Version 1.0 0711

General Certificate of Education (A-level)
June 2011

Computing COMP4

(Specification 2510)

Unit 4: The Computing Practical Project

Report on the Examination

Further copies of this Report on the Examination are available from: aqa.org.uk

Set and published by the Assessment and Qualifications Alliance.

The Assessment and Qualifications Alliance (AQA) is a company limited by guarantee registered in England and Wales (company number 3644723) and a registered charity (registered charity number 1073334).
Registered address: AQA, Devas Street, Manchester M15 6EX.

General

Centres that entered candidates in 2011 should read this Report in conjunction with the specific feedback sent to them on the publication of results. If you do not receive this feedback, please consult your Examinations Officer in the first instance before contacting the AQA Subject Manager. The comments below highlight the observations of Senior Moderators during this year's examination and should be used with the subject specification, assessment criteria and the COMP4: Advice and Information Booklet for the Teachers' Standardising Meetings to ensure that centre assessment is accurate. The latter can be found in the Secure Key Materials section of e-AQA.

There were no changes to the marks awarded for each section of the assessment criteria or the total mark compared with last year. However, some changes to the awardable marks for the Analysis section were made in response to teachers' comments in 2010.

The changes to the COMP4 Analysis assessment criteria were made for the following reasons:

- to provide criteria for Complex Problems 0 to 6 marks, inclusive.
- to provide criteria for Problems of Adequate Complexity 0 to 3 marks, inclusive.
- to make more positive all the criteria for 0 to 3, inclusive.
- to make the criteria for 4 to 6 marks more demanding for Complex Problems and Problems of Adequate Complexity (this was a regulatory requirement).
- to make clearer the terms 'less SMART objectives' and 'less than high level perception of a real end user's needs'.

All of these changes were intended to ensure greater clarity, coherence and accessibility for teachers and candidates. A copy of the revised COMP4 criteria appeared on the Noticeboard of the GCE Computing pages on the AQA website in August 2010. All changes, including changes to the subject content, appeared in the specification on the AQA Website by 1 September 2010 and letters were sent to all existing centres via the Examinations Officer.

In addition, there was the possibility of assessing the Complexity at the Analysis stage at a different level to all of the other sections in response to the small minority of centres in 2010 which felt that the initial problem was undeniably 'Complex', but one or more of their candidates' implementations had fallen far short of meeting the stated objectives.

These changes to the COMP4 assessment criteria were further clarified at the series of Standardising Meetings for the 2011 examination which were held during Autumn 2010.

Most centres had encouraged their candidates to write a full User Manual dealing with the functionality of the **whole** system. However, the most important task is the need to correctly identify the complexity level of the problem and its solution from the outset at the Task Setting/Pre-analysis stage.

This is the second year that the COMP4 specification has operated. Most centres, including new ones, had taken on the requirements of the new specification and managed to get their candidates to program a solution at the level of which they were capable. However, there were some cases where it appeared that much time had been spent on coding because of the complexity of the problem and the Analysis and Design sections were again probably written post-implementation when time was running out. It also appeared that the User Manual and Appraisal sections were also often casualties of poor time management by the candidate. Appraisal was also frequently ignored or very poorly written.

Centres should **not** encourage their entire cohort to produce CAL systems with just differences in subject area. This happened in a minority of cases and one centre allowed three projects based on non-trivial Calculators, with some differences. A variety of projects is expected from individual centres.

Some ingenious non-data processing projects were seen this year including real world Complex problems involving Steganography, Processing of Oil Drilling data and Simulation of the Haber Process. However, data processing projects are still most welcome and often make it easier for candidates to find genuine end users.

Internal standardisation was generally well carried out. Pleasingly, there was no evidence this year of Local Assessors marking only individual sections for their centre without reading through the whole project for each and every candidate – a practice which is to be discouraged. There was some evidence that the standardising training received by one of the Local Assessors had not been cascaded to others at the centre.

Despite the potential for high scoring Complex part-packaged/part-coded solutions, the better projects this year seemed to be the ones where candidates had fully coded the solutions, but very good solutions were also seen with genuine client-server applications. Most centres have moved further away this year from customised Microsoft Access projects with minimal coding as was intended when AQA introduced the new specification.

