Higher National unit specification

General information

Unit title: Computer Programming: Applied Mathematics (SCQF level 7)

Unit code: HH3H 34

Superclass: CC

Publication date: November 2016

Source: Scottish Qualifications Authority

Version: 02

Unit purpose

The purpose of this unit is to permit learners to create numerical, statistical and mathematical programming functions using a high level language. The unit aims to develop learners' coding skills and also their underpinning knowledge of numbers, statistics and mathematics. This knowledge will be developed, and then applied, in a programming context. The choice of programming language is optional.

A significant part of this unit relates to developing this underpinning knowledge. This will include: number types (including complex numbers), number systems (including hexadecimal), scientific notation, factorials, measures of dispersion (including mean absolute deviation), probability, Boolean algebra, set theory and trigonometry.

This knowledge will then be applied in a programming context to develop a library of numerical/statistical/mathematical routines using a contemporary high level language (such as Python).

This is a **non-specialist** unit that can be used within a range of Computing awards. It is particularly well suited to learners who require to apply numerical methods in programming contexts, such as learners who wish to develop computer games or systems software.

At the completion of the unit, learners will have developed their knowledge and understanding of key numerical, statistical and mathematical techniques, and learned how to implement these techniques in a high level programming language.

Learners may progress to further studies in programming or mathematics or progress to a similar unit at a higher level (such as *Computer Programming: Applied Mathematics* (SCQF level 8)).

Higher National unit specification: General information (cont)

Unit title: Computer Programming: Applied Mathematics (SCQF level 7)

Outcomes

On successful completion of the Unit the learner will be able to:

1 Write numerical functions in a high level language.
2 Write statistical functions in a high level language.
3 Write mathematical functions in a high level language.

Credit points and level

1 Higher National unit credit at SCQF level 7: (8 SCQF credit points at SCQF level 7)

Recommended entry to the Unit

Access to this Unit is at the discretion of the centre. However, it is recommended that learners should have basic knowledge of mathematics and programming. This could be evidenced by possession of National Qualifications in Mathematics and Computing Science at SCQF levels 5 or 6, or National Units in these areas (at an appropriate level).

Core Skills

Achievement of this Unit gives automatic certification of the following Core Skills component:

Complete Core Skill	None
Core Skill component	Using Number at SCQF level 6 Critical Thinking at SCQF level 6

There are also opportunities to develop aspects of Core Skills which are highlighted in the Support Notes of this Unit specification.

Context for delivery

If this Unit is delivered as part of a Group Award, it is recommended that it should be taught and assessed within the subject area of the Group Award to which it contributes.

The Assessment Support Pack (ASP) for this Unit provides assessment and marking guidelines that exemplify the national standard for achievement. It is a valid, reliable and practicable assessment. Centres wishing to develop their own assessments should refer to the ASP to ensure a comparable standard. A list of existing ASPs is available to download from SQA's website **(http://www.sqa.org.uk/sqa/46233.2769.html)**.

Equality and inclusion

This Unit specification has been designed to ensure that there are no unnecessary barriers to learning or assessment. The individual needs of learners should be taken into account

when planning learning experiences, selecting assessment methods or considering alternative evidence.

Further advice can be found on our website **www.sqa.org.uk/assessmentarrangements**.

Higher National Unit specification: Statement of standards

Unit title: Computer Programming: Applied Mathematics (SCQF level 7)

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 SQA.

Where evidence for Outcomes is assessed on a sample basis, the whole of the content listed in the Knowledge and/or Skills section must be taught and available for assessment. Learners should not know in advance the items on which they will be assessed and different items should be sampled on each assessment occasion.

Outcome 1

Write numerical functions in a high level language.

Knowledge and/or Skills

- Number types (natural, whole, integer, rational, irrational, real, complex)
- Fundamental operations
- Operators and operator priority
- Percentages, fractions, and decimals
- Numbering systems (bases)
- Scientific notation
- Powers, roots and factorials
- Built-in numerical functions and library routines
- Numerical expression and function creation in a high level language

Outcome 2

Write statistical functions in a high level language.

