L1 Computing - Unit 1 - Computer Science

Relevant LINKS

BACK TO COMPUTING UNITS **[1]**

Handbook home page **[2]**

Overview

Computer Science at Silver Level requires the candidate to understand 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 basic program for controlling a physical system.

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 1 qualifications

- QCF general description for Level 1 qualifications
- Achievement at QCF level 1 (EQF Level 2) reflects the ability to use relevant knowledge, skills and procedures to complete routine tasks. It includes responsibility for completing tasks and procedures subject to direction or guidance.
- Use knowledge of facts, procedures and ideas to complete well-defined, routine tasks. Be aware of information relevant to the area of study or work
- Complete well-defined routine tasks. Use relevant skills and procedures. Select and use relevant information. Identify whether actions have been effective.
- Take responsibility for completing tasks and procedures subject to direction or guidance as needed

Requirements

- Standards must be confirmed by a trained Level 1 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.
- This unit should take an average level 1 learner 50 hours of work to complete, with 40 hours of learning under specific teacher presence.

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. Once the candidate has satisfied all the criteria by demonstrating practical competence in realistic contexts they achieve the unit certificate.

Expansion of the assessment criteria

1. The candidate will design, use and evaluate computational abstractions

1.1 I can develop abstractions to represent physical objects

Candidates should be able to develop simple abstractions related to physical objects

Evidence from assessor observations, content of learner portfolios.

Additional information and guidance

Abstraction is used to reduce complexity. Drawing a circle to represent a wheel is an abstraction of the wheel and represents the possibility to describe many different wheels. Once we have the basic abstraction we can add further information to get different kinds of wheel. e.g. a big or small wheel, a spoked wheel or a tyred wheel. Candidates should be provided with many opportunities to take physical objects and create abstractions by eliminating the details associated with the main attribute being abstracted. This does NOT require a computer (although there is no ban!) Shapes such as a square to represent a box, a sphere to represent a ball, planet, or star are some of the simplest abstractions. Suitable activities could be to get candidates to use simple shapes to represent familiar objects. e.g. 3 equilateral triangles to represent a Christmas Tree. Work in pairs. One draws an object the other has to guess what it is and provide instructions to the first to add more shapes to it to make it easier to decide what the original object was intended to represent. In some case we might find the first idea morphs into something the originator of the first shape did not intend. We can infer from this that greater complexity and more information define an abstraction more precisely and more uniquely. So in the case of the Christmas tree the first student draws a triangle. The second adds two more to make a tree shape. The first adds a star on the top so the tree becomes a Christmas tree and so on. A triangle start could equally be a woman’s skirt or a pyramid. Adding arms. legs and head would then start to define it as an abstraction of a female, adding blocks and putting next to a picture of a Sphinx makes it a pyramid. Notice that the context of an abstraction can help reduce ambiguity about what it is intended to be. Common use of abstractions is in road signs and other simple icons that represent more complex physical objects. Another game to play is taking complex objects such as a car and representing it with the simplest abstraction that a partner can recognise as a car. Doing this in a vector drawing program such as Inkscape will also help develop computer drawing skills. Vector drawing packages are a very easy to understand

implementation of abstraction in software engineering since drawings are built up from simple objects that can be grouped to form other larger more complex objects. By adding detail (information) such as textures to surfaces, drawings become more and more realistic.

1.2 I can use data patterns to represent physical objects

Candidates should be able to relate images to patterns of data.

Evidence: from assessor observations, documentation in portfolios.

Additional information and guidance

Candidates should be given a lot of opportunities to think of images in terms of patterns of dots. This can start with looking at printing images to paper at various resolutions, investigating pictures in magazines or newspapers using magnifying glasses and low power microscopes. Can they use a microscope to work out the resolution of an image by counting the dots? What happens if you move your eye further away from a grainy image? Use Inkscape [3] or similar drawing program to make a 5 x 5 grid. Fill in squares to make letters. Some letters like L or E are easy to get a good representation. M and W are much more difficult. How could we make it easier? Use a grid with more squares. Note that saying use a bigger grid is ambiguous as it could mean just bigger squares which would not help. More squares gives us more information, bigger squares are just the same information magnified. This is why with telescopes and microscopes magnification is not as useful as resolving power. Resolving power lets us see the details, magnification just gives a bigger view of the same level of detail and keep magnifying and you just get a big blurred image. This can be related to the work on abstraction. More relevant information means clearer more specific results. It is a universal principle whether dealing with images, economics or the environment.

