L3 Computing - Unit 1 - Computational Thinking

Relevant LINKS

BACK TO COMPUTING UNITS **[1]**

Handbook home page **[2]**

Overview

Computer Science at Platinum Level requires the candidate to completely understand and describe the basic building blocks of computing such as the use of abstractions to formulate real world models; algorithms for planning program structures; programming languages, both textual and visual; and binary and boolean patterns and statements. As a result of reviewing their work, they will be able to identify and use automated methods or alternative ways of working to improve programming and using computers. Unfamiliar aspects will require support and advice from other people.

A work activity will typically be ‘straightforward or routine’ because:

The task or context will be familiar and involve few variable aspects. The techniques used will be familiar or commonly undertaken.

Example of context – designing, planing, implementing and testing a program for controlling a physical system or solving a complex problem.

Support for the assessment of this award

Example of typical Computing work at this level (Coming Soon)

Assessor's guide to interpreting the criteria

General Information

QCF general description for Level 3 qualifications

- Achievement at Level 3 (EQF Level 4) reflects the ability to identify and use relevant understanding, methods and skills to complete tasks and address problems that, while well defined, have a measure of complexity. It includes taking responsibility for initiating and completing tasks and procedures as well as exercising autonomy and judgment within limited parameters. It also reflects awareness of different perspectives or approaches within an area of study or work.
- Use factual, procedural and theoretical understanding to complete tasks and address problems that, while well defined, may be complex and non-routine.
- Identify, select and use appropriate skills, methods and procedures.
- Use appropriate investigation to inform actions.
- Review how effective methods and actions have been.
- Take responsibility for initiating and completing tasks and procedures, including, where relevant, responsibility for supervising or guiding others.
- Exercise autonomy and judgement within limited parameters information and ideas.

Requirements

(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){ (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o), m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m) })(window,document,'script','//www.google-analytics.com/analytics.js','ga'); ga('create', 'UA-46896377-2', 'auto'); ga('send', 'pageview');

- Standards must be confirmed by a trained Level 3 Assessor or higher.
- Assessors must at a minimum record assessment judgements as entries in the online mark book on the INGOTs.org certification site.
- Routine evidence of work used for judging assessment outcomes in the candidates' records of their day to day work will be available from their ePortfolios and online work.
- Assessors should ensure that relevant web pages are available to their Account Manager on request by supply of the URL.
- When the candidate provides evidence of matching all the criteria to the specification, subject to the guidance below, the assessor can request the award using the link on the certification site. The Account Manager will request a random sample of evidence from candidates' work that verifies the assessor's judgement.
- When the Account Manager is satisfied that the evidence is sufficient to safely make an award, the candidate's success will be confirmed and the unit certificate will be printable from the web site.
- Each unit at Level 3 has recommended guided learning hours based on time required to complete by an average learner.

Assessment Method
Assessors can score each of the criteria N, L, S or H. N indicates no evidence and it is the default setting. L indicates some capability but some help still required to meet the standard. S indicates that the candidate can match the criterion to its required specification in keeping with the overall level descriptor. H indicates performance that goes beyond the expected in at least some aspects. Candidates are required to achieve at least S on all the criteria to achieve the full unit award.

Expansion of the assessment criteria

1. The candidate will understand the computational problem solving process

1.1 I can organise data in terms of logical patterns

Candidates should be able to find patterns in simple and complex data sets and classify and organise the sets appropriately. They should be able to make the link from simple analysis to data mining as a significant application.

Evidence: portfolios of evidence, internal testing.

Additional information and guidance

Candidates should provide evidence of researching how data can be classified according to the way it is organised. Information is organised data and links between information sets can be very useful. This can start with simple number patterns and what the numbers represent e.g. Fibonacci series and spirals in nature, bit patterns that represent letters, numbers or images.
Computer data files are complex patterns but the patterns can be recognised by the right program which is why files can be successfully opened in some software but not all. (Really, file name extensions should not stop a file from being opened as the program should be able to determine whether the file is the right type from its content. Even corrupt text content should be replaceable by something so that most of the data is preserved)

From this they can build into the more macro scale data mining and why it is important. e.g. a bank’s safest customers are not the most profitable and unsafe customers borrow too much and default on payments. Clearly if the bank can work out who are the “not so safe” customers who will be safe enough to to optimise profits they are can target these. Advertising is a very important part of the internet ecosystem and patterns in data search can help companies link advertising to specific individual interests. There are significant privacy issues arising from this.

