International Grades – Open Technologies

High Quality Qualifications for the
2016 School League Tables

The specification for

TLM Level 3 Diploma in Open Systems Computing (QCF) including a Level 3 Award and Certificate

Ian Lynch

ISBN 978-1-291-59755-4

Supported by the Lifelong Learning Project of the European Union through INGOT, SAFE and HANDSONICT projects

This is version 1.0 of the specification for TLM/NAACE Level 3 qualification in Open Systems Computing developed in partnership with the Open Source Consortium and Linux Professional Institute.

The assessment model for the qualification presented in this publication was designed by TLM in consultation with NAACE.

The Learning Machine Ltd, Unit 4D Gagarin, Lichfield Road Industrial Estate, Tamworth, Staffordshire, B79 7GN (www.theingots.org)

NAACE, The Sir Colin Campbell Building, University of Nottingham Innovation Park, Triumph Road, Nottingham, NG7 2TU. (http://www.naace.co.uk)

Contents

1. Introduction

1.1 The internet, technology based on open systems and computational thinking, is the most significant global innovation in our lifetimes. These new qualifications provide the scope for candidates to show what they can do through a new Open Systems Computing curriculum at Level 3. While they are primarily technical qualifications they also reflect the communities of people and working methods that have supported the innovation resulting in the the most profound and most rapid economic changes in history. The dominant operating system in mobile computing is Android – Open Source and based on Linux. The most dominant networking software is the World Wide Web, based on open standards and dominant open source applications such as Apache Web Server. 96% of the world's super computers run on open source software. If we want our dominance in processor design (ARM) to be reflected in software it is essential for students to understand not just the technical but also the commercial, methodological and ethical issues that make turning theory into practice possible. Purely academic computer science is only part of the story. The list of innovators that dropped out of university to start multinational companies is lengthy. Bill Gates (Microsoft), Steve Jobs (Apple), Mark Zuckerberg (Facebook) are just three. This why these qualifications are technical vocational qualifications, not academic qualifications in Computer Science even though they can form a basis for further academic study on that field. They fit well to the 2010 SOC classification 15-1132 and 1133.

1.2 The success of the internet is that it is not owned by any specific commercial or political interest. W3C, HTML, TCP/IP and a whole string of the core internet technologies are open to everyone to use. These qualifications recognise these facts as particularly appropriate for education because it enables learning through participation that is free from particular commercial and political interests.

1.3 In addition, there is a clear intention to reduce the bureaucratic overhead on teachers while preserving the benefits of coursework for

motivating learners and dealing validly with recognition of practical competence in what are essentially practical, technical and vocational skills and activities. We have demonstrated that we can provide Level 1 and Level 2 qualifications that are accessible to all learners while still differentiating the top performing students. This enables a clear progression route for the weakest mainstream learners through to Level 3 attainment and preparation for computer related courses in higher education as well as direct employability in digital technologies occupations, especially those where the growing open source and open systems contexts are important.

1.4 Open Systems Computing is a qualification that offers teachers and learners the opportunity to develop a range of knowledge, understanding and skills fundamental to successful engagement in the professional aspects of the digital technologies industry. This is contextualised in contemporary open systems and the communities that develop and maintain them. It is now widely acknowledged that open systems are necessary to enable a competitive and effective marketplace. Content coverage includes a wide range of technical knowledge, understanding and competences related to digital technologies, from computational thinking through programming to systems management. Progression routes from Level 1 through Level 2 to Level 3 will benefit any young person aspiring to a professional career in the digital technologies sector either as a result of further study, apprenticeship or direct employment.

1.5 This specification is for Level 3 Award, Certificate and Diploma building on Level 1 and 2 Certificate specifications that are already accredited. We plan Entry level qualifications to underpin these later in the year and these might be in place by the time you read this. The qualifications target secondary schools and further education colleges but are also suitable to be used in apprenticeships. They have the following key benefits.

- devised in consultation with leading industry consultants, professional bodies and universities.

- clear and flexible unit based structure referenced to the European Qualifications Framework (EQF).
- straightforward assessment of competence in real rather than contrived contexts.
- grading through controlled exams introduced progressively from KS4.
- provides a focus for continuing professional development for teachers through moderation/verification feedback.
- moderation/verification of coursework on demand.
- two examination opportunities per year for Certificate and Diploma.
- use of open source cloud based technologies to reduce costs and add value for schools.
- reduced bureaucracy for teachers and flexibility for them to target specific interests.

1.6 These qualifications lend themselves to formative assessment practices allied to competence based and summative differentiation by outcome that can optimise and motivate attainment for individuals rather than assume all will reach a certain level or grade at a particular time. We do this by providing a coursework component that is competence based, reflecting the best and most up to date research in assessment in the workplace, complemented by an academic multi-part examination for the Certificate and Diploma.

1.7 All candidates must complete the coursework before being eligible to take an exam. This provides an incentive to complete the coursework and makes it less likely that those sitting an exam fail through inadequate preparation. It provides flexibility to encourage innovation and the use of up to date contexts but limits the influence coursework can have on achievement of the highest grades.

1.8 The Level 3 exams grade candidates across a range from Pass to A* with grades A*, A, B, C, D, E, available as pass grades. If coursework is completed to the Level 3 standard, in keeping with the

Level 3 general descriptor, the candidate can go on to take the Level 3 exam which will then differentiate grades A*-E.

1.9 In this way we can provide valid competence based assessment and rigorous testing of underpinning knowledge and understanding at a lower cost than both traditional vocational and academic methods applied separately. There is research evidence that this approach should enhance motivation that will result in higher attainment by supporting both performance-approach goals that focus on displaying competence and performance-avoidance goals that focus on avoiding a display of incompetence. (Conclusions from Effects of Classroom Assessment Practices on Students' Achievement Goals, Hussain Alkharusi Sultan Qaboos University, Oman.)

2. Summary of the qualification specification

2.1 The Level 3 Award is ungraded and entirely coursework based. It is based on the candidate satisfying the coursework requirements for any of units 1, 2, 4 or 5 together with the L2 IT Security for Users unit.. See annexe D. The Level 3 Certificate is based on the coursework for units 2 and 4 and a grading examination covering these units. The Level 3 Diploma in Open Systems Computing requires all five units and a grading examination drawn from all the units and associated under-pinning knowledge. The Certificate and Diploma are graded across 6 levels from A*- E with A* the highest grade equating to 80%+ of the available marks and grade E equating to a minimum of 50%. Candidates that complete the coursework to the appropriate standard at Level 3 will carry forward 30 marks. Those that achieve 20 more marks from the exam will be awarded a grade E (50%). 55% for a grade D, 60% for grade C, 70% for grade B, 75% for grade A and 80% or more for A*.

Content

2.2 The qualification content has been designed for use in schools and colleges by building it on L1 and 2 foundations that are referenced to the new National Curriculum programmes and testing it against similar assessments carried out in current Level 3 qualifications. It is also designed to enable learners to meet the needs of employers, through consultation with innovative small business employers, progressive universities and professional bodies representing a wide section of the industry. Guidance for coursework is aligned with the CBI employment criteria and the qualifications are therefore specifically targeted on employability.

2.3 Guidance takes into account the lack of experience of many teachers in this area ensuring that the most academically able can be stretched and where appropriate, routed to appropriate academic progression in Higher Education with up to date knowledge. Strong industry support provides great potential for staff development, keeping

teachers up to date in what is still a rapidly changing sector. NAACE has a large number of sponsoring companies from the digital technologies sector.

2.4 The use of real equipment and technology rather than simulations or generalisations provide the real world contexts needed for motivation. There is an emphasis on increasing understanding of the importance of collaborative working using open systems in keeping with recent Cabinet Office policy and the Government Industrial Strategy 2025 and policies to shift government systems to Open Systems.

> "The cost of the government's IT is currently too high and needs to be reduced. There is a lack of market diversity in existing government contracts. A more diverse market and level-playing field for access to government IT contracts is needed to improve competition, reduce cost and improve public service outcomes."
>
> **(Open Standards: Open Opportunities - flexibility and efficiency in government IT:**
> https://www.gov.uk/government/consultations/open-standards-open-opportunities-flexibility-and-efficiency-in-government-it**)**

2.5 This demonstrates the relevance of the qualification to current government policy and employment in the public sector as well as private industry.

Assessment

2.6 The qualifications at Level 3 have two assessment components.

1. Coursework assessed in terms of competence in practical areas where knowledge and understanding can be applied in real and motivating contexts.
2. An externally set and externally marked examination to assess the knowledge and understanding that underpins user competence.

2.7 The Diploma qualification is unit based and consists of 5 units. Units have credit values in the qualifications and credit framework (QCF). A minimum of 50 credits is needed equating to 360 Guided Learning Hours and this provides 1 third of the core credit required for the Modern Baccalaureate at Advanced Level. The Certificate qualification consists of 2 units providing 20 credits equating to 150 GLH. The Award is 2 units, one at Level 3 and one at Level 2 equating to 12 credits and 90 GLH.

2.8 The synoptic examinations of knowledge and understanding that is used for grading in the Diploma and Certificate are based on a syllabus related to all the units for the particular qualification. The design does not allow candidates to compensate for weak coursework by doing well in the exam only or vice versa. The exam is providing an additional level of quality assurance for the evidence of competence provided for the units and it enables grading in order to inform academic and vocational routes in higher education. Candidates must complete the coursework to a satisfactory standard at level 3 to be eligible to take the examination in either the Certificate or the Diploma.

2.9 A weak examination performance will limit the attainment level graded informing the candidate of their likely success in further related academic study. It is likely that candidates that display competence in coursework will at least pass the exam but that is not inevitable and they must take the exam to pass overall as it is a mandatory aspect of the qualification as a whole. The exam then also provides an additional very low cost dimension to external moderation/verification feedback for the coursework. Centres with a high proportion of candidates judged to be satisfactory on coursework yet failing to gain sufficient marks in the examination flag up a need for further investigation and will help prioritise CPD.

Summary of the rationale

2.10 The assessment is specifically designed to motivate learning that will support the highest grade(s) attainable by each candidate but also broader aspects of learning that can not be assessed in a traditional

exam. Learners must demonstrate competence across the units before being eligible for the examination which is used in part as a confirmation of that competence.

2.11 There is considerable flexibility to enable contexts of individual interest to be explored in depth. Those that have completed the coursework in areas of personal interest and to a high standard are far less likely to fail to achieve at least the minimum standards set in the examination. This ensures basic practical competence in realistic and motivating scenarios as well as at least some retained general knowledge and understanding in the more academic sense to enable skills transfer as technology changes. This is essential in a 21st Century workforce as there is never going to be sufficient resource to keep retraining everyone every time something changes.

Aggregation of marks

2.12 For the Diploma, coursework across 5 mandatory units contributes 30 marks, the examination is worth 70 marks. Candidates pass the Diploma with grade E when they achieve 20 marks in the examination on top of the 30 acquired from the coursework, grade D when they achieve 25 marks, grade C for 30 marks, grade B 40 marks, grade A, 45 marks and grade A* for 50 marks. This equates to 50% through to 80% of the marks in the qualification as a whole. Candidates will be provided with their marks as well as their grades. Candidates can take the examination when their assessors judge that they are ready and when they have completed the coursework to a Level 3 standard across the required units.

2.13 The examination questions get progressively more difficult starting with 40 multiple choice questions, then in part 2, 5 short open response questions and then 4 long open response questions. Those achieving the highest marks will be those most likely to cope with further academic higher level study eg at university. Those that produce sound coursework but low exam scores are more likely to be suited to further vocational training, direct employment or apprenticeships.

2.14 A candidate that completes the coursework to a satisfactory standard at Level 3 but fails to gain sufficient marks in the examination can re-take the exam. the next time it is available.

2.15 In the interests of inclusion, there will only be additional fees for additional examinations taken since the coursework cost will already have been paid. An optional subscription model that covers the family of Open Systems Computing qualifications means that schools can enter as many candidates as they believe can meet the criteria and there are no hidden costs such as late entry fees, double entries or replacement certificate fees. This maximises the opportunities for learners to get their achievements recognised without the school worrying about financial penalties and providing the savings associated with economies of scale.

3. Qualification Content

3.1 The qualifications are made up from units in the Qualifications and Credit Framework (QCF). The QCF is referenced to the European Qualifications Framework (EQF), the largest system for referencing nationally accredited qualifications in the world. Unit credit is designed to be compatible with the European international credit transfer system ECVET. The units were designed by TLM in collaboration with teachers currently working in the classroom, industry consultants, professional bodies and universities. In order to provide learners with the knowledge and skills needed by all sector employers and to ensure current government policy for the use of digital technologies in the public sector can be supported.

3.2 Extensive consultation with business leaders has taken place. This specification is a distillation of this extensive market research specifically geared to supporting learning in schools and colleges. There is an emphasis on developing the transferable knowledge, skills and competences in open systems computing that will support lifelong learning providing the grounding needed for future hi-tec professionals. There are references to science and mathematics especially in terms of control of variables, number systems, logic and measurement. Specialist vocabulary with words such as abstraction, algorithm, hexadecimal, and Sourceforge, will help support technical English at a level beyond that of most adults.

