Unit title:	Mathematical skills for computing
Unit number:	19
Level:	3
Credit value:	15
Guided learning hours:	60
Unit reference number:	L/505/5831

UNIT AIM AND PURPOSE

The purpose of this unit is to prepare learners for further study of computing or computer science by equipping them with core mathematical and computational thinking skills that are needed by those working in computer science. These include skills in number and algebra, in propositional logic, basic set and graph theory, combinatorics and co-ordinate graphics.

As a stand-alone computing unit, this unit would be suitable for learners on a Level 3 program planning to progress onto higher education or other further qualifications in computer science, or learners on a computer science Level 4 or 5 programme who need a stronger mathematical foundation.

The focus throughout the unit is on the use of these skills to solve practical problems, rather than learning mathematical methods out of context. In particular, learners will develop the skill of abstraction throughout the unit – the ability to extract the essential information from the context and represent it in a way that enables them to derive a solution.

LEARNING OUTCOMES AND ASSESSMENT CRITERIA

Learning Outcome (LO) The learner will:	**Pass** The assessment criteria are the pass requirements for this unit. The learner can:	**Merit** To achieve a merit the evidence must show that, in addition to the pass criteria, the learner is able to:	**Distinction** To achieve a distinction the evidence must show that, in addition to the pass and merit criteria, the learner is able to:
1 Be able to solve numerical problems using algebraic abstraction	P1 state which values in a problem are integers, real, discrete or continuous		
	P2 evaluate arithmetic expressions involving addition, subtraction, multiplication, division and rounding	M1 explain different types of rounding and justify the type(s) used	
	P3 express numerical problems algebraically, using variables to represent unknowns	M2 solve problems involving integer division or modulo arithmetic and unknown values	
	P4 solve linear equations by algebraic manipulation	M3 solve linear inequalities by determining whether the solution to the corresponding equation is an included/excluded minimum/maximum	D1 use iteration to find a solution to a non-linear equation
2 Be able to use propositional logic to represent and solve problems	P5 create truth tables for propositional logic expressions involving at least three variables	M4 derive propositional expressions to describe a system from truth tables expressing inputs and output conditions of the system	D2 explain how propositional logic expressions can be simplified using the properties of propositional calculus, or other methods

Learning Outcome (LO) The learner will:	**Pass** The assessment criteria are the pass requirements for this unit. The learner can:	**Merit** To achieve a merit the evidence must show that, in addition to the pass criteria, the learner is able to:	**Distinction** To achieve a distinction the evidence must show that, in addition to the pass and merit criteria, the learner is able to:
3 Be able to solve problems involving collections of objects	P6 use sets to solve a problem		
	P7 represent a problem as a graph, stating what the vertices and edges represent	M5 explain how a graph can be used to obtain a solution	
	P8. solve counting problems by listing all the possible combinations or permutations from a set of at least four elements	M6 solve counting problems involving sets of at least 10 elements by calculating factorials, permutations or combinations, using standard notation	D3 explain the order of complexity of counting problems
4 Be able to manipulate graphics using co-ordinates	P9 represent two dimensional shapes using co-ordinates	M7 explain how three dimensional objects are represented using three dimensional co-ordinates	
	P10 carry out transformations of two dimensional shapes	M8 describe transformations of two dimensional shapes given a start and end shape	D4 use matrix multiplication to scale, rotate and reflect two dimensional shapes

TEACHING CONTENT

The Teaching Content describes what has to be taught to cover **all** Learning Outcomes.

Learners must be able to apply relevant examples to their work. Where examples are given in the Teaching content, these are suggestions; they do not have to be the examples that the learner uses.

LO1 Be able to solve numerical problems using algebraic abstraction

- Numbers and quantities
  - distinguishing between integers and real numbers
  - distinguishing between discrete and continuous quantities.
- Operations on numbers
  - basic arithmetic operations (i.e. addition, subtraction, multiplication, division)
  - integer division (quotient and remainder) and modulo arithmetic
  - rounding and associated operations (e.g. trunc(), int(), round(), floor(), ceil())
  - other common operations on numbers (e.g. abs(), square(), sqrt()).
- Algebraic abstraction and manipulation
  - building expressions to represent problems, including the use of variables
  - substituting values for variables in expressions
  - manipulating algebraic expressions (e.g. expansion of brackets, factorisation)
  - solving linear equations with one unknown
  - solving linear inequalities
  - solving non-linear equations numerically, by iteration.