Administration

Centres that accurately complete the Candidate Record Form (CRF), the Project Log Sheet (PLS) and the Centre Declaration Sheet (CDS) can really assist in the moderation process. Moderators always seek to agree with the Local Assessor; this is very difficult to do if all that is provided are the section marks and final total on the CRF and perhaps a perceived level of complexity.

The CRF needs to be fully completed, including signatures on page 1 and section A on page 2. It is permissible for a centre to provide its own marking grid for the separate assessment criteria marks and total mark in place of section B on page 2 of the CRF. A small minority of centres failed to indicate their perceived level of complexity in the 'Concluding comments' section on page 2 of the CRF; this does not help the candidate. If there is difficulty assessing the complexity of a project, section 6 (Reassessing complexity) of the Teacher Resource Bank document, *'Definition of problem types for projects'* may help. The Local Assessor (not the candidate) must select **one** level of complexity and justify it by reference to the criteria described in *the 'Definition of problem types for projects'* document eg 'This candidate has used an algorithm which was more complex than O(n squared),' or, 'They have used principle FC2 – user defined data structures and classes,' etc. This justification is expected to be written here if it is not clearly explained elsewhere on centre-designed documentation (if this has been done, please refer the Moderator to it). It is not appropriate only to make statements like, 'The candidate worked really hard on this project and produced some complex code.' Centres that have devised their own comment sheets are also to be commended. However, these still need to be attached to the CRF for the candidate concerned. Also it is still a **mandatory** Ofqual requirement that the signatures required on the CRF are indeed present.

On the Project Log Sheet, comments do not need to be written in full, but the candidate does need to tick the boxes and give the page number(s). The Local Assessor also needs to tick the boxes that they think are covered, possibly correcting the candidate page numbers as they go. Many centres used the PLS very effectively to explain their rationale for how they had awarded their marks.

The CRF and PLS **must** be the current versions for the examination series. These may be obtained from the AQA website (revised versions will be introduced for the 2012 examination).

A number of centres failed to send in Centre Declaration Forms or did not have them countersigned by the Head of Centre. Some centres also had incorrect additions on their Candidate Record Forms.

There were some very effective examples of script annotation by some Local Assessors. It is useful to indicate particularly important sections of code elsewhere as well as just on the particular page of code.

Centres need to be aware that legacy examination CPT3 contexts must *not* be used for A2 projects because AQA completed the Analysis section for CPT 3 assignments.

Projects need to be bound as a booklet, ideally with two treasury tags or spiral binding to enable them to be read easily. Some projects arrived in a completely inappropriate state eg in a loose sheet state in document folders or in very large ring binders.

A significant number of centres sent in work late, without the prior approval of AQA. This strategy risks the late publication of centres' own candidates' results which may have serious consequences personally for them during the UCAS Clearing process. Centres are reminded that it is **their** responsibility to get the projects to the Moderators by the deadline date of 15 May.

Project must be sent to the Moderator using only **first class post,** retaining proof of postage. Projects sent by recorded delivery (which requires a signature upon delivery) risk being lost or returned to the sender if the Moderator does not have the time or the means to travel to the (sometimes distant) distribution depot to pick up a heavy parcel. Please do not send projects using a method that requires a signature because many Moderators teach full time or have other commitments and cannot be expected to collect work from the courier. Please also ensure that sufficient postage is attached to parcels.

Software

The majority of candidates used Delphi or a version of Visual Basic /VB.Net. Other languages seen included VBA, C # or C++, Java, Python, QT, ASP, PHP, HTML and Actionscript 3. HTML and CSS may have formed part of the solutions presented, but they needed to be combined with one or other of the other programming languages.

Candidates completing a project entirely using a high level language accounted for most of the very high scoring projects, but a number of very good client-server systems were also seen involving PHP and MySQL for example. Although a part-packaged/part-coded approach using Delphi or VB combined with MS Access or SQL Server certainly has the potential to produce high marks, few of these were seen.

There continues to be a problem with coursework written in Actionscript this year. Although the implementation may function well, frequently there was very little candidate-written code with animation being carried out by Actionscript, not the candidate. The later versions of this language support OOP and sophisticated programming constructs and these need to be used to demonstrate high technical competence.