Key subject aims

3.3 The overarching aim is to enable learners to broaden their understanding of technical and professional procedures associated with software engineering and the contextual background needed to work efficiently and effectively in these occupations. Those seeking careers in a digital environment will have an appropriate grounding in collaboration and computing to enable them to make rational decisions about their progression routes into employment in this sector as well s going on to further study.

Subordinate aims include:

- developing the knowledge and skills needed for employment.
- gaining practical experience needed to underpin lifelong learning.
- increasing the knowledge needed to transfer skills and understanding between contexts.
- reinforcement of learning in the core subjects of English, mathematics and science.
- developing practical skills in creativity and problem solving in technological contexts of personal interest.
- developing an understanding of their place in the community and society.
- developing safe, secure and responsible attitudes to working with other people.
- developing the skills to working collaboratively.
- developing knowledge in the field of critical evaluation and feedback.

Knowledge and understanding

3.4 The following knowledge and understanding will be required to underpin the desired learning outcomes.

A strong candidate will be able to demonstrate knowledge and understanding associated with a wide range of computing terms.

- The list below is not exhaustive but provides an indication of the breadth expected. Some terms will be analysed to a much greater level of detail than others but candidates should have some idea of basic definitions and contexts for as wide a range of these terms as possible for the Diploma. Flexibility in the assessment model will allow more in-depth study in areas of particular interest. For the Certificate and Award only those words associated with the specific learning outcomes are required.

Access point, Analogue-to-Digital Converter (ADC), Asymmetric Digital Subscriber Line (ADSL), Algorithm, Alpha Software, ANSI American, National Standards Institute, API Application Program Interface, Archive, ARP Address resolution protocol, Adware, Array, ASCII, Backup, Bandwidth, Base Station, Batch Process, Bcc, Beta Software, Binary, Binary Coded Decimal, BIOS, Bit, Bitmap, Bitrate, BitTorrent, Blog, Bluetooth, Bookmark, Boolean, Boot, Bot, Broadband, Browser, Buffer, Bug, Bus, Byte, C/C++, Cache, Captcha, CCD, Checksum, CISC, Clock Cycle, Clock Speed, Cloud Computing, CMOS, CMYK, Codec, Commercial Software, Compiler, Computer Ethics, Contextual Menu, Cookie, Copyright, CPU, Cron, Crop, Cross-Browser, Cross-platform, CSS, Cyberbullying, Cybercrime, Cyberspace, DAC, Daemon, Data, Data Transfer Rate, Data, Type, Database, DBMS, Debug, Debugger, Deprecated, Default, DHCP, Dialog Box, Digital Signature, Dithering, DNS, Domain Name, Domain Suffix, Driver, DRM, DTD, Error Correction Code (ECC), Electronic Data Interchange (EDI), Emulation, Encryption, EPS, Ethernet, Extranet, FAQ, FAT32, Fibre-Optic Cable, FIFO, File, File Extension, File Format, File System, Filename, Firewall, Firmware, Flash Memory, Flat File, Floating Point, FLOPS, Flowchart, Folder, Format, FPU, Freeware, FSB, FTP, Gateway

Gigabyte, Gigaflops, Gigahertz, GNU, GPS, GPU, GUI, GUID, Hacker, HDMI, HDTV, Heat Sink, Hertz, Heuristic, Hexadecimal, Host, HTML, HTTP, HTTPS, Hyper-Threading, Hyperlink, Hypermedia, Hypertext, I/O, ICANN, Icon, ICT, IEEE, Illegal Operation, IMAP, Integrated Circuit, Intellectual Property, Interface, Internet, Intranet, IP, IP Address, IPv4, IPv6, IRQ, ISO, ISP, Iteration, Keylogger, Keywords, Kilobyte, LAN, Latency, LCD, LDAP, LED, LIFO, Linux, Load Balancing, Localhost, Log, File, Logic Gate, Login, Lossless, Lossy, MACAddress, Macro, Mainframe, Malware, Markup Language, Mbps, MBps, Megabyte, Megahertz, Megapixel, Memory, Memory Leak, Meta Search Engine, Meta Tag, Metadata, Metafile, MIPS, Mirror, Mnemonic, Modem, Motherboard, Mount, Mouse, MP3, MPEG, MTU, Multimedia, Multiplatform, Multiprocessing, Multitasking, Multithreading, Name Server, NAS, Native File, NetBIOS, Netiquette, Netmask, Network, Network, Topology, Node, Null, Null Character, Nybble, OASIS, OCR,

ODBC, OEM, OOP, OpenClipart, Open Firmware, Open Source, OpenGL, Operating System, Overclocking, P2P, Packet, Parallel Port, Parse, Partition, Password, Path, PDF, Peripheral, Perl, Permalink, Petabyte, Petaflops, Pharming, Phishing, PHP, Ping, Pipeline, Piracy, Pixel, Plain Text, PNG, POP3, Port, PostScript, Power Supply, PPGA, PPP, Primary Key, Primary Memory, Process, Processor, Program, PROM, Protocol, Proxy, Server, Pseudocode, Python, RAID, RAM, Raster Graphic, Raw Data, Readme, Real Number, Real-Time, Recursion, Recursive Function, Repeater, Resolution, RFID, RGB, Rich Text, RISC, ROM, Root, Router, RSS, RTE, RTF, Ruby, Samba, SAN, SATA, Schema, Script, SCSI, SD, SDK, SDRAM, SDSLSearch Engine, SEO, Serial Port, Shareware, Shell, Site Map, Skin, Smartphone, SMB, SMS, SNMP, SOAP, Socket, Solid State, Source Code, Spam, Spider, Spoofing, Spyware, SQL, SRAM, SSH, SSL, Stack, String, Subdirectory, Subnet Mask, Superscalar, Surge Protector, Switch, Sync, Syntax, System Analyst, Tag, TCP/IP, Telnet, Terabyte, Teraflops, Terminal, Text Editor, TFT, Thin Client, TIFF, Torrent, Traceroute, Trojan Horse, Troll, TTL, U, UDP User, Datagram Protocol, Unix, UPS, URI, URL, USB, User Interface, Username, Vector, Vector Graphic, Virtual Memory, Virtualization, Virus, Visual Basic, VoIP, VPN, VRML, W3C, WAIS, WAN, Waveform, Web 2.0, Web Host, Web Page, Webcam, Webmail, Webmaster, Website, WEP, WHOIS, Wi-Fi, Wiki, Wildcard, WiMAX, Windows, Worm, WPA, WWW, XHTML, X86, XML, XSLT, Zip.

- Demonstrate mathematical knowledge associated with number systems and logic.
- Demonstrate mathematical knowledge sufficient to be able to model simple real world systems, for example a falling object or more complex situations given the relevant mathematical relationships in the form of formulae.
- Demonstrate scientific knowledge and understanding associated with digital technologies including energy, rates of change, analogue and digital systems, conductors and insulators and transducers.

- Demonstrate the application of understanding digital systems including analogue to digital conversion and digital to analogue conversion, how digital information is stored, transmitted and received, made secure and validated.
- Use fundamental knowledge and understanding to analyse and evaluate systems that might or might not be familiar.
- Use knowledge of open systems to make better decisions in relation to risk and achieving value for money.

3.5 Opportunities are provided to support real skills, the great majority of which will be assessed directly in coursework in valid contexts. For the Diploma, candidates will carry out a major software project based on a real and significant need. A range of appropriate tasks follow the journey building their software application including;

1. Understanding contemporary software development techniques.
2. Working collaboratively and promoting community cohesion.
3. Building information modelling skills.
4. Researching and finding out how to do specific tasks as needs arise.
5. Consideration of efficiency and effectiveness.
6. Developing in-depth knowledge of a computer programming language.
7. Receiving and acting on evaluation feedback.

Although not mandatory, it is recommended that able candidates make contributions to an open source project to evidence their programming competence. Demonstrating competence in a real world context with serious peer review and for a genuine need is the best way of proving their knowledge and skill level. Help and support can be provide from TLM/NAACE on this.

Unit contents

3.6 The content of units is in Annexe C below with some examples of interpreting the criteria. These are available in more detail on the TLM community learning site and will be linked to progressively more free

and open supporting resources and guidance as these become available.

3.7 All centres have an assigned Account Manager who will be very pleased to help at any time. Our aim is to give professional assessors, most of whom are qualified teachers, the confidence to make judgements with a minimum of bureaucracy so that they can focus their time on maintaining their professional knowledge and skills and support learning through effective teaching rather than “chasing paper”.

3.8 There is often a confusion between bureaucracy and rigour, since unnecessarily complex bureaucracy can actually detract from rigour by obscuring the importance of the outcomes in unnecessary process. This is embedded in the philosophy of Agile software development and so directly relevant to these qualifications. All assessors must sign an agreement to uphold standards and feedback from moderation/verification will support consistency.

3.9 Websites - TLM provides support through a cloud based system for evidence management linked to grading and certification. Providing assessment grades and the management of certification through the Awards Site is mandatory and all assessors are provided with training in its use. It is simply a matter of recording learner competence against the unit criteria as the evidence is collected and claiming a certificate on behalf of the learner when a unit has been fully assessed. All assessors must sign an agreement to uphold standards before they can use this site.

3.10 The use of the community learning site is optional at no additional cost. It provides facilities for learners to submit their evidence online, linking it to the assessment criteria across single or multiple units. The assessor can accept or reject this evidence and comment on it providing a full audit trail for evidence. Moderator/verifiers can get immediate access to this evidence and so it is potentially a lot more efficient than alternative methods. No paper, no emails with file attachments are necessary. There are facilities for progress tracking

that can be based on criteria and/or units and reports that can be shared securely online eg with parents or employers. The system can be linked as an extension to any standards compliant VLE/e-portfolio system for centres that are already committed to a specific VLE product. Training can be provided and free support is available from your Account Manager. The aim is to eliminate all paper based bureaucracy, all screenshots and referencing that draws time away from teaching. As far as possible we want assessment of real tasks in real contexts that are truly representative of a real working environment with the candidate at the centre of that assessment. This is a fundamental goal for the competence based assessment at the heart of the Qualifications and Credit Framework and European Vocational Education and Training policy (ECVET). It is the way in which most employers will judge the effectiveness of individuals in their tasks at work.

3.11 **Telephone** and e-mail support is available to all Centres. There is a general convention of firstname.secondname@theingots.org for e-mail addresses. It is usually best to e-mail your account manager in the first instance. Google hangouts can be arranged for video conferencing support.

4. Assessment

Assessment summary

Coursework

4.1 Evidence has to be provided against the unit assessment criteria from practical tasks related to the learners' everyday work. This is likely to be from specialist lessons supported by a subject specialist but links with A levels and other equivalent subjects, for example from maths, science, computing art and design and other relevant subjects is to be encouraged. The way evidence is gathered is up to the assessor, the only requirement is that it clearly supports the judgements against the assessment criteria and the relevant learning outcomes and reflects the learners personal competence in line with the guidance in this handbook and the general description of Level 3 qualifications provided in the QCF. If on moderation the account manager finds gaps in evidence related to a particular candidate they will request more evidence before agreeing that the coursework criteria have been met. Assessors must then adjust their work to ensure all their learners are providing the appropriate level and breadth of evidence.

4.2 We encourage early submission of at least some evidence so that assessors are confident from the feedback that what they are providing is sufficient (and indeed not over-kill). In this way we can maintain standards while supporting improved efficiency.

4.3 Synoptic assessment has become a popular term. In essence all the coursework assessment is synoptic in that the evidence provided is against collectively synoptic assessment criteria underpinning the learning outcomes for the unit. Synoptic evidence of competence across all the units is mandatory for the particular qualification. This equates to a minimum of 360 guided learning hours for the Diploma, 150 for the Certificate and 75 for the Award.

4.4 For the Diploma, there are 5 units each of 10 credits and requiring 75 GLH except for the practical programming unit which due

to its nature requires fewer. Dividing into a unit structure is for convenience and compatibility with international conventions for referencing national qualifications frameworks and to enable credit transfer eg as in the European system ECVET. It is **NOT** intended to determine the method of delivery. Teachers are free to cover units concurrently deciding where the elements are logically related or sequentially as some content comes naturally before others eg research before evaluation. We encourage the use of the flexibility provided to target particular interests of learners, to motivate them in persevering in difficult areas and to raise the level of expectation in cognitive development.

4.5 The central project within the Diploma curriculum is to develop a software application that is useful to other people, preferrably as part of an open source community. This could be improving or building upon an existing application or it could be an original piece of work starting from scratch. As long as there is evidence against the criteria in keeping with Level 3 the candidate will be successful.

4.6 There is an obvious progression from Level 2 to Level 3 where learners will be expected to tackle academically more challenging questions requiring analysis and quantitative skills. The Award and Certificate can be used as stepping stones to the Diploma or treated as qualifications in their own right. The outcomes for individuals in terms of the broad level descriptors allied to the assessment criteria, verified by the teacher/assessor and externally moderated by TLM will determine the final outcome.

Progression and inclusion

4.7 There are some fundamental misunderstandings of unit based assessment with regards to progression and inclusion. The paragraphs below will explain how criticisms related to these issues can be rejected. It is mainly an issue of having higher levels of professional expectation and better CPD strategies rather than simply “dumbing down” to less professional approaches.