LO2 Be able to use propositional logic to represent and solve problems

- The language of propositional logic
  - truth values FALSE/TRUE (0/1)
  - operations AND, OR, NOT and their standard symbols (∧, V and ¬)
  - propositional variables.
- Manipulating logic expressions
  - truth tables for expressions
  - contradictions and tautologies
  - properties of propositional calculus (e.g. commutative, associative, distributive)
  - deriving and simplifying logic expressions.

LO3 Be able to solve problems involving collections of objects

- Set theory
  - concepts of set, subset and set membership – and the standard notation
  - set operations (i.e. intersection, union, complement) and the standard notation
  - Venn diagrams
  - representing problems as sets.
- Graph theory
  - concepts of graph, vertex and edge
  - directed and non-directed graphs, labelled edges
  - representing problems as graphs.
- Combinatorics
  - factorials (including standard notation)
  - permutations and combinations
  - order of complexity (e.g. linear, quadratic, polynomial, exponential).

LO4 Be able to manipulate graphics using co-ordinates

• Co-ordinate systems
  o cartesian co-ordinates in two dimensions (including origins at centre and at top-left of screen)
  o three dimensional co-ordinates.
• Manipulating two dimensional shapes using co-ordinates
  o representing shapes using co-ordinates
  o transformations of shapes (i.e. translation, scaling, reflection, rotation)
• use of matrix multiplication to perform scaling, reflection and rotation

GUIDANCE

Delivery guidance

The focus in this unit is on the practical application of mathematical and computational thinking skills in solving problems, rather than learning abstract mathematical concepts and methods. While it will be necessary to teach some of these concepts and methods, this should be done in a context where the relevance to an actual problem to be solved is evident to the learners. Ideally, where this unit is being delivered as a stand-alone unit alongside Cambridge Technicals in IT (or similar qualifications), the content for the unit should be taught and assessed with the units where the skills are used wherever possible. These include units on software development, numerical modelling using spreadsheets, computer graphics and computer architecture.

Whether it is delivered separately or not, it is expected that learners will use computing solutions such as writing programs, spreadsheets or mathematical tools such as Derive or Wolfram Alpha rather than complete the unit purely using pen and paper. The emphasis is on obtaining and demonstrating a secure understanding of the underlying principles and how they can be used to solve problems, rather than on learning specific pen and paper methods. That said, there are some specific methods that learners should be able to perform without a technological aid and where these occur, they are identified below.

Be able to solve numerical problems using algebraic abstraction

The tutor should make learners aware that quantities can be discrete or continuous. This is fundamental to deciding whether they are best represented as integers or real numbers. However, it is not always clear cut, for example in the case of discrete values which include fractions. This area ties in well with units on data representation where learners are shown how integers and real numbers are represented in the computer and with elementary units in software development where learners have to select appropriate data types for variables.

To explore and solve numerical problems, learners should be given word problems in context that they can relate to, such as the number of tiles/bricks etc. for a building project. The problems should be general enough so that the learners have to determine what values in the problem are relevant, what values are unknown and need to be input to obtain a given solution and how the solution is to be obtained. While learners will most probably be using a technological aid to formalise their solutions (such as a writing a computer program), they should be taught how to manually substitute variables in an expression and to solve linear equations of two or more steps. This can be used to verify that their computerised solution is correct.

Learners should solve problems involving both continuous and discrete data. Tutors should ensure that in the case of discrete data and integers, they are exposed to problems involving integer division and modulo arithmetic such as calculations with time. They should also be

made aware of issues with rounding when dealing with integers. Typically, with a discrete quantity, it is necessary to round either up or down, and learners should be taught how to justify which is used. With continuous quantities, values are more often rounded off to the nearest value available in the precision required. Programming languages, spreadsheet software and other tools used will normally provide a variety of facilities to perform different types of rounding and learners should be taught the differences between them. Tutors should note that this is implemented differently in different software, for example in the way that they round negative numbers up or down, or in the way in which halves are rounded off to the nearest whole.