A challenge. Your friend has 4 pencils. One red, one blue, one green and one black. You have only a black pencil and you can only write the numbers 0, 1, 2, and 3. You have a 20 by 30 grid on a sheet of paper and you want to send your friend enough information for them to draw the Republic of the Gambia Flag.

Send you friend a grid with the information they need to create the flag. They will do the same for you and see if you both get an identical flag at the end. Can you think of any ways you could reduce the amount of information you need to send?

As an extension to this consider getting other colours. Since all colours are made up from red, green and blue we can combine red, green and blue in different amounts to get all the other colours. If we had 3 levels of red, 3 levels of green and 3 levels of blue, how many colours could we make? Make a 3 x 3 grid for R1, R2, R3 and G1, G2, G3. 9 possibilities. add another 3 making the grid a cube and its 27. 3 x 3 x 3. So how many colours from 255 levels of red, 255 levels of green and 255 levels of blue? This is getting well beyond what is required at level 1 but provides a good grounding for Level 2. Ideas of file compression can be opened up by an instruction to print “1” 30 times rather than writing “1” 30 times to print a red stripe. A way of doing that would be to say if there is a change of colour, the first number tells you how many of the next number to print. So 30, 1 would replace 111111111111111111111111111111. We are allowed to use 3 and 0 so this would work as long as the person at the other end knew how to decode it.

1.3 I can follow instructions to develop a software abstraction

Candidates should be able to follow a set of instructions accurately to get a working abstraction

Evidence from assessor observations, content of learner portfolios.

Additional information and guidance

To achieve an abstraction in software a simple case would be to draw a box in logo with REPEAT 4 [FD L RT 90] The size of the box depends on the parameter L. We could add more parameters to change the colour and texture of the box. At this level the simplest cases are sufficient. A software abstraction is using some code to make a simple model of something in the material world and usually it is is an approximation to the real thing achieved by eliminating some of the details. The simplest abstractions such as drawing a line to represent a road or a circle to represent a stick person's head or a triangle for a skirt can all be represented in simple code. If a set of instructions are provided for the candidate to follow in order to get a working program, these constitute an algorithm. We then have an algorithm to produce the code and the code itself is an algorithm to produce the abstraction. The success of the algorithm to produce the code depends on how much information the programmer needs and for sufficient detail to be provided to them that they can understand. Similarly the success of the code as an algorithm depends on its accuracy and the ability of the computer programming language to understand the code as specified by the programmer. If both algorithms are well designed, the output will be as expected.

1.4 I can use software abstractions that model real world systems

Candidates should be able to use software models of real world systems.

Evidence: from assessor observations, schemes of work, content of learner portfolios.

Additional information and guidance

Any use of a computer simulation of a real world system is a target for this criterion. Numpty Physics [4] is a free and appropriate example that should help learners of this age understand the principles of abstraction since it is a simplified representation of the physical world. Using an emulator [5] will also give that feeling of abstration.

1.5 I can identify strengths and weaknesses in computer models

Candidates should be able to list strengths and weaknesses in models that they use.

Evidence: From content of learner portfolios.

Additional information and guidance

If we take the Numpty Physics model, strengths are that is fun and easy to use, weaknesses are that the representation of reality of real objects is visually very approximate. Of course that is also a strength in relating to general physical behaviour in a fun way. In general, encourage debate about why things are considered strengths and weaknesses. A lot of unnecessary detail could provide little benefit and make the software a lot more expensive or demanding of the hardware and it might actually be less fun. The interesting issues are usually the ones where there can be disagreement. This is also an opportunity to review games and to develop descriptive language beyond "I like it" or its good to say why it is good and to see if there is consensus. Another aspect is to identify strengths and weaknesses in coded algorithms. For example, where code is badly documented, inefficient, or particularly elegant. These aspects are very much more difficult to assess except in superficial ways in controlled tests so encourage understanding in the course of normal activities through, for example, peer review.