PHP-based projects created problems with regards to authenticating code and assessing complexity. This made assessment difficult for centres where the candidate had taught themself how to code in PHP and the Local Assessor was in no position to judge the level of

technical competence achieved. PHP-based projects are to be encouraged but care must be taken to assess these appropriately. There are many code templates on the Web for candidates to use. Candidates can be credited for adapting these to the needs of their project but they must acknowledge their source. Judgement of the degree of technical competence should be dependent on the complexity of the original source and the need for the candidate to understand the source in order to adapt it to their needs as well as the degree of adaptation. Selecting complex libraries/code templates is itself complex if the learning curve is steep and high technical skill is needed to use the libraries/code templates, eg OpenGL graphics.

Project development

Low scoring candidates still tended to submit simple data processing projects with little complexity, but most of these were assessed appropriately by the centres.

A very small minority of candidates produced static websites, not client-server applications, which were similarly heavily penalised and are not appropriate to the specification.

There continues to be far too many candidates who pursue a project that merely used either a fully coded or a part-packaged/part-coded solution to enter data, save it, update it and to produce simple reports with little or no data transformation. If it is a part-packaged/part-coded solution using MS Access for example, all validation must be carried out in the code, not by using the Masking /Validation built into the package. QBE must not be used to design and run queries; all queries need to be hard coded in the programming language. Similarly, the Report Wizard should not be used. However, some of these techniques might be useful in an earlier prototyping stage. Regardless of the programming language used (even if it was a fully coded solution) this 'Data Strategy' type of project is not complex enough for A2 and cannot score a good mark however well the report is written and however hard the candidate worked to produce it.

Project reports

The majority of the high scoring candidates used a programming language or a combination of a package and substantial programming to demonstrate good coding skills. They also produced well-tested, well-documented, effective solutions to real problems, with corroborated end user feedback. Particularly good practice was seen where the agreed objectives had been signed and dated by the end user.

Candidates who followed the reporting style of the specification and followed this up by completing the log sheets fully usually justified their assessment. The majority of centres used the Candidate Record Forms and Project Log Sheets or their own centre-designed documentation to record a qualitative indication of the candidates' achievements, rather than to record how much effort a candidate has put into their work.

Most candidates achieving a high mark addressed fully all the items listed in the specification **in the context of their projects**. Some headings are not relevant to a particular project or just require a brief comment as to why they are not appropriate. The project report is not the place for general theory. As in previous years, those who scored the highest marks included well reasoned and justified explanations of all aspects that are listed as indicators in each section of the specification and, in particular, good programming techniques which, together with a high performance level in the other sections, tended to be the features that most distinguished the high scoring candidates from the others.

It would assist the Moderators greatly if items are addressed by candidates in the same order as the specification and with the same headings, especially when essential items appear in

one or more appendices without accurate references as to where these items can be found. Sampling as described in the subject specification reduces bulk. There should not be lots of appendices or projects submitted as seven or eight individually treasury tagged booklets.

Analysis

It cannot be over emphasised just how important this section is to set the scene and confirm the complexity of the project. A number of candidates provided a proper, structured investigation using one or more formal techniques including interviews, authenticated questionnaires, observation and the use of analysed source documents. It is not appropriate for the candidate to assume that their personal view of a problem provides sufficient detail. This happened in the case of a number of Complex projects scoring high marks for Technical Solution. What is required is a careful analysis with appropriate evidence of the current and the proposed systems.

The best examples of good practice contained a comprehensive list of highly detailed specific objectives constructed from more than one interview with the end user. The specific objectives were then improved, clarified and checked for completeness after an initial prototyping stage involving feedback from the end user. The finally agreed set of specific objectives were agreed, signed off and dated by the genuine end user. The best examples of projects also contained an outline of the proposed solution using dataflow diagrams and a careful consideration of the proposed and alternative solutions. The latter is an aid to confirming the complexity level of the problem.

It should be noted that analysis, design, implementation can be carried out in prototyping mode from day one of the project leading to a more and more complete picture of what needs to be done, ie the final analysis., and a clearer idea of how to do it, ie the final design. The structure of the write up follows the Waterfall model, specified in the specification, but the detail of each stage can be achieved very effectively by following a prototyping approach. Staging the write ups of each stage of analysis and design so that sufficient prototyping can be carried out involving the end user often leads to a better understanding than an approach in which no design/coding begins until the analysis has been signed off.