4.8 This qualification provides a unique project based learning opportunity in the context of open source development. There is support for numeracy skills through applied mathematics and literacy through practical communication, providing a general education as well as the specifically vocational elements. This ensures that those that take these qualifications but decide against a career as a specific digital technologies specialist, are well-placed to go on to specialise in other fields with the value of the knowledge, understanding and skills they take from their study broad enough to support a wide range of employment.

4.9 Open Systems Computing links to career progression into the digital technologies sector, from software engineering to systems management or any profession where a high degree of technological literacy is required. The computational thinking dimension lends itself to generic cognitive development but motivated by practical contexts.

4.10 It is very unlikely that any learner embarking on a TLM qualification based on these methods will not achieve at least some kind of recognition for their work at a level appropriate to their current attainment level with a progression route from where they end up to higher levels. Clearly some will take longer than others. This inclusion is achieved without sacrificing rigour for the highest attainers since the questions in the examination targeting the A/A* grades can be as difficult as necessary without risking weaker candidates dropping out of a grade altogether. Indeed able students can start Level 3 work in KS4 differentiated by outcome where appropriate. Currently there is a good argument that candidates achieving A* and A grades across all their subjects are not being adequately stretched and the natural progression from Level 2 to Level 3 can be used to tackle this directly.

4.11 For the highest attainers that gain some Level 3 credit early, it provides additional time to tackle more ambitious real world projects, with opportunities for making early links with universities who are only too pleased to help support the most able candidates. This is the rationale for the Award and Certificate. These can provide supporting

credit for the software project unit and can be used as milestones to enable candidates the research background necessary to participate in open source projects alongside professional level programmers.

4.12 Coursework, at Level 3 should reflect useful and meaningful activities with practical activities that add to the digital resources available to the wider community. An example might be to provide an added feature to an existing open source application and to document it. Another might be to fix identified bugs in open source code or to set up a new open source community project with a like-minded group. We want to encourage work that reflects contemporary society using standard tools and technologies freely available from the internet. We want to enable Level 3 learners to contribute to their potential alongside established professionals. Projects lend themselves to cross-curricular work supporting raising attainment in other subjects, numeracy, literacy, science and information skills but also aesthetic subjects such as art and design. It is far better to learn through creating original and useful work than to do simulations or theoretical exercises. This is a fundamental part of TLM's coursework philosophy and founded in research evidence.

Criticisms of coursework answered

Criticism 1: Coursework is too susceptible to plagiarism and other forms of dishonesty.

A Google search will have a high chance of finding any extended text that has been copied from an online source. If we are genuinely concerned about "copying from the internet" simply inform teachers of how to combat the issue using freely available tools. Require teachers to accept professional responsibility for the authenticity of their learners' evidence. If teachers really want to cheat why would they not simply tell students the answers to an exam question? If learners want to cheat why not simply forge a convincing looking certificate? There is no tradition of easy certificate authentication so there is a high probability that forgery will be successful. Using a complementary examination means that we can check back to see if individual teachers are

"passing" student coursework for a disproportionately high number that then fail the examination. That provides an evidence source to cross-reference the quality assurance in order to better target staff development. Work smarter not harder!

Criticism 2: Unit based assessment means that knowledge is in compartments.

Unit structures are for administrative convenience **NOT** teaching plans. There is nothing to stop elements of several units being supported through one or more projects concurrently. Most academic syllabuses are divided up into sections. That is no different in practice to labelling the sections units. There is no requirement to assess units at a particular time. If most evidence is provided at the end of the course across all units why is that any different in the principles related to timing from a controlled synoptic terminal examination? If on the other hand a unit is naturally covering work at the beginning or end of a project why not assess it at the appropriate time? Rationality is usually better than ideology in these contexts. If teachers do not teach unit based courses effectively, train the teachers, don't make blanket rules that might well be inappropriate in particular circumstances.

Criticism 3: Unit based assessment does not support progression.

On the contrary, the scope of unit based qualifications organised in a levelled framework provides a better support for progression when the unit content and structure is designed for that purpose. Where qualifications are opportunistically designed to simply target one level of fixed duration leading to a terminal examination that is only representative of a subset of the learning, there is a good argument that progession is badly supported but that is true of any qualification whether unit based or not.

Criticism 4: Competence based assessment has to be lowered to the level of the least difficult assessment criterion.

In well designed assessment units the assessment criteria are contextualised to the general level specified in the overall level descriptors. This means all assessment criteria should be interpreted in terms of that overall level descriptor. It is impossible to measure anything with absolute precision and it is scientifically bogus to claim we can, There will always be some uncertainty in assessment outcomes when applied to individuals and that goes beyond simple statistical variance. This is true of both coursework based and exam based methods. The important thing is to get a reasonably consistent set of outcomes within the expected degrees of uncertainty. The competence based component of these qualifications is intended to provide a baseline consistent with the level and to motivate beyond basic competence by providing the flexibility to pursue contextual interests. Grading is achieved by a terminal examination. This means we can match the assessment method to the aspect of attainment such that we cover all aspects of learning but we also provided reliable differentiation that can accurately inform progression routes for individuals.

Criticism 5: Exams have always been the tried and trusted way of assessing attainment. There is no need for anything else.

Written examinations have been widely used for academic assessments in schools and universities. However, that is largely due to their academic heritage where theory is often more important than practice. Even so coursework is well-established where there are practical elements eg in science and medicine. Few jobs assess prospective candidates exclusively using written exams. In most practical areas from brain surgery to teaching, no-one would trust a written examination on its own to prove competence. That is not to say such examinations are not of value. The key is to use coursework **and** examinations intelligently together in order to provide something that is better than either treated in isolation. Polarised arguments about one against the other are missing the entire point.

The Examination

4.13 Examinations at Level 3 are primarily for grading. The details of the way grades relate to marks are provided above in section 2. The examinations also provide a cross reference in order to increase confidence in the validity of the coursework component.

Weightings

4.14 There are two classes of objectives. AO1, AO2, AO3 are generic assessment objectives:

AO1 - Recall, select and communicate knowledge and understanding.

AO2 - Apply knowledge and understanding through analysis, reasoned judgements and drawing conclusions.

AO3 - Practical and technical skills related to applying skills knowledge and understanding in context.

4.15 Additionally, the qualification units each specify subject specific learning outcomes. The qualification design draws on both classes of objective to ensure balanced representation and that the assessment is a valid representation of what has been learnt.

4.16 The assessment objectives provided by the unit learning outcomes are evenly weighted in the coursework element since all must be achieved in order to pass.

4.17 The synoptic examinations are directly related to the unit learning outcomes and assessment criteria using the content definitions in section 3. This is designed to be broadly representative of the aspects of the learning outcomes testable in a synoptic terminal controlled examination related to the learning outcomes. The examination provides a means of testing associated knowledge and understanding, powers of analysis and reasoning and of grading the qualification whereas the course work ensures that there is basic competence in their practical implementation in real and relevant contexts.

4.18 At Level 3 the examination weighting of AO1 is 20% and AO2 80%. This is for both Certificate and Diploma.

4.19 The overall weighting of the objectives varies depending on the grade because for higher grades AO2 will contribute a greater proportion of the marks. This is a deliberate strategy because AO2 is the most important learning when it comes to academic learning in HE. The assessment will therefore better inform progression pathways while still having the characteristic of inclusion.

Grade E approximately weighted AO1 - 40%, AO2 - 40%, AO3 20%.
Grade A* approximately weighted AO1 - 20%, AO2 - 70%, AO3 10%

4.20 This then provides evidence that the Grade A* candidate is likely to be more suited to future academic study whereas the Grade E candidate is likely to find it difficult to cope with courses highly dependent on academic testing.

Learner entry and costs

4.21 TLM/NAACE subscription model enables schools to enter learners at times convenient to them. There are no late entry fees and no additional fees should a learner fail to produce evidence at a particular level but can meet the criteria at a lower level. This can reduce costs to the school by more than 50%. Examination entry will depend on whether or not learners meet the coursework criteria. This again saves money because the school is not paying for examination administration for learners that are unlikely to be successful or for whom there is little or no benefit in taking an exam. There are no fees for replacement certificates or verification of certificates because all certificates can be directly authenticated against a secure database. For details of current subscription costs please contact us or refer to the web site. All of these design features are intended to reduce direct costs but just as importantly the indirect administrative overhead that diverts teachers from teaching.

Online examination

4.22 The examinations can be delivered in a traditional paper based format or online. There is a surcharge for paper based examining reflecting the extra cost involved. The online versions have a secure web user interface and require no software installation. They can run through any standards compliant web browser on any type of computer. The user is restricted to an area in the centre of the screen during the examination and has no access to the internet, or any other storage device without moving the mouse pointer out of the secure area and this will set off a warning. Persistence will result in disqualification from the examination. Since the Level 3 online exams contain open-ended questions they have to be physically marked and so the results will not be immediately available but we will aim to have these ready within 2 weeks of taking the exam. For those taking the examinations in the traditional paper based format it is likely to take 4 weeks to finalise results.

Examination windows

4.23 The exams will be available in two windows per year in December, and June/July. It is the Centre's Principal Assessor's responsibility in line with the agreement signed with TLM to ensure that security is maintained for the examination. No candidate should have prior access to the questions in an examination paper either directly or indirectly, before they sit the paper. We will have several versions of the examination available and if there is any suspicion of compromise of security, the Principal Assessor should contact TLM to work out a solution. Assuming there is no malpractice, it might simply be a matter of scheduling an alternative paper. Papers will be planned to be of similar difficulty. Candidates can retake an examination at the following sitting if they have not claimed a qualification based on a previous result.

Internal standardisation of coursework

4.24 The Principal Assessor has the ultimate responsibility for consistency in assessment standards within a centre and has signed an

agreement to that effect. All assessors have signed a contract agreeing to uphold standards and should therefore co-operate with the Principal Assessor and Account Manager at TLM to ensure that standards across the centre are consistent. It is advisable to send work samples to TLM early to check that evidence is at the right standard so that there is time to make any adjustments necessary to the course and learner expectations. TLM will generally check a higher quantity of work from new assessors and feedback to ensure that they are confident to make appropriate judgements over time. This reduces risk and improves efficiency in the longer term.

Authentication

4.25 All assessors must take reasonable steps to ensure that any coursework evidence submitted by candidates is a true reflection of the candidates' competence. This is in keeping with the assessor undertaking to uphold and maintain standards in the contract with TLM.

4.26 Certificates can be authenticated directly online using the certificate number or by scanning the QR code on the certificate. There is no charge and it makes it more likely that certificates will be checked and that in turn improves security. Certificate forgeries are a significant problem when authentication is not simple and straightforward because convincing forgeries are easy to achieve with recent technologies and will get easier as time goes on.

5. Other considerations

Access arrangements and special requirements

5.1 All TLM's qualifications are intended to be accessible, as widely as possible. There is an extensive policy documented on the web site at https://theingots.org/community/QCF2.13
Centres should contact TLM if they have any questions related to accessibility issues.

Language

5.2 The language for provision of this qualification is English only. This will only change if we have a significant demand in another language that is sufficient to cover the additional costs involved and some cultural alterations will be needed. TLM will actively support any work in this line that can be shown to cover costs.

Malpractice

5.3 TLM has comprehensive policies and procedures for dealing with malpractice. These are documented with links on the web site at https://theingots.org/community/QCF5.29-5.32 Assessors should be familiar with these policies and make them clear to candidates. Assessors should inform their account manager if they suspect any instance of malpractice that could have a material effect on the outcome of any assessments, either for themselves or colleagues. This is part of the upholding of standards that is part of the contract with TLM.

Equality of opportunity

5.4 TLM promotes equality of opportunity through policies and procedures. These are again documented in detail on the web site at https://theingots.org/community/QCF2.11-2.14

Resources, support and training

5.5 A clear goal of these qualifications is to enable learners to support their own learning and to reduce dependency in order to become “lifelong learners”. The IT revolution makes this progressively easier especially with the growth of supportive online communities. As far as possible we encourage the use of technology and up to date methods especially those based on empirical evidence.

5.6 TLM encourages the use of free and open source applications to reduce costs and to further inclusion. However, students are at liberty to use any software they wish to support their coursework. Learning more about free and open source resources provides more choice and better informed decisions.

5.7 It is anticipated that the open source community paradigm will help teachers grow in confidence, develop their own networks of industry based support and be able to develop new projects of their own – ones that may be unique to their local context or that offer specific targeted challenges.

5.8 The curriculum introduces new areas of learning that include close engagement with the world of work and academia. Teachers and learners alike will find it rewarding, challenging and exciting – a combination that guarantees successful outcomes and a learning environment that is happy, productive and fun.

5.9 The qualification is designed to support learning that enables access to Further Education, Higher Education and employment for a wider range of young people.

6. Grade Descriptions

A **grade A** candidate will exhibit most the following characteristics.

6.1 Candidates demonstrate a high level of independence in using their knowledge and understanding to support activities beneficial to themselves and others in everyday contexts. They recall, select and communicate a thorough knowledge and understanding of the general competences needed to support lifelong learning and personal well-being in keeping with the Level 3 general level descriptor in the QCF.

