priced items.)

National Unit Specification: general information (cont)

UNIT Computer Networking (Advanced Higher)

CREDIT VALUE

1 credit at Advanced Higher (8 SCQF credit points at SCQF level 7)*

**SCQF credit points are used to allocate credit to qualifications in the Scottish Credit and Qualifications Framework (SCQF). Each qualification in the Framework is allocated a number of SCQF credit points at an SCQF level. There are 12 SCQF levels, ranging from Access 1 to Doctorates.*

CORE SKILLS

There is no automatic certification of Core Skills or Core Skills components in this Unit.

National Unit Specification: statement of standards

UNIT Computer Networking (Advanced Higher)

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

OUTCOME 1

Demonstrate knowledge and understanding of a range of facts, ideas and terminology relevant to the principles, features and purposes of networking.

Performance criteria

- (a) A wide range of advanced computing terminology is used appropriately.
- (b) Technically accurate descriptions and explanations are related to familiar and unfamiliar contexts.
- (c) Conclusions, predictions and generalisations are made from knowledge and understanding.

Evidence requirements

Written or oral evidence that the candidate can describe and explain the principles, features and purposes of networking accurately and concisely. Evidence should be obtained using questions in a closed-book test, under supervision, lasting no more than 45 minutes. The test must sample across the range of content (see Computing (Advanced Higher) Course content) in each of the following areas:

- ◆ network protocols
- ◆ network applications
- ◆ network security
- ◆ data transmission.

(The content statements are also reproduced for convenience as a table in the support notes for this Unit).

The standard to be applied is illustrated in the National Assessment Bank items available for this Unit.

If a centre wishes to design its own assessments for this Unit, they should be of a comparable standard.

National Unit Specification: statement of standards (cont)

UNIT Computer Networking (Advanced Higher)

OUTCOME 2

Demonstrate practical skills in the context of networking using contemporary hardware and software.

Performance criteria

- (a) A wide range of appropriate hardware is used effectively and efficiently.
- (b) A wide range of features of software is used effectively and efficiently.
- (c) Practical tasks are planned and organised independently.
- (d) Practical tasks are undertaken in an appropriate range of familiar and unfamiliar contexts.

Evidence requirements

Observation checklist showing that the candidate has demonstrated practical skills at an appropriate level in **two** of the following contexts:

- ◆ troubleshooting a network using the ping or trace route utilities
- ◆ creation of firewall rules to prevent unauthorised access
- ◆ creation of a complex web page using HTML

Hard copy evidence should be provided for **one** of these activities.

These practical skills may be demonstrated in a single extended task, or in a number of smaller tasks.

The practical skills should be demonstrated in the context defined in the content statements (see Advanced Higher Course content).

The candidate will be allowed access to books, notes and on-line help while completing the tasks.

(The content statements are also reproduced for convenience as a table in the support notes for this Unit)

The standard to be applied is illustrated in the National Assessment Bank items available for this Unit.

If a centre wishes to design its own assessments for this Unit, they should be of a comparable standard.

National Unit Specification: support notes

UNIT Computer Networking (Advanced Higher)

This part of the Unit specification is offered as guidance. The support notes are not mandatory.

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

GUIDANCE ON THE CONTENT AND CONTEXT FOR THIS UNIT

The content for this Unit is detailed below (and also in the National Course Specifications: Course details.)

Content statements in the left-hand column describe the content covered in the corresponding Unit at Higher level, and are included here to clarify the context for the new learning for this Unit. They indicate the prior learning required by the candidate before undertaking new learning within this Unit.

Content in the right-hand column is the new content for this Unit.