2. The candidate will understand algorithms

2.1 I can write algorithms for everyday tasks

Candidates should be able to use ordered lists of instructions to define procedures for everyday tasks.

Evidence: Assessor observations, local testing, portfolios.

Additional information and guidance

Algorithms are step-by-step problem-solving procedures. It is really a "fancy" word for something everyone does every day if they have any routines. Get up, go to the bathroom, get dressed, go down stairs, put the kettle on. All that is required here is that candidates can list the steps in everyday tasks. They should appreciate that repetitive tasks that are simple to replicate lend themselves to algorithms. Complex unpredictable behaviour doesn't. For example riding a bike to school. Mount bike, repeat (pedal) until I'm at school, dismount, park bike. Of course this is very simplified since we are saying nothing about avoiding traffic or turning corners. In practice a full algorithm that took into account all the possible events that might take place on the journey is a whole lot more complicated. It would be very difficult if not impossible to write an algorithm to coincide with how an individual lives for a month.

2.2 I can identify different algorithms that target the same task

Candidates should be able to recognise at least two different methods of approach to a problem that lends itself to an algorithm

Evidence: Assessor observations, local testing, portfolios.

Additional information and guidance

A useful resource to underpin this learning outcome is at CS Unplugged [6]. A bubble sort and a merge sort are both sort algorithms but they can result in different efficiencies in achieving the outcome. A linear search and a binary search are again different methods targeted on the same problem. At level 1 candidates would be able to determine that two methods are achieving the same result even though the methods might work differently.

2.3 I can compare algorithms

Candidates should be able to compare algorithms based on properties such as how easy they are to follow or understand, how efficient they are and the circumstances where they might work and not work.

Evidence: From local testing and portfolios

Additional information and guidance

Candidates should be able to see that some algorithms are easy to follow and some are much more complex. They should be critical of algorithms that are not documented well enough to understand. They should be able to carry out simple tests comparing e.g. two sort algorithms to see which is the most efficient. Guidance should be provided on the need for controlling variables such as the number of items to be sorted or searched so that the test is fair. This would be a good aspect to relate to the science curriculum. A mixture of algorithmic games and activities away from a computer and some with code will help in developing transferable understanding.

2.4 I can apply logic to efficiency and effectiveness of algorithms

Candidates should be able to provide rational reasons why they would prefer to use one algorithm rather than another.

Evidence: From assessor observations and portfolios

Additional information and guidance

As part of their testing of algorithms, they should be able to provide reasons why one algorithm is more suited to a task than another. This might need reference to a particular context such as search strategies in a game of "Battle Ships". CS Unplugged [6] is a possible resource for this.

2.5 I can change variables in an algorithm and predict the effect

Candidates should use existing code and experiment with changing one variable at a time to see its effect.

Evidence: Assessor observations and portfolios.

Additional information and guidance

This is another opportunity to reinforce control of variables. Experimenting with variables in several different scenarios should enable candidates to make predictions about effects in simple but unfamiliar code.

2.6 I know how instructions and data are stored

The candidate should know that instructions are stored in programs in a particular sequence and executed one after another but very quickly so that often it appears as if several things are happening at the same time.

Evidence: Internal testing, portfolios

Additional information and guidance

Conditional statements in programs determine which instructions are executed and in what order. Modern processors can execute hundreds of billions of instructions in a second. If a display is refreshed 100 times a second (typical of modern screens), more than a billion instructions can be processed in that time. To the observer of the screen it could look as if several things have happened all at the same time whereas they were actually processed one after the other. In the early days of computing performance was governed by the processor clock speed. Make the clock faster and the computing power increased but so did the heat generated. In order to operate at faster speeds the components had to get smaller so processor components became more densely packed. This makes the heating problem even worse. Removing the heat fast enough without frying the processor requires large heat sinks and fans. Apart from the waste of energy, this situation is also totally useless for mobile technologies that are dependent on batteries because the battery life will be insufficient for practical use as the processor churns out heat. (Good opportunity here to link to science, environment and conservation of energy) Most of the effort is now to make processors more efficient and to give them multiple cores - effectively several processors on one chip. Of course writing software more efficiently will also help. If I can find a way of making savings in the number of instructions executed in my program to get the same outcome it will be no different from improving the performance of the hardware.