6.2 They apply knowledge, understanding and skills to a variety of situations including those that are unfamiliar and require some analysis and synthesis of new ideas, selecting and using knowledge and information efficiently to solve problems and produce effective support for their own learning as well as the needs of others. They relate these to comparable activities in the world of work. They manipulate and process data efficiently and effectively based on objective criteria. They interpret information and transfer knowledge and understanding from familiar to unfamiliar contexts and produce new insights in practical circumstances. They work creatively exploring and developing new ideas. They adopt systematic approaches to safety, promoting secure and responsible practices and they can make confident decisions to obtain good value for money.

6.3 They use scientific and mathematical methods to analyse problems such as control of variables, setting up and solving equations. They set hypotheses in relevant contexts and critically analyse and evaluate the knowledge they gain. They use sound knowledge to review their own work and that of others including that of industry professionals making appropriate, supportive and constructive criticisms and recommendations for improvements. They communicate effectively, demonstrating a clear sense of purpose and audience.

A **grade E** candidate will exhibit most of the following characteristics

6.4 Candidates demonstrate the ability to select and use relevant knowledge, ideas, skills and procedures to complete defined tasks and address realistic but straightforward software engineering problems. They take responsibility for completing tasks and procedures initiating and completing tasks and procedures as well as exercising autonomy and judgment within limited parameters.

6.5 They use factual, procedural and theoretical understanding to support their work and use appropriate investigation to inform actions. They address problems that, while well defined, may be complex and non-routine. They appreciate different perspectives on subjective issues.

6.6 They work safely and securely, identifying key risks, taking reasonable actions to avoid them. They can work in teams, where relevant taking responsibility for supervising or guiding others, collaborating in reviewing their work evaluating the way they and others use their construction knowledge and skills and taking positive actions to improve. They use standard English and IT to support clear and efficient communication, demonstrating consideration of purpose and audience and use straightforward mathematical techniques in quantitative work.

Annexe A - Example examinations at Level 3 (70 marks)

The following principles will apply to the design and structure of each exam.

Questions will vary in the general area of the required learning outcomes specified in the relevant units and cover the assessment criteria in the approximate proportions presented in this document. Questions will reflect a balance of the content listed and explained in the guidance in keeping with Level 3 as defined by the QCF global level descriptors and the grade descriptions above.

Model for the Diploma examination.

Part 1 multiple choice questions

Each multiple choice question is worth 0.5 of a mark.

Questions

1. What is does the abbreviation BIT mean?

    a) Big Information Technology.
    b) Binary Digit.
    c) Bytes in Tune.
    d) Business Information Training.

2. An abstraction is

    a) a sample taken from a set of random measurements.
    b) an imaginary drawing.
    c) a simplification representing a real object.
    d) a false result produced by an algorith.

3. In an algorithm, a number is halved at each iteration. If the starting number is 1024 how many iterations will take place before reaching 1?

a) 7
b) 8
c) 9
d) 10

4. The binary number 11111111 in hexadecimal is

a) AA
b) CC
c) EE
d) FF

5. The decimal number 21 is the binary number 010101. It is represented by 6 cups with 3 empty cups and three containing spoons. To multiply the number 21 by 2 and return the result in binary I could

a) shift each spoon one cup to the left.
b) shift each spoon one cup to the right.
c) add a spoon to the furthest left cup.
d) add spoons to all empty cups.

6. Release early release often refers to

a) A strategy to keep employment costs down.
b) A strategy for developing and maintaining software.
c) A mechanism to speed up memory caches.
d) A mechanism for reducing power consumption of CPUs.

7. A computer display has 1366 pixels horizontally and 768 pixels vertically. Each pixel has 24 bits for colour, how many bytes of memory are needed to map the screen?

a) 1324564

b) 2453671
c) 3147264
d) 4256739

8. Classes and inheritance are associated with

a) object oriented programing.
b) procedural programming.
c) command scripts.
d) macros.

9. Software development requires

a) Testing only after the alpha release stage is reached.
b) Testing from the beta release stage onwards.
c) Testing at the release candidate stage.
d) Testing at all points involving end users.

10. A developer wants to license his code to be freely used and redeveloped but he does not want it to be relicensed as proprietary software. A suitable license would be

a) The GPL V3.
b) The BSD license.
c) The Apache Software Foundation license V2.0.
d) The Public Domain License.

11. Copyleft ensures

a) that the copyright logo is displayed in the right place.
b) that the same rights provided to the user are passed on to others.
c) the rights to copy a work are preserved after the copyright has expired.
d) balance between copyright and other aspects of intellectual property.

12. In a large project with many developers contributing software patches and components, it is very advisable if not essential to use

a) only one programming language.
b) only one operating system.
c) a hierarchical management team.
d) distributed revision control systems.

13. European law with regard to software patents

a) is identical to US Law.
b) means general purpose programs can not be patented.
c) means all software can be protected with a patent.
d) makes no explicit mention of software.

14. Sourceforge is

a) a method for compiling source code into executable code.
b) a company that manages proprietary software for companies.
c) a place from where open source projects are downloaded
d) a social networking site for software entrepreneurs.

15. What is an open source community?

a) A group of people who are happy to share their personal details on the internet.
b) A group of people who are primarily interested in the commercialisation of software.
c) A group of people who support software that is developed openly.
d) A group of people who only use object oriented programming.

16. An example of the software as a service economic model is

    a) Buying an operating system license.
    b) Downloading a software application freely and legally from the internet.
    c) Paying a subscription to access useful tools on a web site.
    d) Using web services to enable communication between applications.

17. A break point in a program's source code is

    a) a weakness where failure is likely.
    b) an exit point from a recursive algorithm.
    c) a branch to a subroutine.
    d) a deliberate pause to help in debugging.

18. Open computing systems are

    a) insecure because anyone can access them without a password.
    b) always based upon open source software.
    c) not dependent on protocols owned by individual companies.
    d) used to get better airflow across energy intensive components.

19. In the expression 3.0×10^8, the number 8 is the

    a) exponent
    b) sign
    c) mantissa
    d) subscript

20. To document a software application it is essential to

    a) include clear comments throughout the source code.

b) write a manual for the end user.
c) write a description of the methods adopted in planning.
d) write an evaluation of the project with strengths and weaknesses.

21. Agile software development involves

a) a strict management structure with clear team roles.
b) self-organising, cross-functional teams.
c) a young development team.
d) large commercial organisations with sufficient funds.

22. IPv6 was devised because

a) The original 32 bit IPv4 did not provide enough IP addresses.
b) IPv4 is subject to a patent dispute.
c) Pv4 is not compatible with modern multimedia protocols
d) IPv6 is a natural upgrade to speed up the throughput of data on the internet.

23. In a software development community those that contribute the most code have greatest say in directing the project. This is known as

a) autocracy
b) meritocracy
c) democracy
d) bureaucracy

24. In a digital age intellectual property rights are more difficult to enforce because

a) there is no agreement between countries on the details of law.
b) there are not enough policeman with IT skills.

c) there is a lot of disagreement about the basis for intellectual property.
d) it is easy to copy work without any loss of quality.

25. A USB transfers data using a set of rules encoded in data packets. These rules and the way they are organised are called

a) protocols.
b) serial ports.
c) interrupts.
d) tokens.

26. An effective error message in finalised code should give the highest priority to

a) enabling the user to understand what went wrong.
b) providing technical information to the programmer.
c) not getting in the way of the application as it runs.
d) being compact and not taking up too much memory.

27. A good user interface design will

a) put visual aesthetics ahead of consistency.
b) give all actions equal priority.
c) not allow user errors that could be prevented.
d) make the help link small and not too prominent.

28. Opening Port 80 to HTTP traffic is an operation associated with

a) Firewalls
b) Password security
c) Backup strategy
d) RAID disc security

29. I need to speed up a library routine used by many programs. To do this I could

a) Remove all the comments in the source code.
b) Replace a loop requiring 100 CPU cycles with one requiring 70.
c) Make all the variables in the source code one character in length.
d) Compress the library using a lossy compression application.

30. Care needs to be taken when formatting a disc because

a) Using an incompatible format can damage the disc surface.
b) Formatting will effectively erase all existing data on the disc.
c) Formatting could introduce a virus or other malware.
d) Formatting could cause a copyright violation.

31. Which of these series would produce a convergent iteration when each term is added to the next.

a) 1, 2, 3, 4, ...
b) 1, 3, 5, 7, ...
c) 2, 4, 8, 16,....
d) ½, ¼, ⅛, 1/16.....

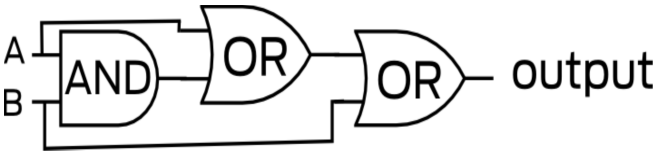
32. A digital measuring instrument is to be made by sampling an analogue signal across its full amplitude and storing the data in one byte. The maximum number of divisions on the scale of the instrument is

a) 100
b) 128
c) 255
d) 511

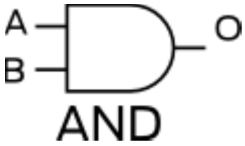
33. The number 42 converted to packed binary coded decimal is

a) 01000010
b) 101010
c) 00040002
d) &42

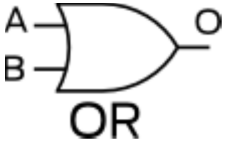
34. Which of the circuits below is the equivalent of the circuit above.



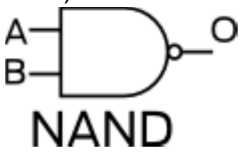
a)



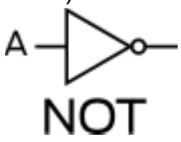
b)



c)



d)



35. I want to write a program to draw the letters of the alphabet on a screen. An economical strategy would be

a) to draw each upper and lower case letter individually using a set of coordinates for each letter.
b) define vertical, horizontal and curved line elements and fit them together as needed to make each letter.
c) draw all the lower case letters first and then the upper case letters in a range of sizes.
d) draw all the letters with curves in them first, then those with straight lines, then those with a combination.

36. The following pseudocode program

```
FOR I = 20 to 30
   SET print_number to true;
   IF I is divisible by 3 THEN
      PRINT "F";
      SET print_number to false;
   IF I is divisible by 5 then
      PRINT "B";
      set print_number to false;
   IF print_number, PRINT I;
NEXT I
```

would be expected to produce

a) 2021222324252627282930
b) 20F22B24F26B27F30
c) BF2223FB26F2829FB
d) FBFBFBFBFBFBFBFB

37. Wireless network connections

a) are always slower than cable connections.
b) are insecure as it’s easy to bipass security.

c) are only effective up to about 100 m.
d) transfer data more slowly the further the distance the user is from the access point.

38. There are 128 printable characters that can be used in a password and 52 upper and lower case letters of the alphabet. What is the difference between the number of combinations for a 2 character password using all 128 possible characters and one that uses only letters of the alphabet.

a) 180 X 76
b) 128 X 52
c) 128 - 52
d) 16832

39. Each day my computer system needs to check a set of files and remove all that ar more than 10 days old. In order to automate this process I would use

a) DHCP.
b) a cron job.
c) Windows update.
d) ping.

40. Here is a short piece of code operating on numbers in two CPU registers R1 and R2.

```
L:
CMP  R1, R2 ; set a condition "NE" if numbers in R1 and R2 are not =
                ; set a condition “GT” if numb in R1 > numb in R2
                ; set a condition "LT" if numb in R1 < numb in R2
 SUBGT  R1, R1, R2   ; if "GT" take num in R2 from num in R1
 SUBLT  R2, R2, R1   ; if "LT" take num in R1 from num in R2
 BNE  L              ; Branch to the label L if not equal
```

The function of this code is

a) to find two numbers that are the same.
b) to find out if either of two number is a prime number.
c) to find the lowest number of two numbers.
d) to find the highest number that will divide into two numbers with no remainder.

40 M/C question (20 marks)

Part 2 - Short answer questions

Q1. Look at the following pattern of characters

AAAABBBBBCDDDDDDEEEEEEEFFF

There are 25 letters and each character takes up 1 byte of storage, 25 bytes altogether. Explain how the data could be stored in less than half the space and still keep all the information. (2 marks).

Q2. Explain two important contributions that end users can make in software development. (2 marks).

Q3. For a computer programming language that you have used, explain 2 reasons why you would or would not use this language in a future project (2 marks)

Q4. Give two reasons why a commercial company might contribute to a free and open source project. (2 marks)

Q5. A friend has a computer that won't boot and has important unbacked up information on its main drive. Suggest a strategy to retrieve the data that does not involve buying a software license or giving the computer to a third party to fix. (2 marks)

Part 3 - Long answer questions (40 marks)

Q1. What are the characteristics of computational thinking? In your answer, refer to organising data, abstraction, algorithm, problem solving and simulations. (10 marks)

Q2. For a software project you know well explain the strategies used to maintain quality. (10 marks)

Q3. Describe 3 commercial models for software development and maintenance. Illustrate your answer with real world examples. (10 marks)

Q4. A small business on a very tight budget has 5 employees and needs basic document editing, simple spreadsheets, sharing information and internet access for conferences with partners in other countries. Write a proposal specifying a technical solution that will meet their needs. Justify each of your recommendations. (10 marks)

Model for the Certificate Examination. (70 Marks)

In the multiple choice questions choose the best response to the statement or question posed.