Content Statements: Network protocols	
<i>Higher</i>	Advanced Higher
	Explanation of the need for organisations enforcing standards including ISO and IEEE.
<i>Name and description of the seven layers of the OSI model.</i>	Description of mapping TCP/IP layers to OSI model layers.
<i>Brief explanation of the purpose of common protocols (TELNET, HTTP, SMTP and FTP).</i>	Explanation of the purpose of common protocols (SMTP, POP and MIME).
<i>Description of an IP address:</i> <ul style="list-style-type: none">◆ <i>structure: 4 octets</i>◆ <i>classes: ABCD</i>◆ <i>limitations</i>	Description of CIDR and binary subnet masks. Description of Trace route and Ping in terms of troubleshooting.
<i>Description of name services (name resolution); DNS (domain names, host name resolution).</i>	

National Unit Specification: support notes (cont)

UNIT Computer Networking (Advanced Higher)

Content Statements: Network Applications	
<i>Higher</i>	Advanced Higher
	<p>Description of the parts of an e-mail message; header (recipients address and other info) and body (containing the message to be sent). Brief description of sending and receiving e-mail including the role of SMTP:</p> <ul style="list-style-type: none"> ◆ connection setup ◆ mail transfer ◆ connection termination
<p><i>Description of a web page using HTML tags (header, body, title, style, font size, alignment and section headers).</i></p>	<p>Description of a web page using HTML tags (start, header, body, title, style, font size, alignment, section headers, colours and hyperlinks). Combination of tags to create a single line of code.</p>
	<p>Description of the process of requesting a web page by a client from a server and its transfer using HTTP from a server to a client:</p> <ul style="list-style-type: none"> ◆ HTTP (overview, types of connections-end to end TCP connection and not end to end TCP connections, three forms of intermediate-proxy, gateway and tunnel) ◆ types of data transmitted ◆ messages (request from client to server and response from server to client, fields-request line, response line, general header, request header, response header, entity header, entity body).
<p><i>Explanation of the advantages and disadvantages of browsers and microbrowsers for use with wireless data (WAP). Description of a web page using WML tags (wireless markup language).</i></p>	<p>Description of commonly used plug-ins to enhance browser functionality for portable documents, multimedia elements and streaming audio/video, naming currently used examples. Description and uses of Java applets and ActiveX.</p>
<p><i>Description of the methods used by search engines to build its indexes (spiders, meta-search engines).</i></p>	
<p><i>Description of the advantages of e-commerce. Implication of fraud in e-sales payment and how it is overcome.</i></p>	<p>Description of a video telephone call and its technical implications (hardware, software, data transmission and data compression):</p> <ul style="list-style-type: none"> ◆ video telephony in the context of teleconferencing (quality of video-dpi, sample rate) ◆ need for compression of video signal

National Unit Specification: support notes (cont)

UNIT Computer Networking (Advanced Higher)

<i>Description of the implications of the Regulation of Investigatory Powers Act 2000.</i>	
<i>Description of the social implications of networks; information rich and information poor, the family, the community and employment.</i> <i>Description of the ethical implications of networks; personal privacy and censorship.</i>	

Content Statements: Network security	
<i>Higher</i>	<i>Advanced Higher</i>
<p><i>Description of security measures:</i></p> <ul style="list-style-type: none"> ◆ <i>user access rights to data-file and folder permissions</i> ◆ <i>user access rights to hardware</i> 	
<p><i>Description of computer and network security requirements (confidentiality, data integrity and availability).</i></p> <p><i>Description of threats to network security in terms of passive (monitoring of transmission) and active (modification of the data stream or the creation of a false stream) attacks.</i></p>	<p>Description of the following methods for network and communication security:</p> <ul style="list-style-type: none"> ◆ <i>conventional encryption (plaintext, encryption algorithm, secret key, ciphertext and decryption algorithm)</i> ◆ <i>message authentication (using conventional encryption and without message encryption)</i> ◆ <i>public-key encryption and digital signatures (plaintext, encryption algorithm, public and private key, ciphertext and decryption algorithm)</i> ◆ <i>Internet architecture security</i>
<p><i>Description of the denial of service attack:</i></p> <ul style="list-style-type: none"> ◆ <i>effect; disruption or denial of services to legitimate users.</i> ◆ <i>costs of attack; system downtime, lost revenue and labour involved in identifying and reacting to an attack.</i> ◆ <i>intent; malicious, personal or political.</i> ◆ <i>types of attacks; bandwidth consumption, resource starvation, programming flaws and routing and DNS attacks.</i> 	<p>Description of the generic denial of service attack and countermeasures taken:</p> <ul style="list-style-type: none"> ◆ <i>Smurf: (attacker sends spoofed ICMP ECHO packet to broadcast address of a network, amplification of attack, bandwidth consumption prevention: disable directed broadcast functionality of border router, configure operating system to silently discard broadcast ICMP ECHO packets)</i> ◆ <i>SYN Flood: (attacker sends SYN packet from spoofed address, recipient sends SYN/ACK packet to spoofed address, recipient does not receive ACK from spoofed address and connection remains until timed out; consequence usually a server is taken out: prevention: increase size of connection queue, decrease connection established timeout period, employ vendor specific patches)</i>