1.2 I can demonstrate how abstractions represent complex data structures and instructions

(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){ (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o), m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m) })(window,document,'script','//www.google-analytics.com/analytics.js','ga'); ga('create', 'UA-46896377-2', 'auto'); ga('send', 'pageview');

Candidates should understand the concept of abstraction to enable humans to interact with digital representations of data and instructions.

Evidence: portfolios of evidence, internal testing.

Additional information and guidance

Candidates should explore a range of ways of describing everyday objects in terms of abstractions. From this they should see the connection to defining data types, and instructions in simpler terms than the complex binary patterns that make them up in digital machines. They should consider many examples to embed the understanding in a range of contexts. For example using libraries and frameworks to speed up develop. All “blackbox” approaches are essentially abstractions. The model of an atom is an abstraction used by scientists. Newtons Laws are essentially abstractions of special and general relativity because they are simplifications of the true complexity of the way moving objects behave but this is good enough for most applications. The power of abstraction is that instructions, sets of instructions and associated data can be defined once and used many times without constantly having to deal with a lot of very complex detail. It is very important to define low level abstractions as efficiently as possible because any inefficiencies in them will become compounded as they are used as the building blocks for higher level abstractions.

1.3 I can iteratively refine solutions to improve efficiency and effectiveness

Candidates should be able to identify ways in which they can apply the principle of iteration to everyday tasks as well as work in computing.

Evidence: documented process in portfolios.

Additional information and guidance

Candidates should have many pieces of work where they have gone over something repeatedly to refine it and improve it with each pass. An example could be drafting an essay. Go over the first draft and focus on the structure of the argument, go over the second draft, improving sentence structure and style. Go over the third draft and remove redundant content and so on. The work will be improved with each iteration. The same could apply to composing music, editing a video or just about any activity. The principle of iterative improvement should be a transferable skill. It becomes a more exact and predictable activity in mathematics so move from these examples to e.g. convergence of a series. Take the numbers 1, 2, 4, 8, 16 etc. The familiar powers of 2. Add them iteratively. First iteration = 1 next iteration = 3 next iteration 7 and so on. We have a number that gets bigger and bigger to infinity. This divergent behaviour is not usually as useful as a convergent iteration. Now take the reciprocals of those numbers 1/1 = 1. Next iteration, 1.5, Next iteration 1.75 etc. If we use a computer to do this for thousands of iterations and plot a curve we will see that the series of numbers is converging to 2. This is a simple case that would not necessarily need a computer to work out where the convergence is leading but iterative methods are very useful in solving very complex problems, often using heuristic methods. Candidates should be able to see the link between patterns and iterations and abstraction.

1.4 I can use multiple algorithms to solve complex problems

Candidates should demonstrate that they can use more than one algorithm to solve problems, for example, breaking them down into smaller components.

Evidence: example problems in portfolios.

Additional information and guidance

Candidates can use algorithms that are defined in programs but also in other contexts. For example, to travel to Prague might require a set of procedures to get to the airport, a set of procedures to get through security and on to the plane, a set of procedures during the flight, going through

(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){ (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o), m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m) })(window,document,'script','//www.google-analytics.com/analytics.js','ga'); ga('create', 'UA-46896377-2', 'auto'); ga('send', 'pageview');

immigration etc. Each one of these components could be used separately as part of a different journey so it would be better to keep them as separate components and simply link them as required. Within each of these procedures there are likely to be subroutines that could again be abstracted from the main routine, e.g. putting your hand in your pocket to take out your passport. The candidate should be able to see the relationship between patterns, algorithms and abstractions in solving computational problems.

1.5 I can consult with relevant industry professionals and academics to improve solutions

Candidates should have opportunities to discuss their ideas and their computational learning with expert third parties.

Evidence: reports, recordings, portfolios.

Additional information and guidance

The main purpose is to get candidates to realise that the community is a great source of on-going learning. The means of achieving this is left to the assessor and the candidate but there must be evidence of some significant interaction.