Programs are normally executed from very fast memory actually on the processor itself (called a cache). So instructions and their associated data are stored in different places at different times. The program will be stored e.g. on a hard disk, pulled into the computer's main memory but the codes being executed will be in faster memory better matched to the speed of the processor. Why not just run all the memory faster? Cost. It is very much more expensive to manufacture faster memory so it is a trade off between capacity and speed at a particular price point. Hard discs are slow compared to RAM but 1000 GB of hard disc space is a lot less expensive than 1000 GB of RAM. With mobile technologies hard discs consume more power and are relatively bulky and the devices can store their information on the internet. At the time of writing there appears to be a transition taking place

from desktop and laptop computers to mobile technologies based more on power efficiency and small energy efficient clients to internet based resources. These are all areas worth exploring as they will have an increasingly significant impact on learners in the coming years. The main requirement for learners for this criterion is to understand the basic concepts of storing instructions in a program and executing them in a program flow.

In a sense, writing any program code is using codes to represent instructions and data. The HTML tag <h1> is a code telling the browser to make the following text adopt a particular style. </h1> is telling it to stop. HTML is a good way of introducing these concepts because it is practical, simple and accessible. will make the text appear bold and switch this off. Characters between the tags are data. They are simply displayed as written. So the tags represent instructions and what is between the tags is data. HTML is a mark-up language rather than a programming language because it doesn't have structures such as loops and conditional statements. These are added by using Javascript. (Note Java script is nothing to do with Java!) For the purpose of this criterion it is sufficient for learners to understand tags are codes representing instructions and anything that isn't a tag is data.

2.7 I can identify situations where codes control events

The candidate should be familiar with machinery that is controlled by code and measuring instrumentation that is automated.

Evidence: Assessors observations, portfolios

Additional information and guidance

Typical examples are 3D printers, robots, CAD/CAM, working models such as Lego, Fischertechnik and similar kits. Candidates should preferably have some hands on experience of programming practical control systems where the code controls physical events. It is true to say that there is an element of this in general purpose computing since software controls mechanics in printers, disc drives and other peripheral devices, however, breadth of experience will help in transfer of learning to unfamiliar and new situations. This would be a good opportunity for links to science in terms of measuring and recording data through experimental investigation and in creating products in design and technology.

3. The candidate will use programming languages

3.1 I can originate code in a visual language

The candidate should be able to originate some useful code in a visual language such as Scratch or Blockly.

Evidence: Portfolios

Additional information and guidance

Candidates should be encouraged to code complete projects but given time constraints they could adapt and modify existing code as long as they can originate at least some of the code themselves. The challenges in Blockly [7] at are a good way to get started.

3.2 I can originate useful code in a text based language

The candidate should be able to originate some useful code in a text based language such as Logo, python, BASIC, Javascript or Java.

Evidence: From candidate's source code in portfolios.

Additional information and guidance

Candidates should be encouraged to code complete projects but can be provided with sufficient structured guidance as long as they can originate at least some of the code themselves. Blockly is free from the web and enables export of code to Javascript and Python. This provides a good potential means of transition from visual programming in Blockly itself to programming in Javascript or Python.

3.3 I can identify structure in programs

The candidate should be able to identify variables, procedures, loops and conditional tests in existing programs.

Evidence: From assessor observations, internal tests.

Additional information and guidance

Candidates should be given opportunities to review source code that is sufficiently well documented for them to identify key structures and decide how they relate to the outcomes produced when the program is run. They should be encouraged to adopt good structural practices in their own work. They should be able to see how tables of data can be separated from program instructions and how a very large and complex program can be made up from many much simpler components. They should become aware that good structure and clarity are desirable e.g. by using peer review. Do other people understand their work and do they understand the work of others?

3.4 I can test code

The candidate should be able to use pauses in program code to identify the places where errors occur.

Evidence: From assessor observations and documentation in portfolios

Additional information and guidance

Being systematic in approach to finding and isolating bugs is the aim. Candidates at level 1 will need clear and concise instructions to support their work. This work could be related to the work on comparing algorithms and also to control of variables.

3.5 I can edit source code to fix a bug

The candidate should follow instructions to make simple edits to source code to fix bugs.