National Unit Specification: support notes (cont)

UNIT Computer Networking (Advanced Higher)

<i>Higher</i>	Advanced Higher
	<ul style="list-style-type: none"> ◆ DNS: (attacker can try to convince a target nameserver to cache information that maps to a nonexistent IP address effectively denying that service; prevention: upgrade to latest version of BIND)
<i>Comparison of Internet content filtering methods: firewalls, Internet filtering software and walled gardens.</i>	
<i>Description of how a firewall can protect a LAN with an internet connection from outside attacks.</i>	Description of a few simple firewall rules used to protect a LAN with an Internet connection from outside attacks.
<p><i>Description of disaster avoidance:</i></p> <ul style="list-style-type: none"> ◆ use of anti-virus software ◆ use of fault tolerance components ◆ use of uninterrupted power supply. ◆ regular maintenance <p><i>Description of backup strategy:</i></p> <ul style="list-style-type: none"> ◆ backup server ◆ mirror disks ◆ tape ◆ backup schedule 	Description of types of backup; full, incremental and differential.

Content Statements: Data transmission	
<i>Higher</i>	Advanced Higher
<i>Description of synchronous and asynchronous data transmission.</i>	Comparison of bandwidth of different transmission systems; UTP, co-axial, fibre and radio waves.
<i>Description of error checking in data transmission (parity and CRC).</i>	
<i>Description of the process of transmitting data over a network using TCP/IP.</i>	
<i>Description of CSMA/CD and its implications for network performance.</i>	
<i>Description of network switching (circuit and packet switching) and its implications for network performance.</i>	
<p><i>Description of the application of modern wireless communication methods:</i></p> <ul style="list-style-type: none"> ◆ WPAN-connect mobile phones, mobile computers and other portable handheld devices ◆ wireless LAN-connecting a mobile LAN ◆ wireless WAN-connection in rural and heavily built-up areas 	<p>Explanation of the advantages, disadvantages and characteristics (data transfer rate, range and frequency) of modern wireless communication methods.</p> <p>Description of data standards of modern wireless communication methods (802.11a, 802.11b, 802.11g, bluetooth, HiperLAN2 and Ultrawideband).</p> <p>Description of the security and performance issues of modern wireless communication methods (WPAN and wireless LAN).</p>

National Unit Specification: support notes (cont)

UNIT Computer Networking (Advanced Higher)

<i>Higher</i>	<i>Advanced Higher</i>
<i>Description of the speed and bandwidth of the types of internet connections (dialup, cable modem, leased line, ISDN and ADSL). Explanation of which type of connection would be most appropriate in a given context.</i>	Description of remote access: <ul style="list-style-type: none"> ◆ dial-up protocols; SLIP and PPP ◆ virtual private network protocols; PPTP and L2TP
<i>Description of function of network interface card. Explanation for the need of a MAC address when transmitting data over a network.</i>	

List of abbreviations:

Higher

ADSL	Asymmetric Digital Subscriber Line
CRC	Cyclic Redundancy Check
CSMA/CD	Carrier Sense Multiple Access/ Collision Detection
DNS	Domain Name System
FTP	File Transfer Protocol
HTML	HyperText Mark-up Language
HTTP	HyperText Transfer Protocol
ISDN	Integrated Services Digital Network
MAC	Media Access Control
TELNET	Standard network virtual terminal protocol
WAP	Wireless Application Protocol
WML	Wireless Markup Language
WPAN	Wireless Personal Area Network
WAN	Wide Area Network