Knowledge and/or Skills

- Measures of central tendency (including mean)
- Measures of dispersion (including mean absolute deviation)
- Probability (including combinations and permutations)
- Standard error
- Built-in statistical functions and library routines
- Statistical expression and function creation in a high level language

Higher National Unit specification: Statement of standards (cont)

Unit title: Computer Programming: Applied Mathematics (SCQF level 7)

Outcome 3

Write mathematical functions in a high level language.

Knowledge and/or Skills

- Elementary algebra
- Boolean algebra (including Boolean logic operators)
- Applications of Set theory (including set operations)
- Working with two dimensional geometry (shapes and points)
- Linear algebra (two dimensional vectors)
- Trigonometric functions (including sine, cosine, and arctangent)
- Built-in mathematical functions and library routines
- Mathematical expression and function creation in a high level language

Evidence Requirements for this Unit

Candidates will need to provide evidence to demonstrate their Knowledge and/or Skills across all Outcomes. Evidence is normally required for all of the Knowledge and Skills in every Outcome.

The Evidence Requirements for this unit will take the form of **evidence of cognitive competence** and **evidence of practical competence**. The evidence of cognitive competence will relate to candidates' underpinning knowledge of the numbers, statistics and mathematics (Outcomes 1, 2 and 3). The evidence of practical competence will relate to candidates' ability to create program functions that perform numerical, statistical and mathematical computations (Outcomes 1, 2 and 3).

This evidence must cover all of the Knowledge and Skills defined within this unit specification (but see the exception below). The amount of evidence should be the minimum consistent with assuring the assessor that the candidate possesses each specific Knowledge and/or Skill. Access to reference material is permissible.

The evidence of cognitive competence will relate to all of the knowledge and understanding contained within Outcomes 1, 2 and 3.

Sampling of the knowledge and understanding may be used when the evidence is in the form of a test of the candidate's knowledge and understanding of the defined Knowledge and/or Skills. The test must sample broadly and widely from the defined Knowledge and/or Skills, and must be unknown and unpredictable to the candidate. The evidence, in this case, must be produced under controlled conditions. Access to reference material is not appropriate if testing is used.

Higher National Unit specification: Statement of standards (cont)

Unit title: Computer Programming: Applied Mathematics (SCQF level 7)

The evidence of practical competence will relate to the practical aspects of Outcomes 1, 2 and 3. This evidence will take any appropriate form so long at it demonstrates competence in the creation of all, or a sample of, the program functions in a high level language. The program functions will cover a range of numerical, statistical and mathematical functions. The selection of the functions that will be demonstrated in code will be done by the candidate. However, the functions must be non-trivial and sample broadly and widely across all three Outcomes. This competence may be demonstrated in separate program functions or one (large) program that incorporates a selection of functions.

Evidence may be wholly or partly produced under controlled conditions. When evidence is produced in uncontrolled or loosely controlled conditions it must be authenticated. The Guide to Assessment provides further advice on methods of authentication. Evidence of authentication must be provided when any of the evidence is generated under loosely controlled conditions.

There are no time limitations on the production of evidence (unless testing is used). The evidence may be produced at any time during the life of the unit.

The Guidelines on Approaches to Assessment (see the Support Notes section of this specification) provides specific examples of instruments of assessment.

Higher National Unit Support Notes

Unit title: Computer Programming: Applied Mathematics (SCQF level 7)

Unit Support Notes are offered as guidance and 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

Unit purpose

This unit is intended to improve learners' understanding of how to implement mathematical techniques in a programming context.