Evidence: From assessor observations and documentation in portfolios

Additional information and guidance

Level 1 candidates will need support and clear instructions in any but the very most straightforward cases. A key point is to immediately check any edit so that the effect of the edit is obvious. Avoid trying to fix several bugs at the same time since additional bugs can be introduced unintentionally and then it is not so obvious which edit was responsible. This might be a good time to introduce the concept of "many eyes making bugs shallow" and encourage working in pairs. They should also appreciate the importance of access to the source code if bugs are to get fixed, and also to really understand what a program is doing.

3.6 I can choose variable names that aid clarity

The candidate should spend time choosing variable names that are meaningful in the context of their code.

Evidence: From source code in portfolios

Additional information and guidance

This is all part of documenting source code to enable maintenance and development. Other people usually take over code from the original author and so it is important for them to be able to quickly understand the software. At one time there was an argument for short labels and names because the space available to run programs was very limited. This is no longer the case and the only reason for using short and cryptic variable names is laziness.

4. The candidate will understand Boolean Logic

4.1 I can predict the outcome of statements containing ABD, NOT and OR

The candidate should be able to predict the outcomes of simple conditional statements containing each operator.

Evidence: From internal testing.

Additional information and guidance

Conditional statements are of the the form IF <condition> THEN <do something> e.g. IF 3 > 2 THEN PRINT "true". Involving the Boolean operator AND could result in IF 3 >2 AND 8 >6 THEN PRINT "true" AND requires both conditions to be true. If we replace AND with OR true will be the result of either condition being "true". These conditions are more powerful when the numbers are variables. eg IF A > B AND X > Y THEN PROCEDUREgreaterthan. So if A is a bigger number than B and X is a bigger number than Y do some processing contained in a PROCEDURE called greaterthan. NOT effectively inverts a condition. IF NOT A>B PRINT "B is bigger than A". Candidates should be able to predict the results of simple statements of the type presented here. This will need some practice. Some clear supporting resources are at HowStuffWorks [8] and Cut the Knot Game [9].

4.2 I can include AND, NOT and OR in information searches

Candidates should be able to use the equivalents of AND, NOT and OR in searches using a particular search engine.

Evidence: From assessor observations and internal tests.

Additional information and guidance

Modern search engines are very good at finding relevant information without much need to know about Boolean Logic however they normally provide ways of including operators in searches. As an example, Google provides the following information [10]. The whole indexing of searches and the algorithms used to work out relevance is very complex. This link [11] provides and overview. At this level candidates should realise that searching a list is easier if you sort it first. The activities in Computer Science Unplugged [9] on algorithms is relevant here.

4.3 I can identify reasons why some search results are likely to be more important than others

The candidate should know some factors that search engines use to decide which sites are the most important.

Evidence: Internal testing, portfolios.

Additional information and guidance

Although not published in detail, Google uses over 200 factors in ranking the importance of sites for searches. These include.

Links – links to and from the site, the quality of those links, and ratios between links and even links with pages on the site and how it interrelates and how often it changes and the rate of change.

Site content – the content of the site, keyword density and interrelationships of content on the page and content within the site itself and how it interrelates and how often it changes and the rate of change.

Visitor related factors – how many visitors return, how many visits the site receives and the rate of change (increase or decrease) of new visits.

Domain name factors – How long the domain name has been registered for and how long it has been owned. How many times it has changed ownership.

Why does Google not publish its search algorithms in any detail?

4.4 I can relate boolean logic to program flow

Candidates should be able to identify the use of Boolean Logic in existing code, saying broadly how it affects the routes of program execution

Evidence: From assessor observation and portfolios

Additional information and guidance

The code studied should where possible have relevance to the candidate's interests, projects or other work. Conditional statements such as IF, WHILE, CASE, etc. More of this here [12].

4.5 I can use wild cards in searches

Candidates should be able to replace parts of words with wild card characters when performing searches.

Evidence: From assessor observations

Additional information and guidance

The most appropriate place to demonstrate this is likely to be searching for files on local systems. Wild cards tend not to be very useful in general search engine type searches because the number of matches is likely to be enormous.

4.6 I can represent numbers using binary patterns

Candidates should be able to represent decimal numbers by binary patterns and use codes to represent letters and characters.