1. A feature based software release strategy is

    a) less likely to engage end users in the development process.
    b) is more likely to result in more frequent software updates.
    c) is less likely to be associated with large commercial projects.
    d) is more likely to be used for web development.

2. A well-designed computer help system

    a) requires anti-virus software.
    b) should always be written using HTML.
    c) should be inconspicuously linked to the desktop.
    d) should provide information in the current context.

3. A user might complain about a Captcha because

    a) they are colour blind and can't make out the letters and numbers.
    b) they don't want their information gathered without their permission.
    c) they need more memory for the application.
    d) they do not remember their user name and password.

4. When a 404 error occurs it means a page does not exist from the link the user has accessed. Which of these would be the most appropriate error message for an end-user in such an event?

    a) Error - 404

b) Information you are trying to access does not seem to exist from that link.
c) Silly mistake, no URL recognised
d) An error occurred and it is probably because you clicked on a bad link so don't do it again!

5. A software engineer wants to reduce the risk of an error in his code. Which one of these is the best strategy.

a) Use the Python Programming Language.
b) Use the Java Programming language.
c) Get a partner to evaluate the code at the end.
d) Get a partner to develop the code alongside him.

6. Before being released, all code should be

a) tested by an external expert.
b) tested by the author's most trusted peer.
c) tested by an end user.
d) tested by as many and as diverse a range of people as possible.

7. A procedural programming language is one that

a) is based on defining objects with properties that can be inherited.
b) is based on a series of steps carried out sequentially.
c) is based on assembly codes.
d) is visual and minimises the need for writing source code.

8. A programmer suggests she can document her work entirely by using comments in the source code.

a) this is not possible because it would slow the program down too much.
b) this is not possible except in the smallest projects.

c) this is possible and most useful if the code is open source.
d) this is possible but not something that is considered good practice.

9. A release candidate will go on to general release if

a) No serious bugs are found by the end of a certain time period.
b) If it passes through alpha and beta stages after being a release candidate.
c) The source code is relicensed as open source.
d) The code is certified as not being malware.

10. Breaking a large software project down into many smaller components

a) Is generally not possible because all the components have to be compatible with one another.
b) Is possible but likely to produce a lot of errors and therefore a lot of debugging.
c) Is only advisable when the number of developers needed gets to double figures.
d) Is always advisable because the software will be easier to maintain.

11. For two software communities that you have studied explain 4 key similarities and/or differences in the processes they use for development. Consider technical, community and licensing issues.

(4 marks)

12. The Apache Software Foundation only accepts “Committers” to projects as individuals. It does not accept companies in this role. Committers are people authorised to commit contributions to software projects. Explain the reasons why ASF might have adopted this policy.

(2 marks)

13. LibreOffice is a large open source development project that uses a license that requires all subsequent code to be released under the same license (GPL). The Apache OpenOffice project releases code under the Apache license that allows code to be relicensed as open or closed source software. Explain why developers in the LibreOffice community can take code from the Apache OpenOffice project but Apache Committers can not take code from the LibreOffice project.

(2 marks)

14. What are the essential differences between a work licensed under Creative Commons and a work in the Public Domain?

(2 marks)

15. Explain the difference between quality control and quality assurance in relation to software development? How are these terms best related to the methods of Agile software development?

(4 marks)

16. Two software models are monolithic, where all the code for a complex application is in one large program, and modular, where code is divided into modules that can communicate with each other. Discuss possible advantages and disadvantages of these two systems.

(4 marks)

17. A software engineer wants to make a living from developing open source code. Suggest three methods that could make this possible.

(3 marks)

18. Explain why a software program placed for free download from the internet is not necessarily Free software as defined by the Free Software Foundation.

(3 marks)

19. Digital Restrictions Management imposes technological restrictions that control what users can do with digital media, typically preventing

copying and redistribution. Explain whether you think this is an effective way for copyright holders to ensure they get paid fairly for their work.

(6 marks)

20. “In 1991, software-related patents were less than a quarter of all patents issued by the U.S. Patent and Trademark Office. In 2011, software patents accounted for the majority of all patents issued”

Discuss the case for and against software patents in Europe.

(10 marks)

21. Describe the software as a service model (SaaS), it's key benefits and how it might be used to generate revenue to pay for the service.

(10 marks)

22. Describe a small scale software project or projects that you completed from the perspective of quality assurance.

(10 marks)

Annexe B - Level 3 Units

Level 3 Open Systems Computing

Unit 1: Computational Thinking
10 credits (75 GLH) H/505/5835

1. understand the computational problem solving process	**2. be able to apply number systems and logic to computing problems**	**3. analyse problems to create computational solutions.**
1.1 organise data in terms of logical patterns.	2.1 explain the relationship between binary and hexadecimal numbers.	3.1 identify practical problems suitable for a computational solution.
1.2 demonstrate how abstractions represent complex data structures and instructions.	2.2 explain how digital computers can work with a full range of real numbers.	3.2 analyse complex problems into simpler related components.
1.3 iteratively refine solutions to improve efficiency and effectiveness.	2.3 explain the difference between packed and unpacked binary coded decimal.	3.3 explain computational solutions in terms of sequential automated steps.
1.4 use multiple algorithms to solve complex problems.	2.4 use mathematical functions in practical algorithms.	3.4 find ways of making computational solutions more efficient.
1.5 consult with relevant industry professionals and academics to improve solutions.	2.5 analyse expressions in boolean logic to simplify them.	3.5 work collaboratively and persistently to achieve a good computational solution.

This unit provides the academic knowledge and understanding to under-pin practical digital technology projects.

Level 3 Open Systems Computing.

Unit 2: Principles of software engineering
10 credits (75 GLH) F/505/5843

1. understand the role of the target audience	2. understand strategies for maintaining quality	3. adopt suitable methods to match circumstances
1.1 describe the rationale for release early, release often.	2.1 identify design techniques to reduce risk.	3.1 compare procedural and object oriented programming.
1.2 explain principles of user interface design.	2.2 explain a sound testing strategy.	3.2 compare formal and agile methods.
1.3 receive user feedback and act positively.	2.3 establish clear communication channels with critical reviewers.	3.3. specify a documentation strategy.
1.4 compare the user role in a range of software development models.	2.4. explain and demonstrate the importance of courage and persistence in solving problems.	3.4. describe an open source community project and its methods.
1.5 describe methods for providing feedback to users from errors in the code.	2.5 demonstrate quality strategies through small scale projects.	3.5. explain the different demands of large scale and small scale projects.

This unit is about key considerations when tackling practical digital technology projects. Evidence can come from planning, implementing and evaluating such projects.

Level 3 Open Systems Computing.

Unit 3: Delivering a software project
10 credits (60 GLH) A/505/5842

1. plan a suitable project.	2. carry out a significant practical software project.	3. communicate project outcomes to others.
1.1 identify an area of interest and scope the project.	2.1 produce substantial code that works effectively.	3.1 provide regular updates on progress to a mentor.
1.2 agree and adopt the software development method.	2.2 produce source code that has effective embedded documentation.	3.2 analyse issues arising and establish priorities for resolution.
1.3 present the proposal to critical experts.	2.3 use logical techniques to debug code.	3.3 gather opinions through peer review.
1.4 make modifications as a result of feedback.	2.4 show courage and determination to overcome problems.	3.4 use IT tools to enhance communication.
1.5 meet deadlines.	2.5 test code regularly involving third parties.	3.5 make a final presentation to a critical audience.

This unit is about practical realisation of a software project. The ideal method is to contribute to an existing open source project as this will provide many additional relevant learning opportunities. This is not mandatory, but all projects should be planned to provide outcomes useful to other people rather than as academic programming exercises.

Level 3 Open Systems Computing

Unit 4: Open Systems and Community Development 10 credits (75 GLH) T/505/5841

1. understand the process of community development.	**2. understand licensing and intellectual property.**	**3. understand commercial models for software development.**
1.1 compare and contrast the processes of software development communities.	2.1 explain the relationship between copyright and licensing.	3.1 describe the dual licensing model for software development.
1.2 explain the principles of the Open Source Way.	2.2 describe and explain the freedoms associated with free and open source software.	3.2 describe the software as a service model.
1.3 explain the relationships between commercial and volunteer interests in a software development community.	2.3 analyse the effects of digital technologies on the enforcement of intellectual property rights.	3.3 describe the perpetual license model for software development.
1.4 describe Sourceforge and its role in community development.	2.4 describe the advantages and disadvantages of software patents.	3.4 describe the fremium model for software development.
1.5 explain the importance of distributed revision control systems in community software development.	2.5 explain the terms trademark, copyleft, creative commons, and public domain.	3.5 describe an advertising model to support software development.

This unit is about free and open source software communities, their culture and ways of working.

Level 3 Open Systems Computing

Unit 5: Computer Systems Management - 10 credits (75 GLH) J/505/5844

1. set up systems.	**2. support system storage and security.**	**3. maintain systems.**	**4. understand key internet systems.**
1.1 install and set up an operating system.	2.1 format and partition storage devices.	3.1 install software updates and dependencies.	4.1 describe the terms HTML, W3C and HTTP.
1.2 set and customise boot sequence and options.	2.2 devise and implement a backup strategy.	3.2 set up cron jobs to automate regular procedures	4.2 explain the function of a web server.
1.3 customise the display to personal preference.	2.3 set up and understand how to customise a firewall for network connection.	3.3 set up a secure virtual connection to manage a system from a remote location.	4.3 explain the importance of TCP/IP.
1.4 set up network connections.	2.4. write a risk assessment for system security including passwords and malware.	3.4 install and remove new applications.	4.4 explain the effects of proprietary standards and lock-in.
1.5 solve problems in systems setup and configuration.	2.5 describe a range of storage devices and their strengths and weaknesses.	3.5 provide effective support for system users.	4.5 explain the role of an internet service provider.

This unit is about managing computer systems in an internet open systems context.

Annexe C - 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

- 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 e-portfolios 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

Unit 1: Computational Thinking

1. The candidate will understand the computational problem solving process.

I can:

1.1 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 eg 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. eg 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 demonstrate how abstractions represent complex data structures and instructions

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

Evidence: portfolio 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 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 eg 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 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 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. eg 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 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

I can:

2.1 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 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 eg 3.8×10^7. 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 at the link below.
http://www.tfinley.net/notes/cps104/floating.html

2.3 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. eg 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 sonme bit capacity is redundant because we are only using 0 - 9 when there is enough capacity for 0 - 15.

2.4 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 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:
http://www.allaboutcircuits.com/vol_4/chpt_7/1.html

This link 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.

3. The candidate will analyse problems to create computational solutions.

I can:

3.1 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 eg http://www.huffingtonpost.co.uk/2012/10/11/physicists-may-have-evide_n_1957777.html
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 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 eg 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 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 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?

Further 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. eg research on sort or search algorithms, efficient data storage, using integer arithmetic, using more efficient libraries.

3.5 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.

Further 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.

Unit 2: Principles of Software Engineering

1. The candidate will understand the role of the target audience.

I can:

1.1 describe the rationale for release early, release often.

candidates should be familiar with the work of Eric Raymond in the Cathedral and the Bazaar.

Evidence: portfolios.

Additional information and guidance: Candidates should read about the history and background that led to the release early and release often philosophy. They should be able to refer to specific examples of software produced on this and alternative models and debate the advantages and disadvantages of different approaches.

1.2 explain principles of user interface design.

Candidates should provide a study of user interface design explaining key things learnt for their own designs.

Evidence: portfolios.

Additional information and guidance: Candidates should provide evidence of significant research including the use of web search but also studies of popular sites analysing them for strengths and weaknesses. Which features would they use and why?

1.3 receive user feedback and act positively.

Candidates should actively seek feedback on their work from third parties and act positively on the information provided.

Evidence: portfolios.

Additional information and guidance: Candidates should regularly seek feedback from peers and other third parties and be prepared to

give the same. Feedback should be constructive and objective and lead to improvement. Evidence should span all their projects from small exercises to their large scale project.

1.4 compare the user role in a range of software development models.

Candidates should carry out a study of software development models from the point of view of user involvement.

Evidence: Portfolios.

Additional information and guidance: In some software development communities the users are often project members and the distinction becomes blurred. In some cases it is easy for users to contribute feedback and even file bug reports and track them. In others it might not be possible to do this at all. Does the maxim “many eyes make bugs shallow” stand up to evidence. This could be a good subject for a team approach to gather evidence, pool resources and discuss final conclusions. This is to be encouraged as long as assessors can be clear that individuals meet the criterion.

1.5 describe methods for providing feedback to users from errors in the code.

Candidates should do a critical survey of the error messages provided to users in commonly used software and use the outcomes to inform their own coding projects.

Evidence: Portfolios.