Advanced Higher

BIND	Berkeley Internet Name Domain
CIDR	Classless Inter-Domain Routing
ICMP	Internet Control Message Protocol
IEEE	Institute of Electronic and Electrical Engineers
ISO	International Standards Organisation
LAN	Local Area Network
L2TP	Layer 2 Tunnelling Protocol
MIME	Multi-purpose Internet Mail Extender
OSI	Open Systems Interchange
PING	Packet Internet Groper
POP	Post Office Protocol
PPP	Point to Point Protocol
PPTP	Point to Point Tunnelling Protocol
SLIP	Serial Line Internet Protocol
SMTP	Simple Mail Transport Protocol
TCP/IP	Transmission Control Protocol/ Internet Protocol
UTP	Unshielded Twisted Pair
WPAN	Wireless Personal Area Network

National Unit Specification: support notes (cont)

UNIT Computer Networking (Advanced Higher)

GUIDANCE ON LEARNING AND TEACHING APPROACHES FOR THIS UNIT

Candidates will require individual access to appropriate computer hardware and software throughout this Unit. While the learning may be achieved in the context of one computer system, students will benefit from having some experience of alternative operating systems.

The two Outcomes should be delivered in an integrated way rather than sequentially. For Outcome 2, the practical activities should be taught and used to illustrate and exemplify the knowledge and understanding required for Outcome 1.

Candidates who have completed the *Computer Networking* Unit at Higher level should already have covered the content listed in the left-hand column of the content grids, but may need to revise this material before progressing to the right-hand column.

The amount of time spent on each area of content will vary depending on the teaching methodology used and the ability and prior experience of the candidates. However, the following times are suggested as a rough guide:

network protocols	6 hours
network applications	10 hours
network security	10 hours
data transmission	10 hours

1½ hours should be set aside to:

- ◆ administer the Outcome 1 test
- ◆ gather evidence for Outcome 2

A further 2 ½ hours is allowed for remediation and re-assessment if required.

If the Unit is delivered as part of a Course, the Course documentation will provide further information on teaching and learning in a Course context, including the identification of a number of ‘themes’ to facilitate holistic learning across the Course.

GUIDANCE ON APPROACHES TO ASSESSMENT FOR THIS UNIT

National Assessment Bank tests have been created specifically to assess Outcome 1 of the Unit. This assessment consists of a closed-book test, and must be conducted under supervision. In order to gain success in this Outcome, the candidate must achieve at least the cut-off score for the test. If the centre wishes to design its own assessments for this Unit, they should be of a comparable standard.

Outcome 2 requires the candidate to demonstrate practical skills while using contemporary hardware and software. These practical skills may be demonstrated in a single extended task or a number of relatively small tasks, undertaken by the candidate during the teaching and learning activities of the Unit, rather than as separate formal assessment activities. The candidate will be allowed access to books, notes and on-line help while completing the task(s).

National Unit Specification: support notes (cont)

UNIT Computer Networking (Advanced Higher)

To gain success in this Outcome, the candidate must demonstrate practical skills in **two** of the following contexts as defined in the content statements (see Computing (Advanced Higher) Course content):

- ◆ troubleshooting a network using the ping or trace route utilities
- ◆ creation of firewall rules to prevent unauthorised access
- ◆ creation of a complex web page using HTML

Hard copy evidence should be provided for **one** of these activities. Note that this evidence need not be a formal report; it could consist of a printout or a screen shot from any of the practical activities.

A pro-forma observation checklist for Outcome 2 is provided in the National Assessment Bank materials.

All evidence must be retained by the centre. The assessment of this Unit is subject to moderation by SQA.

CANDIDATES WITH ADDITIONAL SUPPORT NEEDS

This Unit Specification is intended to ensure that there are no artificial barriers to learning or assessment. The additional support needs of individual candidates should be taken into account when planning learning experiences, selecting assessment instruments or considering alternative Outcomes for Units. For information on these, please refer to the document *Guidance on Assessment Arrangements for Candidates with Disabilities and/or Additional Support Needs* (SQA, 2004).

National Unit Specification: general information

UNIT **Computer Architecture (Advanced Higher)**

NUMBER DM44 13

COURSE Computing (Advanced Higher)

SUMMARY

This Unit is designed to develop knowledge and understanding of the principles of computer architecture and provides an opportunity to apply this through the use of contemporary hardware and software. The overall theme of the Unit is the relationship between design and performance. This knowledge, understanding and related practical skills may then be applied by the candidate to solve practical problems related to computer systems.