The programming environment should be a high level language, and the learner is not expected to write code that performs the mathematical operations themselves, but rather to apply existing mathematical operations in order to solve programming problems. For example, students should not be expected to write their own code to perform a Taylor series expansion to work out sine values; but it would be reasonable to ask learners to make use of existing 'sin' and 'cos' functions in order to calculate the X and Y components of a vector. Most widely-used languages have appropriate function libraries available, such as C++, C#, Python, and Java. The unit is aimed at learners who already have at least some basic understanding of mathematics and particularly programming. They should not need to study programming concepts such as control structures, data structures, or objects.

Unit Outcomes

The three Outcomes should cover commonly used techniques in the three areas, focusing on mathematical functions and their practical applications as far as is feasible. The following outlines suggested topics for delivery within those areas.

Numerical functions

The knowledge and skills learned here are intended to provide a basis for a wide range of programming techniques. As such it starts from early principles, and not all of this knowledge is directly applicable in a realistic scenario on their own, but instead form a fundamental part of many later solutions.

- Number types:
  - Natural numbers/unsigned integers
  - Whole numbers/integers, including negative numbers
  - Rational numbers/ratio data types
  - Irrational numbers, eg approximate value of pi compared to recursive calculation
  - Real numbers/floating and fixed point numbers, including issues associated with accuracy
  - Complex numbers, eg their use in performing 2d rotation

- Fundamental operations:
  - Addition
  - Subtraction
  - Multiplication
  - Division (including modulus)
- Operators and operator priority (BODMAS/PEMDAS/PFEMDAS)
- Percentages, Fractions and decimals
- Numbering systems (bases):
  - Decimal
  - Binary
  - Hexadecimal
  - Octal
- Scientific notation
- Powers, roots and factorials
- Built-in numerical functions and library routines:
  - These will vary by language, but for example in C# you may wish to cover Abs(), Exp(), Pow(), and Sign().
- Expression and function composition and evaluation:
  - Using the return value of one function in the parameters of another function
  - It is suggested that this section is delivered and assessed holistically with the similarly-named section from the other Outcomes. It is also intrinsically linked to operator priority.

Statistical functions

This Outcome is focused on ways of measuring data in a given sample space, which has wide practical applications, from relatively simple analysis of averages, to stochastic (probability-based) functions which derive solutions to problems by repeated random tests.

- Measures of central tendency:
  - Mean
  - Median
  - Mode
- Measures of dispersion:
  - Range
  - Variance
  - Standard deviation
  - Mean absolute deviation
- Probability:
  - Sample space, events, and Outcomes
  - Odds (calculating the probability of given events)
  - Stochastic methods, eg The Monte Carlo method or Las Vegas Method
  - Combinations and permutations
- Standard error:
  - Sampling distribution
  - Standard error of the mean
  - Confidence intervals

- Built-in statistical functions and library routines:
  - These will vary by language, but for example in C# you would need to utilise a third-party library such as MethNet.Numerics.Statistics, which contains the functions Mean(), Variance(), and StandardDeviation().
- Expression and function composition and evaluation:
  - It is suggested that this section is delivered and assessed holistically with the similarly-named section from the other Outcomes.

Mathematical functions

This Outcome covers a number of important topics, each of which could have numerous applications, often overlapping. For example, in many cases, trigonometric functions are used in order to calculate vector components.

- Elementary algebra:
  - Variables
  - Constants
  - Exponents
  - Coefficients
  - Terms
  - Operators and operands
  - Reciprocals/Multiplicative inverse
- Boolean algebra:
  - Boolean logic operators:
    - AND
    - OR
    - NOT
    - XOR
  - Truth tables
  - Bitwise logic
- Set theory:
  - Set operations:
    - Union
    - Intersection
    - Difference
  - Implementing sets as data structures
- Working with two dimensional geometry:
  - Co-ordinate systems: Cartesian, Polar
  - 2D Geometrical points and shapes:
    - Points and vertexes
    - Lines/Rays/Line segments (midpoints, intersection, parallel, perpendicular)
    - Angles (acute, obtuse, right-angle)
    - Circles/Arcs (radius, diameter, circumference)
    - Polygons