Evidence: From assessor observations, local tests and portfolios,

Additional information and guidance

There is a good introduction to binary numbers here [13]. More resources at 1 [14], 2 [15], 3 [16], and 4 [17].

At the most fundamental level all instructions and data are stored as voltages. A 0 voltage is zero, and voltage significantly higher than zero is 1. All the sophistication and complexity of all computer systems boils down to this. The 1s and 0s are called binary digits or bits. 8 binary digits makes a byte. If we set 8 wires to each have 0 volts we have 0 and if we set each to say 5V we have 11111111 = 255. Why 255? Because there are 255 possible patterns using 1s and 0s on 8 wires. Every time we add another wire we double the number of possible combinations. So 9 wires will give

512 and 10 will give 1024 and so on. (See CS unplugged for binary activities) The original microcomputers popular in the early 80s were 8 bit. Now the norm is 64 bit. No great fundamental difference except that if each instruction is represented in 8 bits, a 64 bit processor could process 8 at the same time. That is quite apart from the clock speed increases to make them go faster.

8 bits has a special significance in computing because it is the fundamental unit for many codings. e.g. all the numbers, letters of the alphabet and characters such as £,*,$ etc each have a unique 8 bit code. A letter "A" is for example code 65, B code 66, the space bar code 32. We don't need to remember these because computers do the translations automatically although it is easy to find tables of them http :// www. ascii. cl/ Computers are in fact translating the binary patterns to the numbers and letters we see on the screen. The decimal numbers are really just because most humans are more familiar with them. So a set of 8 wires representing the letter A or code 65 would be 01000001. where 0 is zero volts and 1 is a higher voltage. Storing the letter A in RAM or on a hard disk means there is a tiny area on the disc or in the chip with that pattern on it in some physical form. If we could make them say ionised and unionised atoms, each letter could be represented in the space of 8 atoms. Currently it takes about 10 million times as much space. The theory is a lot easier than the practical technology though!

Candidates should be able to tackle simple problems such as if I have the bit pattern for the letter A 01000001 how would it change to a letter B given that A is 65 and B is 66? We need the next number up so it is 01000010. They should appreciate how counting in binary works.

They can make secret messages in binary and use the converters to decode them. Of course not that secret if other people have access to the decoder. They could of course write some sort of binary encoder e.g. changing all the 1s to 5s and 0s to 4s so that a binary decoder would not work unless they first put the message through something to convert the 5s and 4s back to 1s and 0s.

A lot of scope here for coding games, secret messages and similar.

Moderation/verification

The assessor should keep a record of assessment judgements made for each candidate guided by the above guidance. Criteria should be interpreted in the context of the general descriptors of QCF Level 1 qualifications. They should make notes of any significant issues for any candidate and be in a position to advise candidates on suitable routes for progression. They must be prepared to enter into dialogue with their Account Manager and provide their assessment records to the Account Manager through the on-line mark book. They should be prepared to provide evidence as a basis for their judgements through reference to candidate e-portfolios. Before authorising certification, the Account Manager must be satisfied that the assessors judgements are sound. In the event of missing evidence, the assessor will be requested to gather appropriate information before the award can be made.

Links

[1] http://theingots.org/community/Computing_qualification_info_units
[2] https://theingots.org/community/sites/default/files/uploads/site_icons/handbook-computing-L1-L2.jpg
[3] https://inkscape.org/en/download/
[4] http://numptyphysics.garage.maemo.org/
[5] https://opensource.com/life/16/9/coding-raspberry-pi-web-emulator?sc_cid=701600000011te3AAA
[6] http://csunplugged.org/searching-algorithms
[7] https://code.google.com/p/blockly/
[8] http://computer.howstuffworks.com/boolean1.htm
[9] http://www.cut-the-knot.org/game_st.shtml
[10] https://support.google.com/websearch/answer/136861?hl=en
[11] http://www.google.com/intl/en/insidesearch/howsearchworks/thestory/

[12] https://en.wikipedia.org/wiki/Conditional_(computer_programming)
[13] http://csunplugged.org/activities
[14] http://www.convertbinary.com/
[15] http://http://home.paulschou.net/tools/xlate/
[16] http://http://www.mathsisfun.com/binary-decimal-hexadecimal-converter.html
[17] http://http://nickciske.com/tools/binary.php

(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');