2. The candidate will be able to apply number systems and logic to computing problems

2.1 I can explain the relationship between binary and hexadecimal numbers

Candidates should be able to handle operations on binary and hexadecimal numbers explaining the relationships between them and how they illustrate number system theories in general.

Evidence: internal testing portfolios.

Additional information and guidance

Candidates should be able to appreciate operations on binary numbers such as the implications of shifting bits to the left and right and addition, subtraction, division and multiplication. They should be able to relate binary numbers such as 11111111 to FF in Hexadecimal and 1111 to F.

2.2 I can explain how digital computers can work with a full range of real numbers.

Candidates should understand the principles of how a computer can store and manipulate floating point numbers.

Evidence: Internal testing and portfolios.

Additional information and guidance:

Candidates should understand that negative numbers can be represented by the "twos-complement" method. Flip the 0's to 1's, and vice versa, then add 1. (Why does this
work?)

For example:

Represent -2 in binary:

2 in binary is 00000010

Flip the bits, 11111101

add 1

-2 = 111111110

Note the negative number has 9 bits so to store it we'd need to reserve at least 2 bytes.

In order to deal with floating point numbers (note this is like decimals but not necessarily a decimal number so floating point is used rather than what we would say is the decimal point in our most familiar number system) the usual method is to represent the number by using a sign +ve or -ve, an exponent and a mantissa. This is like using standard form in decimals e.g. 3.8 x 107 . In this case the sign is +ve, the exponent is 7 and the mantissa is 3.8. Candidates should be able to transfer these ideas to floating point binary and see how floating point numbers need special consideration. Systems for dealing with floating point numbers can be developed in software or they can be implemented in hardware. Doing it in hardware makes the processing faster because we are taking the load off the CPU and giving it to a separate piece of hardware. Candidates should appreciate that since computers work with binary numbers, specific methods are needed to deal with floating point as well as negative numbers.

A simple introduction can be found here [3].

2.3 I can explain the difference between packed and unpacked binary coded decimal.

Candidates should understand the principles of representing numbers in BCD and packed BCD.

Evidence: internal testing, portfolios.

Additional information and guidance

Packed BCD numbers are stored two digits to a byte in 4 bit groups referred as nibbles. e.g. 42 in unpacked BCD would represent the number 4 in one byte and 2 in another whereas in packed BCD, 4 bits would store the number 4 and another 4 bits the number 2 taking up just one byte in total. Compare to hexadecimal where the digits take up all combinations in 4 bits whereas in BCD some bit capacity is redundant because we are only using 0 - 9 when there is enough capacity for 0 - 15.

2.4 I can use mathematical functions in practical algorithms.

Candidate should demonstrate the use of appropriate mathematical functions in their work.

Evidence: source code in portfolios.

There is no prescriptive list but candidates should be able to support practical functions that are relevant to their projects in specific contexts. For example, if they need the trajectory of an object flying laterally they should be able to find a suitable mathematical function to support this. They do not necessarily need to be able to derive the function, only to use it to get the desired outcome. Having said this it is also possible that in applying a function they will get to understand the maths behind it.

2.5 I can analyse expressions in boolean logic to simplify them.

Candidates should be familiar with the main Boolean operators and how boolean algebra works to simplify logic circuits.

Evidence: internal testing, portfolios.

Additional information and guidance

This link [4] provides an understandable comprehensive treatment including De Morgan's law and its implementation. Candidates should be able to use De Morgan's law to simplify multi-gate designs.

(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){ (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o), m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m) })(window,document,'script','//www.google-analytics.com/analytics.js','ga'); ga('create', 'UA-46896377-2', 'auto'); ga('send', 'pageview');

3. The candidate will analyse problems to create computational solutions

3.1 I can identify practical problems suitable for a computational solution

Candidates should be able to distinguish between problems that lend themselves to computational solutions and those that do not.

Evidence: Internal testing, portfolios.

Additional information and guidance

Generally problems that can be defined with a mathematical basis can be provided with computational solutions although in some cases the complexity of a problem could make an effective abstraction difficult or impossible. In principle, we could make a computer simulation of the entire universe but it would only be an approximate abstraction with a lot of detail missing. On the other hand, there is a lot of speculation about the possibility that the universe IS just a computer simulation e.g. here [5].