  - Triangles (right-angle, equilateral, scalene, isosceles, hypotenuse)
  - Quads (Rectangle, rhombus, parallelogram, square)
- Linear algebra — 2d Vectors:
  - Using vectors in 2D space
  - Vector arithmetic (add, subtract, scalar multiplication, dot product)
  - Normalisation
- Trigonometric functions:
  - Use of degrees and radians
  - Understanding sin, cos and tan
  - Inverse trig functions, including arctan
  - Pythagoras theorem and calculating distance in vectors
- Built-in mathematical functions and library routines:
  - These will vary by language, but for example in C# you may wish to cover Sin(), Cos(), Atan2(), HashSet.Union(), etc. Third party libraries could be added to cover such functions as rectangle intersection, vector dot product, vector normalisation, and vector length.
- Expression and function composition and evaluation:
  - It is suggested that this section is delivered and assessed holistically with the similarly-named section from the other Outcomes.

Contextualisation

As far as is feasible, the problems presented to the students to solve should be contextualised by using relevant problems for the learner to solve using mathematical techniques.

For example, learners studying for a Computer Game Development Group Award could use the Monte Carlo method to investigate game balance in a card game, or use geometry for collision detection in a real-time game.

Students studying for Software Development might instead use sets for working with databases, or use geometry in Computer Aided Design, for example:

Progression and links

This unit provides underpinning knowledge for the unit *Computer Programming: Applied Mathematics* (level 8); and may also be suitable for other programming units which involve mathematics.

It is suitable for learners who intend to progress onto courses which continue to use mathematics in programming, such as an HND or university course in Software Development, Computer Game Development, or general Computer Science.

The unit focuses on vocational skill in applying mathematical techniques in programming environments, preparing learners for a career in programming where mathematics is often used, such as Statistical Analysis, Game Development, and so on. As such the theoretical aspects of mathematics such as formal proofs should be kept to a minimum when delivering this unit.

Higher National Unit Support Notes (cont)

Unit title: Computer Programming: Applied Mathematics (SCQF level 7)

This unit currently has no special recognition by professional bodies, and there are no National Occupational Standards specific to this area at this time.

Guidance on approaches to delivery of this Unit

It is suggested that the material be delivered in the order of the Outcomes and criteria. This is due to the fact that some areas of mathematics are interconnected, and learning in certain areas can be useful or even necessary before attempting later techniques. For example, understanding percentages, etc is likely to be necessary when calculating probabilities and odds.

However, that does not necessarily mean that the material needs to be assessed in that same order. For example, the basic numerical knowledge can easily be assessed later alongside more technical requirements which they underpin.

It is recommended that the unit be delivered using varied approaches to teaching and learning where possible. However due to the nature of the content and the wide range of material covered, opportunities for group activities may be limited and might only be feasible for formative work. Emphasis should be placed on practical tasks, particularly coding where possible. Although certain elements may require working with maths problems in a theoretical setting, this should be kept to a minimum, and a long series of paper-based maths problems would not be well suited to this unit.

Outcome 1 is fundamental and starts from first principles — it is suggested that no more than 20% of the learning and teaching time is devoted to it. Outcome 2 does involve more practical applications and is expected to take about 30% of the time, while Outcome 3 contains more topics and is expected to take approximately 50% of the learning and teaching time.

Guidance on approaches to assessment of this Unit

Evidence can be generated using different types of assessment. The following are suggestions only. There may be other methods that would be more suitable to candidates.

The assessment for this unit could comprise two assessments: one relating to knowledge and understanding (for evidence of cognitive competence) and one relating to practical abilities (for evidence of practical competence).

The assessment of knowledge and understanding could comprise a traditional test that assesses the learner's knowledge and understanding of the defined knowledge domain (across all Outcomes). This would be a timed, closed-book test of their knowledge of numerical, statistical and mathematical methods, with an appropriate pass mark (50% normally).