Additional information and guidance: There is still a lot of poor practice in commercial software applications when it comes to error messages. A good starting point for research is http://www.developsense.com/essays/AReviewOfErrorMessages.html With the advent of the web, more applications are using the web for help files. This could also be extended to error messages from applications. If there is a knowledge base for a software application in the web, it should be possible to automatically search it when an error occurs and add to it useful information about the circumstances that generated the error. If it is an error due to a bug in the code, it can then be automatically routed to the bug tracking system and linked to a user

log file so that programmers could see what caused the bug and eliminate it. If it was a user error it can inform design to make the user error impossible. The overall goal should be to eliminate the need for error messages but where they occur they should be as useful as possible to the user. Linking errors to an online content management system makes it easy to edit the messages, provide language translations etc.

2. The candidate will understand strategies for maintaining quality.

I can:

2.1 identify design techniques to reduce risk.

Candidates should research and identify software design methods that will reduce risk to quality in their current or future projects.

Evidence: from portfolios.

Additional information and guidance: Candidates should use their research to help them decide on their approach to quality assurance in their software development. Their evidence should show that they have considered a wide range of views and opinions.

2.2 explain a sound testing strategy.

Testing strategies for software development should be based on accepted industry practice.

Evidence: documentation in portfolios.

Additional information and guidance: Candidates should show that they have researched testing methods and considered a range of views. They should come to conclusions about how they will implement testing and they should be prepared to modify these in the light of practical application.

2.3 establish clear communication channels with critical reviewers.

Candidates should develop practical communication channels with reviewers of their work using collaborative technologies to aid the process.

Evidence: assessor observations and portfolios.

Additional information and guidance: Candidates should set up, organise and use appropriate collaborative technologies to make review and feedback straightforward and regular.

2.4 explain and demonstrate the importance of courage and persistence in solving problems.

Candidates should understand and demonstrate these attitudes in the pursuit of quality.

Evidence: Assessor observations, documentation in portfolios.

Additional information and guidance: Candidates should keep a diary or Blog to document their use of time in solving problems related to quality in their projects.

2.5 demonstrate quality strategies through small scale projects.

Candidates should trial the quality assurance strategies they have researched in preliminary small scale projects in preparation for their main project.

Evidence: from documented small scale projects.

Additional information and guidance: The small scale projects should be used as a practical vehicle for trying out research findings in the more theoretical aspects of the work and as a preparation for their large scale project. Quality will be enhanced in a large scale project by understanding the issues through practical experience in small scale projects.

3. The candidate will Adopt suitable methods to match circumstances.

I can:

3.1 compare procedural and object oriented programming.

The candidate will be able to make clear judgements about programming methods based on evidence.

Evidence: from assessor judgements and content of portfolios.

Additional information and guidance: Candidates should carry out small scale research into programming methods (formally referencing sources) and gain some experience of procedural and object oriented programming in small scale projects so they have some practical basis for their judgements. Initially they will need considerable guidance but the aim is to get them more experience and therefore more confidence to make judgements themselves.

3.2 compare formal and agile methods.

Candidates should have enough knowledge of wider opinions on methods to make objective comparisons analysing strengths and weaknesses.

Evidence: Assessor observations and contents of portfolios.

Additional information and guidance: Candidates should carry out small scale research into programming methods (formally referencing sources) and gain some experience of agile and formal methods in small scale projects so they have some practical basis for their judgements. Initially they will need considerable guidance but the aim is to get them more experience and therefore more confidence to make judgements themselves.

3.3 specify a documentation strategy.

The candidate will use research evidence to specify their preferred method of documenting their work and justify it with evidence.

Evidence: Assessor observations and portfolios.

Additional information and guidance: Candidates should carry out small scale research into documentation methods (formally referencing sources) and use documentation of small scale projects so they have some practical basis for their judgements. They should see differences in the documentation philosophies in agile and formal approaches. There are also issues related to openness of the source code, user interface design and how interactive help works. the power of searching web based documentation has to be weighed against inconvenience it the internet is not readily available or expensive to access.

3.4 describe an open source community project and its methods.

Candidates should have studied an open source community project in some detail.

Evidence: Assessor observation and contents of portfolios.

Additional information and guidance: Candidates should choose an open source project to study. Sourceforge is a good starting point. As a minimum they should understand the governance of the project eg Apache OpenOffice has a project management committee within the Apache Software Foundation and uses Apache software licensing for the release products whereas LibreOffice is managed by The Document Foundation and releases LibreOffice under the LGPLv3+ and MPL. This means code can be taken from Apache OpenOffice and used in LibreOffice but code can not be taken from LibreOffice to be used in OpenOffice. This could and should be an opportunity to link to criteria on intellectual property. If members of a group study different Open Source projects it would be a useful exercise to present each project to the whole group to widen understanding. If candidates use this learning to make a contribution to an Open Source project using their large scale project, so much the better.

3.5 explain the different demands of large scale and small scale projects.

Candidates should provide a detailed explanation of the range of demands of small and large scale projects based on analysis of experienced views and their own empirical experience.

Evidence: portfolios.

Additional information and guidance: Candidates should carry out small scale research to determine views on these issues. Evidence might include what goes wrong on large scale government IT projects? Why are they so expensive? At what point does a small scale project become large scale? How does the number of developers involved affect a project's efficiency? Can small scale projects scale to become large scale? - eg the Linux Kernel was started by one developer (Linus Torvalds) in 1991 and is now estimated to be used in nearly half of all consumer devices with around 200 active developers and 15 million lines of code. The main outcome from this criterion should be a rational appreciation of considerations for project planning.

Unit 3: Delivering a software project

1. The candidate will plan a suitable project.

I can:

1.1 identify an area of interest and scope the project.

Candidates should collate a range of information gathered in other parts of the course and use it and any other background to inform their decisions.

Evidence: Assessor observations and portfolios.

Additional information and guidance: It is probably advisable to have an initial planning and preparation phase in the first half of the course and start work on the main project around the mid point or a little before. This will give time for evidence gathering and practice of techniques in the context of small scale projects to help inform planning. A successful project is very likely if the preparation is sound.

1.2 agree and adopt the software development method.

Candidates should use background learning to decide on their approach and agree it with their assessor or mentor.

Evidence: Assessor observations, portfolios.

Additional information and guidance: The assessor or mentor should act as guides to ensure that the candidate does not set off on an unrealistic route that could jeopardise the entire project. Candidates should have realistic input and once agreed need to show appropriate commitment to the method.

1.3 present the proposal to critical experts.

Candidates should prepare a presentation of their agreed proposal and methods and present it to a critical independent person(s).

Evidence: documented presentation in portfolios.

Additional information and guidance: Once the project outline and method is agreed, candidates should use it as a basis for a presentation to a critical third party that has knowledge and experience in the field. The presentation does not have to be physical, it could be by web cast or web video conference eg Google Hangouts or Skype.

1.4 make modifications as a result of feedback.

Candidates should be able to use the feedback from the expert source to make improvements to their proposal.

Evidence: from documentation in portfolios.

Additional information and guidance: Improvements might be in the software tools, in the scope of the project or implementation methods. In some cases there might be little to do as there is no point it making modifications for the sake of it. Whatever the case there should be clear evidence that consideration was given to the findings of the consultation.

1.5 meet deadlines.

Candidates should be able to estimate milestones in the project and once committed to them meet the deadlines.

Evidence: from assessor observations, project portfolio evidence of timings.

Additional information and guidance: Candidates should be familiar with SMART targets and the need to consider deadlines and ensure they are met. If in the assessor's judgement a deadline was missed through no fault of the student it can be ignored. If on the other hand, the student is simply being lazy or disorganised a warning should be given and there should be clear evidence of a real effort to get things back on track. The final deadline is project completion and if it is not achieved at least in some useful form their should be a very good reason.

2. The candidate will carry out a significant practical software project.

I can:

2.1 produce substantial code that works effectively.

The aim should be to produce high quality well documented code that is consistent with 100 hours of work including their planning and consultation.

Evidence: from assessor observations, software application and its documentation in portfolios.

Additional information and guidance: There is no guide such as lines of code produced because we do not want to encourage producing software that is inefficient just to meet a lines quota. Assessors and candidates should be constantly reviewing productivity and the time spent producing. Peer review could be helpful here in determining what has been achieved. For particularly good projects we will provide additional recognition outside the scope of the qualification and candidates should realise that being able to demonstrate a high quality useful software application might have as much or more weight with an employer or university admissions tutor as the qualification certificate. Assessors should ensure that the actual coding is the candidate's own work. Mentors and assessors can provide guidance and advice but they should NOT provide code. If candidates use source code eg from an Open Source application they should reference it and acknowledge the source in the same way as with book references. They should make it clear which is their own work and which was other peoples'. In this way more than one student could work on the same project.

2.2 produce source code that has effective embedded documentation.

Candidates should document their source code with embedded comments at least well enough to for a technically knowledgeable person to follow.

Evidence: from documented source code in portfolios.

Additional information and guidance: There is a school of thought that embedded documentation is all that should be needed. Whether or not this is the case, the documentation should be good enough on its own to see what the program is doing.

2.3 use logical techniques to debug code.

Candidates should demonstrate techniques such as control of variables and program breaks to isolate bugs.

Evidence: Documentation of methods in portfolios.

Additional information and guidance: Assessors should provide guidance in strategies for isolating bugs and providing useful error messages.

2.4 show courage and determination to overcome problems.

The candidate will provide evidence of courage and perseverance in overcoming adversity.

Evidence: from assessor observations and documents in portfolio.

Additional information and guidance: Candidates should be able to demonstrate that they have made real efforts to overcome problems that they find difficult. This means that able candidates might well need to go beyond basic level 3 work to demonstrate this characteristic. It

would be a good idea for candidates to keep diaries in their portfolios so that they can record evidence of the need for courage and determination as it arises.

2.5 test code regularly with third parties.

The candidate will make testing a routine part of development.

Evidence: from assessor observations and documents in portfolio.

Additional information and guidance: Candidates should involve peers, mentors and assessors in the testing process and respond positively to any feedback. Testing might help eliminate bugs but it might also help improve functionality through user feedback.

3. The candidate will communicate project outcomes to others.

I can:

3.1 provide regular updates on progress to a mentor.

Candidates should meet with a mentor at several points in the development process to report on progress.

Evidence: from assessor observations and documents in portfolio.

Additional information and guidance: Candidates keeping diaries should simply record update issues and mentor comments. This could be combined with the testing strategy. The mentor could be the assessor but does not have to be.

3.2 analyse issues arising and establish priorities for resolution.

As for the criterion.

Evidence: from assessor observations and documents in portfolio.

Additional information and guidance: Candidates keeping diaries should simply record a brief analysis of any issues arising from feedback from third parties and record actions as a result.

3.3 gather opinions through peer review.

Candidates should use regular peer review and provide the same for others

Evidence: from assessor observations and documents in portfolio.

Additional information and guidance: Candidates should contribute to an objective and rational peer review process. They should receive criticism graciously and provide it objectively and sensitively.

3.4 use IT tools to enhance communication.

Candidates should use appropriate IT tools to support communication with peers mentors, assessor and other third parties.

Evidence: from assessor observations and documents in portfolio.

Additional information and guidance: Candidates should use a range of collaborative tools to support communication. It is not necessary for all communications to be face to face meetings. Aspects can be by e-mail, videoconference, sharing resources etc. Encourage experimentation to go beyond stereotypical Powerpoint slides.

3.5 make a final presentation to a critical audience.

Candidates should make a final presentation of their completed project.

Evidence: from assessor observations and documents in portfolio.

Additional information and guidance: Candidates should have the opportunity to make a professional presentation to a suitable audience that is knowledgeable enough to provide fair criticism and feedback.

Unit 4: Open Systems and Community Development

1. The candidate will understand the process of community development.

I can:

1.1 compare and contrast the processes of software development communities.

Candidates can compare the working methods of two distinct communities and explain how they are similar and how they are different.

Evidence: Written reports in portfolios.

Additional information and guidance: Communities have similarities and differences in the ways they operate. They have different release procedures for code and they operate with different licensing policies. Candidates should research two such communities and produce a final report of about 1000 words summarising their findings. The report should start with an abstract containing the headline findings and then provide a general description of each project and how it is organised, any key personalities and any key statistics - size, number of downloads a month etc. The descriptions should be used as a source for the comparisons and any judgements and conclusions. Candidates should realise that structure helps in documentation as well as code.

1.2 explain the principles of the Open Source Way.

Candidates be familiar with the 5 principles as defined by OpenSource.com

Evidence: from portfolios of evidence.

Additional information and guidance: The principles are

- Community
- Rapid prototyping
- Open exchange
- Participation

- Meritocracy

CROPM might help remember them. While these principles have arisen in Open Source software development communities they also have general application in other fields. Candidates should be encouraged to consider which aspects transfer well to other fields and which do not. Software is an unusual product in that it has a development cost but virtually no cost of manufacture and distribution. Other mass produced products have most of their cost in manufacture and distribution and this gets more expensive the large the product volume. With software distribution is often referred to as viral because it can be like the spread of a virus as replication and spread has much lower barriers when compared to hardware.

1.3 explain the relationships between commercial and volunteer interests in a software development community.

Candidates should explain how both commercial and voluntary interests can be supported by software development communities.

Evidence: from documentation in portfolios.