Learners should practise a number of skills to manipulate polynomial expressions which they will use as the basis for solving equations and inequalities. These should include expanding brackets with a coefficient, multiplying two polynomials and factorising. In solving equations, they should learn to change the subject of the equation to the unknown quantity (e.g. for example by performing identical operations to both sides). Inequalities can be approached in the same way as equations, but learners need to consider the problem in context to decide the inequality in the solution. Learners should also solve problems that are non-linear. While they may be able to solve some of these algebraically, for example by making x2 the subject of an equation and finding the root, they also need to learn to use numerical method and a technological aid to obtain a solution when there is no obvious algebraic method. A typical example is a loan amortization calculation where they find how long it would take to pay off a loan with a given interest rate. Learners should be aware that, depending on the problem, such a method may only find an approximate or local solution. (A local solution occurs when the iteration leads to a solution in the range explored by the iteration, but it's possible that there may be other possibly better solutions out of that range).

Be able to use propositional logic to represent and solve problems
For the purpose of this unit, propositional logic and Boolean algebra can be considered as synonyms. It is likely that the tutor will introduce many of the concepts theoretically, but learners should then also have the opportunity to see them applied in context. Typically, the context will be specifications of input and output conditions of systems such as security systems . Alternatively, logic circuits may be used – if so, the tutor should clearly place this in the context of circuit design (e.g. for example in designing the controller to a 7-segment display). Although the contextualisation may be introduced using systems with only two inputs and one output, learners should progress to systems with more inputs and outputs.

Truth tables will be the main tool that learners will use to describe such systems. The tutor should introduce the concepts of a tautology and a contradiction as expressions which always result in TRUE and FALSE respectively on all the rows of the truth table. The tutor will need to explain the commutative, associative and distributive properties of logical operations and the learners will probably need some manual exercises in simplifying expressions using these properties before they are able to apply it in a problem- solving context.

As well as being able to derive truth tables from expressions, the learners will need to learn how to derive the expressions if given a truth table. At a basic level, this can be done by selecting the rows where the value of the output is 1 and expressing these conditions in disjunctive normal form. Learners should attempt to simplify these expressions by using the properties learnt. Other properties and methods such as De Morgan's laws and Karnaugh maps can also be used.

There are obvious connections between this learning outcome and the set theory section of learning outcome 3. While the two learning outcomes are likely to be delivered and assessed separately, the tutor may wish to point out these connections, to encourage a deeper and fuller understanding.

Be able to solve problems involving collections of objects

The tutor should introduce the theoretical concepts involving sets and the associated notation first using concrete examples and exercises to ensure that learners understand. The learners can then apply these concepts to express and solve problems. It is unlikely that learners will write computer programs that will solve the problems that require sets and graphs, but they may well write programs that compute the number of combinations in a scenario or other similar problem. For sets and graphs, it is recommended that tutors enhance the understanding of the concepts involved by the use of specialised mathematical software such as WolframAlpha (general, but will perform operations on sets) and various graphical modelling packages that contain facilities for drawing and manipulating Venn diagrams and graphs. Tutors should remember that the key focus at this level is to learn how to create abstractions of problems using sets and graphs, rather than implementing algorithms on these structures. Hence the use of computer tools to assist the modelling is encouraged.

Learners should have the opportunity to use sets as an abstraction to a problem by deciding, from the problem, what sets are needed, what the membership of each set is and how the sets need to be combined to obtain the solution.

A similar approach should be taken for graph theory with an introduction to the basic concepts and notation. Learners should be made aware of the meanings of the terms in this context and that these may be different from other uses of these terms. Also, some sources use the terms node for vertex and arc for (directed) edge. The emphasis here is to be able to use a graph as a visual representation of a problem. Learners should understand why, in the context of a given problem, edges may need to be directed or labelled. Having been able to abstract the problem into a graph, learners should then be able to explain how they use this, with or without the help of a computer tool, to solve the problem.