Candidate's should be encouraged to explore and debate such ideas and test what they have learnt about computations thinking against the ideas of their peers and others with more and less experience. Problems that are impossible to solve computationally are of the type “I have a problem with my partner...” because these have no mathematical basis. (Ok, we could possibly inform a partial solution through statistics but it is not going to be a generalisable solution that works in all cases). Candidates should realise that there are a lot of grey areas where a computational solution lends itself to some insight but no more.

3.2 I can analyse complex problems into simpler related components.

Candidates should demonstrate many examples where they have broken complex problems down into manageable related components.

Evidence: portfolios.

Additional evidence and guidance

Candidates should keep a good range of diverse examples in their portfolios. These can come from other work e.g. in science, mathematics, other subjects or areas of interest. Example in engineering might be to consider the forces on a structure and work out a key quantity by considering the vertical and horizontal components of the forces and taking moments about a point that eliminates unknown quantities. An example in biology might be to design three experiments to see how Woodlice react to environment. One experiment to test the effect of light, one experiment to test the effect of heat and one experiment to test the effect of moisture. Evidence for this criterion will also include analysis related to their own programming projects.

3.3 I can explain computational solutions in terms of sequential automated steps.

The candidate should be able to explain how computational solutions have or can be be achieved through descriptions of the sequential steps that are apparent.

Evidence: portfolios.

Additional evidence and guidance

Given common computational solutions candidates should be able to give plausible sequences of key events and control that explain how the solution has been achieved. For example, drawing a circle. Move forward 1/360 of the circumference and turn left 1 degree. Repeat this sequence 360 times. To play the national anthem. Make a file containing the data for the sounds of the notes. Each note will have data for pitch, amplitude and duration. Starting at the first note use the data in the note to make the sound defined by the data. Repeat for each note in sequence until the end of the file.

Candidates should have many opportunities to produce short working programs and describe the in these terms. They should see how many short programs can be aggregated to produce larger more complex systems but that the principle of automated sequences is present in many different parts. This can be related to more specialist work on convergent and divergent iterations.

3.4 I can find ways of making computational solutions more efficient.

Candidates should be constantly looking for ways of making their solutions to problems more efficient.

Evidence: descriptions of times where they have succeeded in improving the efficiency of one of their solutions. What did they do and why?

Additional information and guidance

Evidence might also include research that candidates have done to support their learning in a specific area so they know how to implement something as efficiently as possible. e.g. research on sort or search algorithms, efficient data storage, using integer arithmetic, using more efficient libraries.

3.5 I can work collaboratively and persistently to achieve a good computational solution.

Candidates should be cooperative with their peers, assessors and third parties. They should show courage and persistence to get tasks completed and to solve problems that cause them difficulties.

Evidence: Assessor observations. Documentation in portfolios.

Additional information and guidance

Assessors should bring this criterion to the attention of any candidates that have weak attitudes.

They need to satisfy this criterion before being eligible to take the grading exam. Where a candidate goes through a bad patch but then turns things around, this should be recorded in the evidence base.

Moderation/verification

The assessor should keep a record of assessment judgements made for each candidate and make notes of any significant issues for any candidate. They must be prepared to enter into dialogue with their Account Manager and provide their assessment records to the Account Manager through the online mark book. They should be prepared to provide evidence as a basis for their judgements through reference to candidate e-portfolios and through signed witness statements associated with the criteria matching marks in the on-line mark book. Before authorizing certification, the Account Manager must be satisfied that the assessors judgements are sound.

Source URL: https://theingots.org/community/cpl3u1ctx

Links
[1] http://theingots.org/community/Computing_qualification_info_units
[2] https://theingots.org/community/sites/default/files/uploads/common/Handbooks/Computing/L3_OpenSystems_Computing_v2%20.pdf
[3] http://www.tfinley.net/notes/cps104/floating.html
[4] http://www.allaboutcircuits.com/vol_4/chpt_7/1.html
[5] http://www.huffingtonpost.co.uk /2012/10/11/physicists-may-have-evide_n_1957777.html

(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){ (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o), m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m) })(window,document,'script','//www.google-analytics.com/analytics.js','ga'); ga('create', 'UA-46896377-2', 'auto'); ga('send', 'pageview');