Additional information and guidance: They should use research to find projects that involve commercial and volunteer interests and consider how referring to software as “commercial” or “Open Source” is misleading. Open Source can be commercialised, just not by selling licenses. Open Source code can be taken and then developed further as closed source eg as with ios, open source code can be combined with closed source elements as in the Android OS. Some licenses allow code to change from Open Source to closed eg BSD and Apache while the GPL doesn’t. Commercial companies such as Canonical work with GPL licensed software using a business model that is not dependent on selling licenses. It is a complex field with many misunderstandings. Those used to closed licenses linked to commercial interests have traditionally had a lot of trouble accepting and understanding Open Source as a business model. These business models tend to be much more about cooperation than competition but competition is not entirely absent, It is more a shift in emphasis. Many large software communities have a mixture of volunteer and commercially sponsored developers. There are many different relationships and motivations.

1.4 describe Sourceforge and its role in community development.

Candidates should be familiar with Sourceforge as the main repository for open source code.

Evidence: from documentation in portfolios.

Additional information and guidance: Candidates should research Sourceforge, its projects and purpose and write a concise 500 word description of it in their portfolios. Similar lengths podcasts, videos or other media presentations are to be encouraged.

1.5 explain the importance of distributed revision control systems in community software development

Candidates should be able to explain version control as important to the management of any sizeable project.

Evidence: Documentation in portfolios.

Additional information and guidance: Candidates should describe one application for distributed version control referring to projects in which it is used, why the particular application was chosen and its strengths and any weaknesses.

2. The candidate will understand licensing and intellectual property.

I can:

2.1 explain the relationship between copyright and licensing.

Candidates should demonstrate they understand that any software or documentation is automatically copyright protected on origination and the license then says how it can be used.

Evidence: internal testing, documentation in portfolios.

Additional information and guidance: Copyright is becoming more and more important as a concept as information becomes the dominant world commodity. It is essential for anyone working in computing to understand copyright. Copyright works be giving the owner of the copyright power to license their work. traditionally this was “All rights reserved” meaning no-one could do anything with the work without the copyright holders permission. Now it has been realised that it is not always in the copyright holders best interests to be as conservative as this and a much greater range of licensing has emerged. As an example, Tommy Emmanuel a virtuoso guitar player has become extremely popular as a result of videos posted on YouTube. These videos would have been removed or prevented from being uploaded if the licensing regime for copyright of the late 20th Century had been applied and Mr Emmanuel would have made a lot less money. The nature of the internet means that copyright holders realise that the model of high degrees of control and throwing a lot of money at promotion does not work so effectively in a digital world. It is complicated and each product and circumstance can change the balance of risk and reward. For this reason there is now a vast range of different licenses for different products. Software licenses can be conservative as in “All rights reserved” and determining things like having to buy a now license if you buy a new computer through to very liberal, do whatever you like with the source code but don’t sue us if anything goes wrong. Candidates should also realise that if they originate work in the context of their employment it is likely that is is the employer that owns the copyright. This is because it is the employer that is paying them to originate the work. In some cases employers will have required ALL work originated by the employee to be owned by the employer even if it was done “out of hours.” There is clearly a lot of scope for legal disputes!

Some people refer to “copyright free” which would only really apply to work in the public domain. Such work has been put their by the copyright holder who has declared they have no longer any copyright interest in the work. Open Source software is NOT public domain and it is not copyright free, it just has a license that allows more freedom than

is usually the case with closed source software. There are many Open Source licenses and some are more liberal than others.

2.2 describe and explain the freedoms associated with free and open source software.

Candidates should be able to describe free and open source software in terms of the freedoms provided to users.

Evidence: portfolios and/or internal testing.

Additional information and guidance: This is an opportunity to research the background that has led to the success of the internet and much of its supporting technologies. There is a mass of information on the internet and GNU, Richard Stallman and the Free Software Foundation are good starting points. The 4 freedoms are:

0. The freedom to run the program, for any purpose (freedom 0).
1. The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1). Access to the source code is a precondition for this.
2. The freedom to redistribute copies so you can help your neighbour (freedom 2).
3. The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

These freedoms are also associated with Open Source software but different licenses will stipulate different conditions. For example the GPL requires all derived software as from freedom 1 to be redistributed (freedom 3) with the same license. In other words you can't just take the code make a few changes and re-license it as closed source or non-free software. The BSD and Apache licenses allow exactly this. Apple's iOS is derived from BSD and now restricted by restrictive licensing by Apple.

The Linux kernel on the other hand is licensed with the GPL. Note that free and open source software is not just about free of charge, it requires the 4 freedoms to be free and open source. In fact, you can charge for free software if you can get people to pay, the free of charge bit is a bi-product of the freedom part.

Candidates should be able to support their understanding of the 4 freedoms with examples from key implementations.

There are endless debates about the ethics of taking community developed code and then restricting it for individual commercial gain. On the other hand, having to develop every software application from scratch duplicating a lot of well covered ground is very inefficient and restrictive licenses also tend to generate technology monopolies. The better these issues are understood the more likely good decisions will be made in any particular set of circumstances. Certainly billions have probably been wasted globally to date because of lack of understanding of these emerging changes.

2.3 analyse the effects of digital technologies on the enforcement of intellectual property rights.

Candidates should be able to draw rational conclusions about the practicalities of intellectual property in a digital age.

Evidence: portfolios and/or internal tests.

Additional information and guidance: The key property of digital software technologies that makes it difficult to enforce intellectual property rights is the ease of copying it. Couple this to the fact that copies are absolutely identical with no loss of quality and “piracy” (unauthorised use of copyright materials) becomes very attractive. Publishers have tried many ways to prevent this with Digital Rights Management Technologies. The problem with DRM is that it is always possible to get round it, it adds to cost and inconvenience to users. On the other hand, without it why would publishers bother publishing

anything if it was likely they would never make a return? As usual, the devil is in the detail. In some circumstances, the only way to recoup the investment is to sell licenses and in order to do that the product has to be protected even if the protection isn't perfect. In others, as with Tommy Emmanuel cited above, by making the product freely available it eliminates the need for a promotion company ie a publisher as the artist can just do it themselves. If that promotes their concerts and they sell merchandise at the concerts they still make a good living. In the software development world big changes have taken place and continue to take place as a result of the way digital technologies affect the traditional commercial and economic agents. Microsoft Windows used to be the operating system on well over 95% of computing devices in 2004 and now less than 10 years later its 20%. While this is no doubt due to the rise of smartphones and tablets, the speed of development of those is almost certainly the result of ios and Android being built on free and open source operating systems when most of the work was done and could be freely used by Apple and Google. Apple had an initial headstart and went down a closed source route with a vertical market owned entirely by them. Google went down an Open Source route with multiple suppliers. At the time of writing the growth of Android looks likely to force both Apple and Windows into niche markets, but time will tell.

There is currently massive change taking place in the economic models that are most successful in the computing world and much of this is to do with the way intellectual property rights have evolved in the context of open systems computing.
There is a mass of information on this subject on the internet but analysis is often weak. Encourage candidates to be critical and analytical. Get them to debate issues and back their arguments with evidence. Larry Lessig is a very good source of insight into intellectual property and ethics in the digital age.

2.4 describe the advantages and disadvantages of software patents.

Candidates should demonstrate an understanding of patents in general and how these fit the software paradigm.

Evidence: Documentation in portfolios.

Additional information and guidance: A patent provides a monopoly for the patent holder for a limited time. Patents are traditionally granted for inventions and were devised before the age of software. Patents were never intended to cover books, music, artistic works or scientific laws, that is the realm of copyright. Patents have been granted for some very strange things, especially in the United States. Of course any patent can be over-turned if someone shows that the invention was not new but had already existed before the patent. This is known as prior art. The snag is that is costs money to do this and so fighting patent legal battles can be unrealistic for the small players that patents were originally designed to protect. Software is a bit of a grey area because it is already subject to copyright and has many of the features of works that have been copyrighted. On the other hand it has some of the characteristics of invention too if it is a brand new application doing an original job. There are some real practical problems with software patents. Is it realistic to be able to patent a software routine that puts a character on a screen? Imagine if one company owned such a patent and demanded say 1 cent royalty every time any computer wrote a character to the screen. Imagine a large company saying to a new start competitor your software infringes one of our patents we are not going to tell you which or which part of your software. Stop distributing your software or we will see you in court. That could ramp up costs that bankrupt the small company. Software patents are granted in the USA but not in Europe. This then affects competition across these two markets.

Candidates should make a study of the software patent issue and as a result write a 500 word summary of the case for or against. It would be useful to use the best ones to set up a formal debate on the subject.

2.5 explain the terms trademark, copyleft, creative commons, and public domain.

Candidates should be able to explain each of the terms in the context of open systems computing.

Evidence: Documentation in portfolios.

Additional information and guidance: These terms can be learnt in the context of other work in the unit and reinforced by writing a summary of each in their portfolio documentation in their own words.

3. The candidate will understand commercial models for software development.

I can:

3.1 describe the dual licensing model for software development.

Candidates should research and provide a description of such a project.

Evidence: Documentation in portfolios.

Additional information and guidance: Dual licenses are where there are two different licenses eg one for commercial organisations who have to pay and the other that is for not for profit or personal use that is free of charge. Free of charge in itself does not mean Open Source. A dual license strategy might be used to get a big take up to generate confidence quickly and then to concentrate on the development of the non-free product once established.

An example of a large project that uses dual licensing is Oracle's MySQL.

3.2 describe the software as a service model.

Candidates should research and provide a description of such a project.

Evidence: from portfolios.

Additional information and guidance: Word press is and open source example, Google Apps is a proprietary example**.** SaaS is is any software of specific functionality which is intended to carry through a browser over a remote feed, rather than run from a local machine or local network.

3.3 describe the perpetual license model for software development.

Candidates should research and provide a description of such a project.

Evidence: Portfolios of evidence.

Additional information and guidance: A perpetual license allows you to use the particular version of the software for ever. MS Windows XP has such a license. In practice the software will become unsupported so it is unlikely anyone will actually use such software for an indefinite time. When microcomputers first emerged this was the most common form of license but it has now been challenged by an increasing range of business models at least in part due to the rise of the world-wide web.

3.4 describe the fremium model for software development.

Candidates should research and provide a description of such a project.

Evidence: Portfolios of evidence.

Additional information and guidance: A freemium business model provides a proprietary product or service free of charge, but charges for advanced features, functionality, or virtual goods. LinkedIN is an example, The basic service is free but you can pay for additional features. Note that a lot of freemium models are also software as a service models and vice versa.

3.5 describe an advertising model to support software development.

Candidates should research and provide a description of such a project.

Evidence: Portfolio.

Additional information and guidance: The most obvious example is Google and its search engine. This is not so much a project as an entire company. The revenue from advertising enables Google to give away products that its competitors charge for. That gives it long term sustainable competitive advantage. Any strategically useful application that will attract people will generate advertising revenue so there is no need to charge license fees for it. Google can use its competitors to spread its advertising by simply letting them use their products.

Unit 5: Computer Systems Management

1. The candidate will set up systems.

I can:

1.1 install and set up an operating system.

Candidates should be able to install a new operating system on a computer device and deal with common issues.

Evidence: Assessor observations and documentation in portfolios.

Additional information and guidance: A good practical way to do this is to use more than one Linux distribution on more than one device. This will show candidates that there are common principles and give them confidence in any unfamiliar new situations. Candidates should be confident to install an operating system and know that they need to be sure that the hardware drivers are available to make it operational. Using a live CD or USB installation is a good way to check this because it means the proposed software can easily be removed in a situation where eg the screen drivers fail so you can't see any display output.

1.2 set and customise boot sequence and options.

Candidates should be able to set bios boot options and any further options that can be adjusted as the machine starts up.

Evidence: assessor observations, documentation in portfolios.

Additional information and guidance: Candidates should have practical experience of accessing the BIOS at start up and know what the main options are. This varies a bit from machine to machine but it will normally include the boot sequence for devices so for example the machine can be set to boot from a USB key, the network or a DVD. Date and time are other possible settings and the BIOS will provide information about hardware. It is also possible to set a password for accessing the bios but for personal use this is probably a risk not worth taking. If you for get the bios password it will be necessary to reset it and that is usually using a pin on the motherboard that might not be obvious. In the case of Linux systems, the boot loader eg Grub or Lilo is

started by the BIOS and parameters can be changed to fix boot problems. Internet searches will often result in finding any fixes for particular hardware. It would be well worth candidates examining some bootloader files to see what they are doing. These files can be edited in a text editor but of course it will be difficult to get to them if the machine won't boot.
If there are RaspberryPIs available for practice, the link at provides instructions about how to set up a SD card to boot the machine. http://elinux.org/RPi_Easy_SD_Card_Setup#SD_card_setup

1.3 customise the display to personal preference.

Candidates should be confident to make display adjustments and be aware of display issues with different display hardware.

Evidence: from assessor observation and documentation in portfolios.