The learners should solve simple combinatoric/counting problems over a relatively small set by enumeration. The number of possibilities can then be investigated and this will enable the tutor to introduce the concepts, notations and formulae for factorials, permutations and combinations. Learners should then apply these to larger problems. By investigating how the number of possibilities grows with the size of the set, learners should classify such problems by their order of complexity. They should encounter some problems that are linear, polynomial and exponential and may also deal with problems of different orders of complexity (e.g. constant or logarithmic). The use of big-O notation is not required. Drawing graphs showing the growth of the complexity against set size, and comparing this with standard graphs for $y=x$, $y=xn$ and $y=nx$ will provide a visual way of conveying these concepts which will appeal to some learners. Computer tools (such as the regression tool in Microsoft Excel or TI-Nspire) can also be used to obtain the equations for the growth graphs and thus justify the order of complexity.

Be able to manipulate graphics using co-ordinates

Learners should be familiar with Cartesian co-ordinates in two and three dimensions although most of the work for this learning objective will be in two dimensions. While learners will probably already be familiar with two2 dimensional co-ordinates from their prior learning, the tutor should point out that the position of the origin and the direction of the axes are arbitrary. Learners should experience this over a range of tools – they will be accustomed to having the origin at the centre or bottom left and the x and y axis horizontally and vertically respectively. However, many 3-D representations will have the z -axis vertical, with the y axis perpendicular to the page. Also, many programming systems have the origin at the top right corner of the screen space, with the y axis going down rather than up.

The tutor should introduce the four transformations theoretically. Learners will probably use pencil and paper methods initially to explore these as they carry them out or to determine what transformation has been carried out. Learners should be taught to specify translations using a translation vector. Scaling requires a scaling factor and a centre. Rotation requires an angle, direction and centre. Rotation by multiples of 90° around the origin will be typical, but more advanced learners may explore more. Reflection will typically be across the x or y axes, or the lines y=x or y=-x.

The tutor should introduce learners to matrix multiplication within the context of carrying out transformations, demonstrating transformation matrices as a common way of specifying and computing transformations. For the transformations listed above, as long as the centre of rotation or scaling is the origin, the matrices involved are accessible at this level. Learners need to learn how to multiply a 2x2 matrix by a 2xn matrix. Once they understand how to carry out this multiplication, learners may research the required transformation matrices or derive them by trial and error. Advanced learners should be given the opportunity, with appropriate technological tools and an explanation by the tutor of the trigonometry involved, to derive and use the matrices for rotations by an angle that is not a multiple of 30.

This learning outcome is more difficult to contextualise as problems than the others. The most obvious solution is to require learners to write a program that displays shapes and their transformations (for example a teaching aid). In any case, learners should have the opportunity to use co-ordinates on a computer and to use a computer to calculate matrix multiplications. The tutor should point out that it covers the mathematical basics required to work with computer graphics at an appropriate level, and that if learners progress to programming in 3-D graphics for example, these basics can be built on to understand essential mathematical tools that underpin this area.

Assessment evidence guidance

Assessment Criteria P1, P2, P3, M1, M2

Learners should be given a scenario which describes a situation for which a numerical solution is needed. The scenario should be complex enough to allow the learner to demonstrate the different skills needed for the assessment criteria (or a series of scenarios may be used). The scenario should be in a context that the learner can understand sufficiently to be able to derive the algebraic expressions needed themselves. For example, if the case study involves writing a computer program to calculate taxi fares, the scenario should describe the fare structure without stating precisely what operations are needed (e.g. “£1.20 per mile” rather than “£1.20 times the number of miles”).

The same scenario could be used to assess learners for competence in other units such as software development or spreadsheet modelling units.

Learners must identify the quantities in the scenario that are relevant to solving the problem and identify whether they are discrete or continuous, integer or real. If any relevant quantities are unknown, learners must identify these using variables. If they are writing a program to solve the problem, then this is likely to be in their analysis of the problem to be solved or can partly be evidenced by their declaration of variables to hold these quantities. In order to solve the problems there should be scope for learners to use all the four basic operations and rounding and learners must show an understanding of the use of these by using them correctly in their programmed solution and calculating expected results for test data.