It is designed as an option for candidates undertaking the Advanced Higher Computing Course, but is also suitable for anyone wishing to extend and deepen their experience of computer systems beyond Higher level.

OUTCOMES

1. Demonstrate knowledge and understanding of the main theories, concepts and principles relevant to the process of designing computer systems to maximise performance.
2. Demonstrate practical skills in the context of computer architecture using contemporary hardware and software.

RECOMMENDED ENTRY

While entry is at the discretion of the centre, candidates would normally be expected to have attained one of the following, or equivalent:

- ◆ *Computer Systems (Higher) Unit*
- ◆ Higher Computing

Administrative Information

Superclass: CA

Publication date: April 2005

Source: Scottish Qualifications Authority

Version: 01

© Scottish Qualifications Authority 2005

This publication 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 this unit specification can be purchased from the Scottish Qualifications Authority. The cost for each unit specification is £2.50. (A handling charge of £1.95 will apply to all orders for priced items.)

National Unit Specification: general information (cont)

UNIT Computer Architecture (Advanced Higher)

CREDIT VALUE

1 credit at Advanced Higher (8 SCQF points at SCQF level 7).*

*SCQF credit points are used to allocate credit to qualifications in the Scottish Credit and Qualifications Framework (SCQF). Each qualification in the Framework is allocated a number of SCQF credit points at an SCQF level. There are 12 SCQF levels, ranging from Access 1 to Doctorates.

CORE SKILLS

There is no automatic certification of Core Skills or Core Skills components in this Unit.

National Unit Specification: statement of standards

UNIT Computer Architecture (Advanced Higher)

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

OUTCOME 1

Demonstrate knowledge and understanding of the main theories, concepts and principles relevant to the process of designing computer systems to maximise performance.

Performance criteria

- (a) A broad range of computing terminology is used appropriately.
- (b) Descriptions and explanations are related to familiar and unfamiliar contexts.
- (c) Conclusions, predictions and generalisations are made from knowledge and understanding.

Evidence requirements

Written or oral evidence that the candidate can describe and explain the principles and features relevant to the process of designing computer systems to maximise performance accurately and concisely. Evidence should be obtained using questions in a closed-book test, under supervision, lasting no more than 45 minutes. The test must sample across the range of the content (see Computing (Advanced Higher) Course content) in each of the following areas:

- ◆ computer structure
- ◆ processor structure
- ◆ processor development
- ◆ operating systems.

(The content statements are also reproduced for convenience as a table in the support notes for this Unit).

The standard to be applied is illustrated in the National Assessment Bank items available for this Unit.

If a centre wishes to design its own assessments for this Unit, they should be of a comparable standard.

National Unit Specification: statement of standards (cont)

UNIT Computer Architecture (Advanced Higher)

OUTCOME 2

Demonstrate practical skills in the context of computer systems using contemporary hardware and software.

Performance criteria

- (a) A range of features of hardware is used effectively and efficiently.
- (b) An appropriate range of features of software is used effectively and efficiently.
- (c) Practical tasks are planned and organized independently.
- (d) Practical tasks are undertaken in an appropriate range of non-routine contexts.

Evidence requirements

Observation checklist showing that the candidate has demonstrated the following practical skills in the context and at a level defined by the content statements (see Computing (Advanced Higher) Course content):

- ◆ comparing the facilities of two operating systems
- ◆ implementing simple assembly language instructions (using an assembler or an emulator).

A brief report on the comparison of two operating systems should be provided by the candidate.

These practical skills may all be demonstrated in a single extended task, or in of a number of smaller tasks.

The practical skills should be demonstrated in the context defined in the content statements (see Computing (Advanced Higher) Course content).

The candidate will be allowed access to books, notes and on-line help while completing the task(s).

(The content statements are also reproduced for convenience as a table in the support notes for this Unit).

The standard to be applied is illustrated in the National Assessment Bank items available for this Unit.

If a centre wishes to design its own assessments for this Unit, they should be of a comparable standard.

National Unit Specification: support notes

UNIT Computer Architecture (Advanced Higher)

This part of the Unit specification is offered as guidance. The support notes are not mandatory.