Higher National Unit Support Notes (cont)

Unit title: Computer Programming: Applied Mathematics (SCQF level 7)

The assessment of practical abilities could comprise a portfolio of programs/functions that demonstrate the learner's ability to write program functions that cover a range of numerical, statistical and mathematical functions. The programs (or program segments) could demonstrate that the learner can write code that involves operators and operator priority, percentages and fractions, converts between number systems, works out factorials, computes mean, media and mode, evaluates probability, works out standard deviation, performs algebraic manipulations, demonstrates Boolean algebra, and involves trigonometric functions. The selection of code segments to include in portfolios would be the decision of the learner. This portfolio would be generated throughout the life of the unit and produced under loosely controlled conditions. Because of this, some means of authentication would be used to verify that it is the learner's work (such as oral questioning).

Centres are reminded that prior verification of centre-devised assessments would help to ensure that the national standard is being met. Where learners experience a range of assessment methods, this helps them to develop different skills that should be transferable to work or further and higher education.

An example of one possible scheme for assessment of this unit is as follows, contextualised for delivery as part of a computer game qualification:

Cognitive evidence

Some of the Evidence Requirements do not easily lend themselves to practical assessments in a suitable context. As such, centres might decide to assess certain topics using a set of 30 multiple choice questions, or two sets of 15 multiple-choice questions, covering criteria from multiple Outcomes. For example:

Assessment 1:

- Number types: five questions regarding numeric data types, storage of real and rational numbers, accuracy of real and irrational numbers, effects of mixing data types in an expression, use of complex numbers, etc.
- Numbering systems: five questions regarding decimal, binary and hex conversion.
- Scientific notation: five questions regarding mantissa and exponent, normalized scientific notation, and orders of magnitude.

Assessment 2:

- Probability: five questions on combinations and permutations.
- Boolean algebra: five questions regarding Boolean logic operators, truth tables, and bitwise logic.
- Set theory: five questions regarding set union, intersection, difference, and set data structures in programming languages.

Higher National Unit Support Notes (cont)

Unit title: Computer Programming: Applied Mathematics (SCQF level 7)

Practical evidence

This could comprise a portfolio of projects containing code to solve a variety of programming problems using mathematical techniques. The portfolio would likely be based on code outlines provided by the tutor, which the learner would have to apply mathematical techniques to in order to achieve the stated goals. The use of mathematical functions in these solutions would fulfil the requirements to use 'built in functions and routines', and the 'expression and function composition and evaluation' is inherent in the proper handling of data in these solutions.

- In the interests of game balance, developers are analysing the value (in 'dps') of items dropped in a particular dungeon, so they can compare it to other dungeons and make changes to gameplay as appropriate. Given a data set showing dps for all items, write code to work out the average dps as mean, median and mode. Also, write code to display the range, variance, and standard deviation of these data points. Extend this code to run multiple sample collections, and calculate a standard error of the mean. (**Measures of central tendency, measures of dispersion, standard error, roots, fundamental operations, operator priority, elementary algebra)**
- In game design, one of the tasks of the 'combat designer' is to provide a suitable level of challenge and depth in a simulation with many variables at work. Given variables such as: a particular weapon is used, which does varying amounts of damage at different ranges; hitting the enemy in a critical area causes double damage; the chance of hitting the enemy and/or getting a critical hit depends on a given player character's Accuracy Rating; and that the player will be given a time limit which allows him a certain number of attacks; write a Monte Carlo simulation that will perform a large number of combat simulations and gather a data set to find the chance of defeating the enemy. Express this chance as a percentage, a fraction or ratio, and as a decimal p-value**. (Probability, percentages, fractions and decimals, fundamental operations, elementary algebra)**
- As a greatly simplified physics simulation, a top-down racing game uses vectors to represent the velocity of a car, and when the player turns the car, it maintains the same speed in the new direction. Write code that will calculate the magnitude of this vector, and use a trigonometric function to find the theta angle. Then, according to player input, turn the car by modifying the angle. 'Wrap' the angle so that it is in the range -π to +π, and use more trigonometric functions to calculate the new vector components. (**Vectors, trigonometry, fundamental operations, elementary algebra, powers and roots)**
- As part of the user interface, a game may need to determine whether the mouse has clicked on certain UI elements, based on their geometry on the screen. Given buttons in shapes such as rectangles and circles, make the game react by highlighting the shapes when the point under the mouse cursor is within the shape. (**2d geometry, fundamental operations, elementary algebra)**