For merit criterion M1, learners must also demonstrate (e.g. for example in an evaluation of their solution) that they are aware with issues caused by rounding and how these affect their solution. They should explain the different types of rounding (ideally by referring to the

rounding functions available in the software) and justify the type they have used in the context of the problem being solved.

For merit criterion M2, learners must have solved problems which involve integer division or modulo arithmetic. While it is useful that learners highlight and justify where this is used in their solution, this is not necessary. It is sufficient that these have been used correctly within the solution for the criterion to be awarded. A suitable complex scenario should contain scope for both P3 and M2, rather than a separate scenario being created just for M2.

Assessment Criteria P4, M3, D1

Learners must demonstrate that they can solve linear equations by algebraic manipulation. This should be within the context of the problem being solved in the scenario used for criteria P1, P2 and P3. However, this will be done on paper rather than through their programmed solution, either within the context of providing test data for the program or justifying/explaining results obtained, and it is expected that learners will show the working for solving the equation, detailing what manipulations have been carried out to make the unknown quantity the subject of the equation.

For merit criterion M3, learners must correctly solve at least two linear inequalities. This will normally be in context, by solving a corresponding equation as required for P4. For example, if the fare for a taxi journey is given by $1.2x + 3$ and the learner wants a result greater than £10, they need to first solve $1.2x + 3 = 10$ to obtain 5.8333... as required by P4. For M3, they should then justify whether this means they want values greater or less than 5.83333, how this value should be rounded and whether the value itself is acceptable, by referring to the context.

For distinction criterion D1, the learners should use a computer program or spreadsheet to obtain a solution for a non-linear problem by trial and improvement. They are not required to also solve the problem algebraically (if this is possible). The problem should be set within the context of the scenario used for P1, P2, and P3 as an extension task.

Assessment Criteria P5, M4, D2

Learners must demonstrate that they can construct truth tables to solve problems. They should be given a scenario involving at least three binary inputs into a system. Examples include the sensors in a security system and desired responses, attributes of people in an organisation and tasks they should be given or logic circuits. The scenario should give learners enough information to enable them to derive logic expressions that describe the desired outputs without spelling out the expressions. The scenario can be used further by asking learners to then implement/simulate the system as evidence for a different unit. However, only the ability to draw the truth table is required for this unit.

For merit criterion M4, learners should be able to do this process in reverse. They should be given a description of a scenario which tells them the outputs of the system under certain conditions. This scenario will ideally be related to the scenario used for P5 to make a cohesive whole, but will obviously have to be different. Using the truth table, the learners should demonstrate that they can derive logic expressions for the outputs by inspecting the rows that are true. The result at merit level may be in disjunctive normal form. For example, if the inputs are A, B and C and only rows 0 1 0 and 1 1 0 give a required output, the answer may be given as (NOT A AND B AND NOT C) OR (A AND B AND NOT C). There should be an attempt to check the expression obtained against the system specified scenario.

For distinction criterion D2, learners should correctly simplify answers obtained from M4, explaining each step of their simplification using properties of propositional calculus (or other means) and giving reasons for their final expression in relation to the scenario. In the example given above, an adequate simplification would be (B and NOT C).

Assessment Criteria P6, P7, P8, M5, M6, D3
Learners should be given problems to solve from a scenario which enables all the assessment criteria to be addressed. Such a scenario could be the organisation of a sports tournament. In order to put the players or teams into groups that meet specific requirements, sets can be used to group the players/teams according to various criteria and, from these, optimal groupings can be derived and justified. In order to predict the likely winner of any game between two players/teams, a graph can be used to represent all previous encounters. By using appropriately directed and labelled edges, predicting the likely winner becomes a classic network flow problem. The number of games needed and how this varies with the number of groupings of players/teams is a suitable counting problem. Alternatively, three separate scenarios may be given.