Additional information and guidance: Candidates should understand the background principles of display technologies and relate this to the work on computation in unit 1. Spacial resolution, colour resolution and refresh rate. Typically, modern displays use 8 bits for each of the 3 primary colours, Red, Green and Blue. The graphics cards driving the displays can use 10 bits per colour and can use an extra field for transparency but in most cases 8 bit is sufficient . So it would be possible to have 256 levels of transparency for each of 16 million (256 x 256 x 256) colours. That would take 4 bytes per pixel. 1 byte for each of RBG and 1 for transparency. If a screen is 1000 pixels by 1000 pixels that is 1 million and 4 Mbytes of storage is needed. Screens are constantly refreshed to show moving pictures. If this happens 100 times a second it will not be noticeable to humans because our eyes take time to refresh too. So 100 Hz refresh rate means 400Mbytes of data every second. The higher the resolution and refresh rates, the faster the hardware has to be to get the data onto the screen. Another property of the display is the aspect ratio. This governs the height and width of the screen. Usually the width dimension is greater than the height dimension. In High definition TV the vertical height is 1080 pixels and the horizontal spatial resolution 1920 pixels. This is referred to as 1080p. With the convergence of computer TV and mobile technologies, these numbers are becoming more common in all technologies. When choosing a display for a computer, the computer graphics hardware has to be able to cope with the demands of the display. If the refresh rate of the display is too high for the graphics hardware there will be no picture.

If the display has a lower resolution that the computer hardware (not uncommon with data projectors) the computer has to reduce its output to meet the needs of the projector. In some cases it is possible to connect two displays with different resolutions to the same computer. Candidates should experiment with display options to be confident about situations where they can and can not make appropriate adjustments. Other aspects they will be able to customise include background image, screensaver, positions and visibility of toolbars and possibly the entire window theme.

1.4 set up network connections.

Candidates should be able to set up straightforward computer networks understanding the issues with network connections.

Evidence: assessor observations and documentation in portfolios.

Additional information and guidance: In most cases the connection will be automatic and so candidates need to understand more in case anything does not happen as anticipated. They should have the opportunity to set up client machines on a network . They should be familiar with network connection types including wireless, UTP cable and fibre and the advantages and disadvantages of each. They should understand that on the internet, a network connection requires an IP address, and that IPv4 and IPv6 are the descriptions. They should relate these to the work on binary numbers in unit 1. They should know about the DHCP protocol for assigning IP addresses, and the DNS for assigning IP addresses to meaningful names. Candidates should be able to use a background knowledge of how networking works to troubleshoot network problems.

1.5 solve problems in systems setup and configuration.

Candidates should use the knowledge they have gained and internet searches to help solve problems in systems set up and configuration.

Evidence: Assessor observations and portfolio documentation.

Additional information and guidance: Candidates should have the opportunity to solve typical problems such as a machine that refuses to boot, a faulty cable connection, a failed hardware component or similar. These can be set up on old computers as challenges to be resolved.

Candidates should be encouraged to be systematic in their approach to fault finding.

2. The candidate will support system storage and security.

I can:

2.1 format and partition storage devices.

The candidate should be able to format and partition storage devices understanding the risks of losing data in doing so.

Evidence: Assessor observations and documents in portfolios.

Additional information and guidance: Candidates should understand why discs need to be formatted and partitioned and relate this to concepts of computing such as binary numbers and data patterns. Practical experience can be gained with old computers and operating systems such as Linux distributions.

2.2 devise and implement a backup strategy.

Candidates should be able to match a backup strategy to risk, convenience and cost.

Evidence: documentation in portfolios.

Additional information and guidance: Candidates should carry out a research study of backup strategies and summarise this in a 1000 word portfolio document with an emphasis on cost, risk and security.

2.3 set up and understand how to customise a firewall for network connection.

Candidates should understand the principles of firewalls and understand how these work.

Evidence: assessor observations and portfolios

Additional information and guidance: Most PC operating systems include software-based firewalls to protect against threats from the public Internet. Routers that pass data between networks contain firewall components and firewalls can perform basic routing functions. Candidates should be familiar with the basic ports that can be opened or blocked by a firewall and three key firewall methods.
Packet filtering - Packets of incoming data compared to a set of filters. Packets that make it through the filters are sent to the requesting system and all others are discarded.
Proxy service - Information from the Internet is retrieved by the firewall and then sent to the requesting system and vice versa after checking.
Stateful inspection - Compares certain key parts of the packet to a database of trusted information. Information traveling from inside the firewall to the outside is checked for specific attributes, then incoming information is compared to these and if there is a reasonable match, the information is allowed through. Otherwise it is discarded.

2.4 write a risk assessment for system security including passwords and malware.

Candidates should write a risk assessment for a familiar computer installation identifying and prioritising risks.

Evidence: Documentation in portfolios.

Additional information and guidance: The candidates risk assessment should include the major risks to a computer system connected to a network. This could be a mobile phone network, the internet or LAN. The important thing is for the candidate to research the nature of the use of the network and where the security vulnerabilities are greatest including the possibilities of human error. They should test drafts of their risk assessment with peer review and make amendments accordingly before finalising it. Assessors should provide feedback on the strengths and weaknesses of their final work.

2.5 describe a range of storage devices and their strengths and weaknesses.

Candidates should carry out a survey of storage devices and provide comparative descriptions of them in terms of their properties and cost.

Evidence: Documentation in portfolios.

Additional information and guidance: As a minimum candidates should produce a table of devices covering electromechanical, solid state and optical media. They should understand at least some of the reasons why there is such a diversity and consider whether some are devices are destined for obsolescence and if so why.

3. The candidate will maintain systems.

I can:

3.1 install software updates and dependencies.

Candidates should understand why it is important to keep software up to date and be practically capable of taking responsibility for doing so.

Evidence: Assessor observations and documentation in portfolios.

Additional information and guidance: Regular updates and patches help reduce security risks and ensure bugs get eliminated. Different systems have different methods of providing updates but most are automated.

3.2 set up cron jobs to automate regular procedures.

Candidates should understand the concept of cron jobs to automate regular actions and how to practically implement them.

Evidence: from assessor observations and portfolios.

Additional information and guidance: On Windows a cron job is a scheduled task. Candidates should ideally see how to setup cron jobs

on more that one system and both on graphical and command line systems. One obvious candidate for a regular automated process is backup.

3.3 set up a secure virtual connection to manage a system from a remote location.

Candidates should be able to manage a computer remotely using a secure remote connection.

Evidence: Assessor observations and portfolios of evidence.

Additional information and guidance: There are a number of different ways of achieving this. A good starting point is http://en.wikipedia.org/wiki/Virtual_private_network

3.4 install and remove applications.

Candidates should be confident to install and remove applications.

Evidence: Assessor observations and portfolio of evidence.

Additional information and guidance: Candidates should understand the need for trusted sources for applications that they install. It is highly inadvisable to install any executable file if it is not from a secure and trusted source. Great care especially needs to be exercised on the internet to make sure that web sites are genuine. They should also know the importance of using the correct tools to remove software since simply deleting files is dangerous owing to possible dependencies. They should also understand that there are potential IPR issues when installing software.

3.5 provide effective support for system users.

Candidates should be provided with opportunities to support other less experienced users.

Evidence: Assessor observations and portfolio documents.

Additional information and guidance: Candidates could spend time in a junior school, local business or similar environment providing support.

This activity would be most appropriate in the second half of the course when they have had more time to learn.

4. The candidate will understand key internet systems.

I can:

4.1 describe the terms HTML, W3C and HTTP.

Candidates should understand these terms well enough to provide general explanations illustrated by examples.

Evidence: Internal tests and portfolios of evidence

Additional information and guidance:
http://www.w3schools.com/html/ This link will provide all the information anyone needs about HTML. The depth for a particular student will depend on the nature of their interests eg in their main project. All candidates should have an idea about how HTML works including very common tags such as <h1> and </h1>, links to pages and other resources such as images and structure through elements. They should understand the principle of styles and CSS but a detailed knowledge of key words and syntax is not required.

The W3C mission is to lead the World Wide Web to its full potential by developing protocols and guidelines that ensure the long-term growth of the Web. Candidates should be familiar with how W3C came about, its aims, values and role in maintaining the web.

HTTP is a key protocol familiar from its role in browser URLs. It is a good example of a widely used protocol that is open and royalty free.
There is a comprehensive description at
http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
Candidates should understand the principles but will not be expected to memorise the details.

4.2 explain the function of a web server.

Candidates should be familiar with the basic functions of a web server.

Evidence: From internal tests and portfolios of evidence.

Additional information and guidance: There is a reasonable overview at http://en.wikipedia.org/wiki/Web_server Apache has by far the biggest market share and is open source. It would be a good example on which to gain some practical experience.

4.3 explain the importance of TCP/IP.

Candidates should explain how TCP/IP is a fundamental protocol for the internet and use it as an example for computational thinking.

Evidence: Internal tests and portfolios of evidence.

Additional information and guidance:
http://en.wikipedia.org/wiki/Internet_protocol_suite
While the details are complex and not expected, candidates should have some basic concepts such as layers and this is a good opportunity to reinforce the concept of abstraction. It should be increasingly obvious that these very complex systems are fundamentally made up from binary patterns. Concepts such as layers and objects enable us to reduce the amount of detail anyone has to consider at any time. Someone has to know all the details of specific parts but no-one needs to know all the details of all the parts.

4.4 explain the effects of proprietary standards and lock-in.

Candidates should understand the relationship between control over particular standards and the capacity to establish a monopoly.

Evidence: Internal tests and portfolios of evidence.

Additional information and guidance: Candidates should understand that the greater the dependency on a standard the easier it is for the owner of the standard to determine the price paid for using the standard. If one company owned the patent for ASCII numbers it would mean that all text stored on every computer would be locked into that company unless everyone made an effort to translate all those characters to a different system and modified all the software that operates on text. How likely would that be to happen? Probably if a royalty of £1 a character was charged change would occur because without it, it would probably bankrupt much of the world. On the but not be damaging other hand £1 for 10,000 characters might make the company a fortune but not be damaging enough to precipitate change.

The barrier to any competition would be high simply because of the dependencies of so many systems on the ASCII standard. Companies and governments have taken a long time to realise that there is a solution to these issues by demanding open systems, from protocols to data formats and even computer application code. HTML is an excellent example. If this had been patented and royalties charged, the power given to the owner over everyone else would have been enormous. The owner of HTML could also have become the owner of web browsers, web servers and a whole range of other technologies. With no competition development and innovation would have suffered. Look at the rate of growth and innovation in the mobile computing space where there is far more competition than in the desktop space.

4.5 explain the role of an internet service provider.

Candidates should be able to explain how ISPs operate, and the range of services that they provide.

Evidence: Assessor observations and portfolio documents.

Additional information and guidance:
http://en.wikipedia.org/wiki/Internet_service_provider
Internet service providers can provide a range of different services on a range of business models. Candidates should understand the boundary between traditional service provision providing access to the internet has now evolved to web hosting and consequently applications hosting. The gradual shift to Cloud Computing makes internet service provision and software as a service, increasingly likely to dominate. Candidates should consider this in the context of lock-in to proprietary standards. Perhaps a bigger danger is having all your data locked into a particular provider.

Annexe D Summary of the units and their credit and guided learning hour recommendation.

Unit 1: Computational Thinking
10 credits (75 GLH)

Unit 2: Principles of software engineering
10 credits (75 GLH)

Unit 3: Delivering a software project
10 credits (60 GLH)

Unit 4: Open Systems and Community Development
10 credits (75 GLH)

Unit 5: Computer Systems Management
10 credits (75 GLH)

Any of units 1, 2, 4 or 5 to gether with the Level 2 unit IT Security for Users will provide the **Award.**
12 credits and 90 GLH.

Units 2 and 4 + the Certificate examination are required for the **Certificate.** 20 credits and 150 GLH.

All units + the Diploma examination are required for the **Diploma.** 50 credits and 360 GLH.

Units can be assessed concurrently or consecutively enabling the school to decide how to organise teaching and progression routes. There is no requirement to do any particular combination of these qualifications.

Annexe E - Useful links and supporting information

The INGOT community learning site www.theINGOTs.org has a wealth of supporting information and practical tools for managing evidence, progress tracking and reporting. These are all free for participating schools. Contact TLM for further details or training if required. We will update and add to supporting materials as time goes on.

The INGOT web site supports multiple languages and it is not very difficult to provide new translations. If you want to teach in the context of a modern foreign language it is possible and we will provide support where we can.

Annexe F – Involvement of Employers

In the Certificate and Diploma it is mandatory to involve employers in the assessment process. There are some obvious opportunities for this. Ideally the candidate should gain some significant work experience through a placement in a company where they can make a meaningful programming contribution that can contribute to Unit 2 or Unit 3. If this is not possible, employer involvement can be associated with specific assessment criteria, for example in Unit 3

1.3 present the proposal to critical experts.
1.4 make modifications as a result of feedback.
2.5 test code regularly involving third parties.
3.1 provide regular updates on progress to a mentor.
3.5 make a final presentation to a critical audience.

There are also opportunities for candidates to earn money from open source code contributions eg for those 18 or over Google Summer of Code Stipends. This could be the extension of a project after the qualification has been completed forming part of an apprenticeship. This is in keeping with the Richard Review recommendation 6.

The Government should encourage diversity and innovation in delivering apprenticeships. There will be many paths and approaches that an apprentice can take to reach ‘the standard’ and we should strip out any unnecessary prescription and regulation of the process for getting there. (Recommendation 6 of the Richard Review).

Annexe G - Coursework assessment flowchart