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

GUIDANCE ON THE CONTENT AND CONTEXT FOR THIS UNIT

The content for this Unit is detailed below (and also in the National Course Specifications: Course details).

Content Statements: Computer structure
Detailed description of processor and registers including MAR, MDR, IR, PC and general purpose registers. Description of relationship between buses and processor registers. Description of the fetch-execute cycle in terms of buses and registers.
Description of the following types of computer memory: <ul style="list-style-type: none">◆ internal memory (registers, cache , main memory)◆ external memory (magnetic disk, CD-ROM, CD-RW, DVD-ROM, rewritable DVD, tape) Description of these memory technologies in a hierarchy, using the following factors: <ul style="list-style-type: none">◆ decreasing cost per bit◆ increasing capacity◆ increasing access time◆ decreasing frequency of access Explanation of the importance of these factors when designing a system for performance.
Description of the structured use of cache memory to improve processor performance referring to the use of level 1 and level 2 cache as well as the use of static RAM.
Description of how memory interleaving operates. Description of how memory interleaving can improve system performance by enabling multiple memory accesses simultaneously.
Description of how direct memory access can improve system performance.
Description of the effect on system performance of increasing clock speeds and increasing the width of data buses with reference to 8 bit, 16 bit, 32 bit and 64 bit microprocessors.
Description of PCI and PCI-X buses in terms of the following characteristics: throughput described as bits per second, width, multipoint topology, function. Description of the importance of bus throughput for system performance.
Content Statements: Processor structure
Description and exemplification of the structure of assembly language instructions as op-code and operand.
Description and exemplification of the classification of assembly language instructions using the following categories: data transfer, arithmetic, logical, shift and rotate, branch.
Description of the following key features which distinguish RISC from CISC: a limited and simple instruction set, the use of register oriented instructions with limited memory access, the use of limited addressing modes and a large bank of registers.
Description of the performance gain derived from the use of SIMD (single instruction multiple data) instructions with reference to multimedia processing operations.

National Unit Specification: support notes (cont)

UNIT Computer Architecture (Advanced Higher)

Description of how the following techniques can be used to optimise the instruction and data stream:

- ◆ branch prediction
- ◆ data flow analysis
- ◆ speculative loading of data
- ◆ speculative execution of instructions
- ◆ predication

Description of how pipelining operates and how it can improve system performance.

Description of possible problems with pipelining caused by branch instructions and by instructions of different lengths.

Descriptions of how superscalar processing operates and how it can improve system performance.

Content Statements: Processor development

Description of the evolution of the following microprocessor architectures: the Power PC series, the Intel X86 series and the Intel IA-64 in terms, where appropriate, of the following features and techniques:

- ◆ increasing clock speeds
- ◆ data bus widths
- ◆ pipelining
- ◆ superscalar processing
- ◆ branch prediction
- ◆ speculative loading of data and executing of instructions
- ◆ predication
- ◆ the number and function of registers used
- ◆ SIMD
- ◆ RISC
- ◆ CISC

Explanation of the relationship between these developments and system performance.

Description of how parallel computers function referring to their use of:

- ◆ local (cache) as well as main memory
- ◆ pipelining
- ◆ local pathways and packet switching to achieve communication between CPUs.

Description of the performance benefits of parallel computers.

Content Statements: Operating systems

Description of the following techniques used by operating systems to manage memory:

- ◆ variable partitioning of memory
- ◆ use of best-fit, worst fit and first fit algorithms for the allocation of memory.

Comparison of the operation of best fit, worst fit and first fit algorithms in terms of efficient use of memory.

Description of the relationship between the size of data blocks used in memory allocation and access speed.

Description of the need for operating systems to schedule programs in a multitasking system:

Comparison of the following types of pre-emptive scheduling in terms of their effect on system performance:

Round robin scheduling, multi-level feedback queue.