Learners should use sets to model a problem in the scenario and show how these sets have helped them to obtain the solution. Using Venn diagrams or otherwise, they should describe how the scenario has been modelled and the solution they have obtained as a result, including any set operations that were needed. It is not required that the solution produced be an optimal solution, only that the problem was correctly modelled and this model clearly used to obtain the solution.

Although learners have to demonstrate that they can model and solve problems using sets, for graphs it is sufficient at Pass level that learners demonstrate that they can correctly model the problem in the scenario, since working with graphs will probably involve a higher level of skill than working with sets. Their evidence should include the graph produced as well as an explanation of how this models the problem. This must include what the vertices and edges in the graph represent and if the edges are directed and/or labelled then this should be justified. The graph produced must correctly follow from the scenario.
Learners should also demonstrate that they can solve counting/combinatoric problems over a set of at least four elements, in which they determine the number of possible combinations that satisfy a requirement by listing them exhaustively. They should solve at least two such problems, obtaining the correct count and list of combinations in each case.

For merit criterion M5, learners must obtain a solution to the problem in the scenario and explain how this solution was obtained using the graph produced for criterion P7. If a computer tool has been used, the evidence must include screenshots that show that the graph was entered into the tool, the report requested (e.g. shortest path or maximum flow between two vertices) and the result obtained. The learner should also explain why the report used solves the problem in the scenario. If a computer tool has not been used, learners can give an explanation of their solution from the graph in prose, explaining their working. The solution should be reasonable, but it is not necessary that the solution be optimal.

For merit criterion M6, learners must extend at least one of the problems solved for criterion P8, by determining the number of possible combinations that solve the problem when the set is at least 10, and sufficiently large that it is not feasible or reasonable to solve this by enumeration. Enough evidence should be provided to demonstrate how the number of combinations was calculated. Conventional mathematical notation should be used. For factorials, this is the post-fix operator! (exclamation mark). For permutations and combinations, there are a number of standard notations any of which are acceptable. Where this unit is combined with a software development unit, producing a program that correctly calculates the result is sufficient to achieve this criterion. In this case, the learner will have used the functions provided by the programming language for factorials, permutations and combinations, and this is also adequate.

For distinction criterion D3, learners demonstrate an understanding of linear, polynomial and exponential order complexity and correctly explain why problems they have solved for P8 and M6 fall under at least two of these categories.

Assessment Criteria P9, P10, M7, M8, D4

Learners must demonstrate that they can correctly represent shapes as a set of two dimensional co-ordinates. While this can be done manually, evidence could also be in the form of a program that draws a variety of shapes. In this case, the use of co-ordinates for vertices, centres etc. (rather than logo-style turtle graphics) must be evident. A variety (at least three different) of shapes should be included. As an example, learners may be asked to use the graphics functions in a programming language to draw a simple face with circles for an outline and eyes, a triangle for the nose and a rectangle for the mouth.

Learners should also correctly carry out at least one each of the following transformations: translation given the translation vector; scaling given the centre and scale factor; rotation given the centre and angle of rotation (the centre will normally be the origin and the angle a multiple of 90°); and reflection given the line of reflection. It is sufficient for the same shape (e.g. a rectangle) to be used for this purpose.

For merit criterion M7, learners must demonstrate that they understand how three dimensional co-ordinates work. The explanation must include diagrams showing the use of axes in 3D, an example of a simple three dimensional object and the co-ordinates that can be used to represent it. The learner may draw these diagrams themselves or use screenshots generated using 3D drawing software. This explanation can be set as a written task at the end of the assignment for this learning outcome.

For merit criterion M8, learners must describe the transformation that has occurred, given the start and end shapes. This should include at least one of each of translation, scaling, rotation and reflection.

For distinction criterion D4, Learners must show how matrices can be used to carry out transformations on Cartesian co-ordinates. This must include scaling (about the origin), rotation (about the origin) and reflection (in the x and y axes and in the lines x=y and x=-y). Writing a computer program or producing a spreadsheet model is adequate evidence of the ability to multiply matrices, and, if used, the evidence should show examples of this solution tested with each of the three transformations. If the criterion is to be achieved without the use of a computer tool, learners will need to show evidence of having carried out the multiplication.