It may well be possible to integrate assessment of certain criteria in this unit along with other Outcomes in this unit, or as part of other units. For example, if the learner produces a solution to a programming problem as part of a software development unit, which includes geometric functions in it, the assessor may accept that solution as part of the folio of practical applications for this unit.

Higher National Unit Support Notes (cont)

Unit title: Computer Programming: Applied Mathematics (SCQF level 7)

It is recommended that the practical tasks be undertaken in supervised conditions, to help ensure authenticity of learners' submissions. If the centre allows learners to work on programming problems unsupervised (eg at home, or in a student-led study area) then assessors should take additional steps to ensure the authenticity of the submission, such as asking the learner to explain parts of the code verbally and how the code solves the problem.

Opportunities for e-assessment

E-assessment may be appropriate for some assessments in this Unit. By e-assessment we mean assessment which is supported by Information and Communication Technology (ICT), such as e-testing or the use of e-portfolios or social software. Centres which wish to use e-assessment must ensure that the national standard is applied to all learner evidence and that conditions of assessment as specified in the Evidence Requirements are met, regardless of the mode of gathering evidence. The most up-to-date guidance on the use of e-assessment to support SQA's qualifications is available at **www.sqa.org.uk/e-assessment**.

Opportunities for developing Core and other essential skills

There is likely to be numerous opportunities to develop elements of the Core Skills of *Numeracy* in this unit, and possibly some elements of *Problem Solving*.

This Unit has the Using Number component of Numeracy and Critical Thinking component of Problem Solving embedded in it. This means that when learners achieve the Unit, their Core Skills profile will also be updated to show they have achieved Using Number at SCQF level 6 and Critical Thinking at SCQF level 6.

History of changes to Unit

Version	Description of change	Date
02	Core Skills Components Using Number and Critical Thinking at SCQF level 6 embedded.	21/02/17

General information for learners

Unit title: Computer Programming: Applied Mathematics (SCQF level 7)

This section will help you decide whether this is the Unit for you by explaining what the Unit is about, what you should know or be able to do before you start, what you will need to do during the unit and opportunities for further learning and employment.

This is an introductory unit into using mathematical techniques and algorithms in programming. Although it is assumed that you have some general understanding of programming, it is not necessary to begin this unit with any more than a very fundamental understanding of basic maths. It introduces fundamental concepts of mathematics as used in practical programming applications, and is suited to a number of computing courses, but may also be studied on its own by learners wishing to improve their programming and mathematics skills.

You will be presented with a variety of programming problems which you will be required to solve using the mathematical functions available. These solutions will form a complete or partial portfolio. Your tutor will inform you whether the portfolio should be complete, covering all the unit requirements, or whether it should be partial, with the remaining requirements being met by a written assessment.

The solutions will cover areas such as:

- Fundamental numeracy, from basic operations like adding and subtracting, to factorials and roots. The basics of data representation in computers will also be covered.
- Statistical functions, including techniques related to finding averages, distributions of data, and probability.
- General mathematical functions, in areas such as Boolean logic, set theory, geometry, vector mathematics, and trigonometry.

While solving these problems, you will develop elements of Core Skills in the areas of *Numeracy* and *Problem Solving*.

This Unit has the Using Number component of Numeracy and Critical Thinking component of Problem Solving embedded in it. This means that when you achieve the Unit, your Core Skills profile will also be updated to show you have achieved Using Number at SCQF level 6 and Critical Thinking at SCQF level 6.