National Unit Specification: support notes (cont)

UNIT Computer Architecture (Advanced Higher)

Description of the use of direct memory access to manage input/output data transfers in order to improve system performance.
Description of the key function of the file management system as mapping between the logical view of files and their physical location. Description of contiguous and non-contiguous methods of allocation of files to available storage space.
Description of the way in which the trend towards designing GUI based on the design principle of 'convenience for the user' leads to: <ul style="list-style-type: none">◆ increasing software complexity◆ more demands on system resources (processor and memory)◆ a consequent burden on system performance Illustration of the demands a GUI makes on the system resources by describing the steps involved in processing a simple operation such as a mouse click. Comparison with the demands a CLI makes on the system.
Description of the trend to expand the role of operating systems to include services and provide capabilities that were formerly within applications themselves. Description of those services as: <ul style="list-style-type: none">◆ providing a standard look and feel for applications◆ simplifying and extending graphic capabilities of application programs◆ improving the capability of programs to communicate and pass data◆ the capability of launching one application inside another. Description of the use of libraries of objects that applications call as required. Description of techniques used by OS to support communication between applications and document embedding.
Description of procedures used to control user access in both multi-user and single user systems. Description of the purpose of file attributes. Description of backup facilities.
Exemplification and comparison of the features of two specific operating systems.

GUIDANCE ON LEARNING AND TEACHING APPROACHES FOR THIS UNIT

Candidates will require individual access to appropriate computer hardware and software throughout this Unit.

The two Outcomes should be delivered in an integrated way rather than sequentially. For Outcome 2, the practical activities should be taught and used to illustrate and exemplify the knowledge and understanding required for Outcome 1. The underlying theme of 'design for performance' should be illustrated and exemplified throughout the Unit.

Although most of the work of this Unit can be completed using a single computer system, candidates will require some access to an alternative operating system. They will also require a level of access which allows making changes to operating system parameters. Practical work involving assembly language could be undertaken using a real assembler or using a simulation program.

National Unit Specification: support notes (cont)

UNIT Computer Architecture (Advanced Higher)

The amount of time spent on each area of content will vary depending on the teaching methodology used and the ability and prior experience of the candidates. However, the following times may be used as a rough guide:

computer structure	8 hours
processor structure	8 hours
processor development	6 hours
operating systems	14 hours

1½ hours should be set aside to:

- ◆ administer the Outcome 1 test
- ◆ gather evidence for Outcome 2.

A further 2½ hours is allowed for remediation and re-assessment if required.

If the Unit is delivered as part of a Course, the Course documentation will provide further information on teaching and learning in a Course context, including the identification of a number of ‘themes’ to facilitate holistic learning across the Course.

GUIDANCE ON APPROACHES TO ASSESSMENT FOR THIS UNIT

National Assessment Bank tests have been created specifically to assess Outcome 1 of the Unit. This assessment consists of a closed-book test, and must be conducted under supervision. In order to gain success in this Outcome, the candidate must achieve at least the cut-off score for the test. If a centre wishes to design its own assessments for this Unit, they should be of a comparable standard.

Outcome 2 requires the candidate to demonstrate practical skills while using contemporary hardware and software. These practical skills may be demonstrated in a single extended task or a number of relatively small tasks. The task(s) will normally be undertaken by the candidate as part of the teaching and learning activities of the Unit, rather than as separate formal assessment activities. The candidate will be allowed access to books, notes and on-line help while completing the task(s). The practical skills should be demonstrated in the context defined in the content statements (see Computing (Advanced Higher) Course content).

To gain success in this Outcome, the candidate must demonstrate practical skills at an appropriate level in both of the following contexts:

- ◆ comparing the facilities of two operating systems
- ◆ implementing simple assembly language instructions (using an assembler or an emulator).

A brief report on the comparison of two operating systems should be provided by the candidate.

A pro-forma observation checklist for Outcome 2 is provided in the National Assessment Bank materials.

All evidence must be retained by the centre. The assessment of this Unit is subject to moderation by SQA.

National Unit Specification: support notes (cont)

UNIT Computer Architecture (Advanced Higher)

CANDIDATES WITH ADDITIONAL SUPPORT NEEDS

This Unit Specification is intended to ensure that there are no artificial barriers to learning or assessment. The additional support needs of individual candidates should be taken into account when planning learning experiences, selecting assessment instruments or considering alternative Outcomes for Units. For information on these, please refer to the document *Guidance on Assessment Arrangements for Candidates with Disabilities and/or Additional Support Needs* (SQA, 2004).