The analysis data dictionary still remains a problem area with many candidates ignoring it or just copying the one presented in Design. The analysis data dictionary captures the data items and their properties that appear on the artifacts of the current system, eg a paper based booking form and that which will be required in the proposed system. The properties will include such items as length, range and size of fields, data types of fields and example values. The data types may come from any of the formal methods employed and will be as perceived by the end user, eg a whole number, a decimal number with two decimal places; not the data types that will be used in the chosen programming language eg Long Integer, Float(5,2) which will be stated in the design data dictionary.

Numerous candidates totally ignored the section on data volumes even though it was relevant in their case because they were tackling a data processing problem either fully coded or alternatively part-packaged/part-coded. In some cases, the data volumes may be relevant in determining the level of complexity of the project. If it is not relevant for a particular problem, candidates are expected briefly to justify this reason for its exclusion.

Not all candidates produced DFDs for the existing system even if they did for the proposed system. Candidates should be encouraged to use standard notation and labelling especially for DFDs and ERDs. Too frequently there was no data labelling of the flows where these were not obvious, but sometimes actions were given. An adaptation of DFDs in which the datastores perform the role of event/message stores and the processes perform the role of actions that respond to specific events/messages is perfectly acceptable.

Many candidates produced clear and measurable objectives that were specific to their projects rather than generalised, 'text book' type objectives such as 'user friendly' or 'system must load in a certain number of seconds' (which are not actually wrong in themselves). 'SMART' objectives as defined in the current specification enable candidates to achieve higher marks in both this and the Appraisal section. An example of a specific objective is, 'The system must allow the user to enter the starting x, y coordinates of the object specified in specific objective 12.' An example of a constraint specific objective is, 'The system must be able to project through a data projector of maximum resolution 800 x 600.' The objectives of some candidates often indicated little processing, but centres still claimed these to be 'Complex'. The complexity of a problem can be assessed by examining the specific objectives, the dataflow diagrams of the proposed system, the data model where the latter is appropriate, an outline of the proposed solution and by discussion with the candidate. This becomes more difficult if the candidate fails to provide sufficient detail. In this case, examination of the design and implementation should throw light on the complexity level. The process can be helped by an initial prototype and identification of the critical path for the project, ie how difficult is the most difficult path in the project? For example, synchronising files on two geographically separated computers connected by the Internet. The critical path in this case is reading the properties of a remote file, ie data and time of creation. and transferring a copy of the most up-to-date file to the other computer. If this is clearly stated as a specific objective(s) then it becomes easier to assess the project's complexity because the steps that contribute to the critical path are sufficiently articulated for an assessor in the centre to focus on the problem(s) that must be solved for the project to succeed.

Many identified end users (client-users) seemed to have very little influence on the specific objectives and could provide little or no meaningful feedback in the later Appraisal section. A few candidates got their end user to sign off and date the agreed objectives. This is good practice. Candidates producing clear and measurable objectives that were specific to their project allowed the complexity of the project to be clearly identified. Vague, non-specific, non-quantifiable 'specific' objectives are unhelpful at A2 level as they would be at any level seeking to meet specific goals.

'SMART' objectives enable candidates to achieve higher marks in both this and the Appraisal section. The basic purpose of their system should be included in their understanding of the general objectives eg, 'To produce a system which helps teach the Advanced Level Mathematics Topics of Matrix Transformations,' or, 'To produce a system which helps teach the topic of The Haber Process in Advanced Level Chemistry.'

Please note that the awardable mark for Analysis is linked to the level of complexity of a project. If a centre determines the level of complexity as anything other than 'Complex', they cannot then award marks in Band 4 for Analysis.

After reading the Analysis section, the conclusion should be: 'Yes, I know what the aim and scope of this project is.' If this is not the case, and/or all the relevant Analysis assessment criteria are not fully met, then the project cannot warrant a mark from the top of the range for the complexity level assigned to the project.

Design

In the past two years, we have been trying to widen the range of project types submitted. Traditionally when the majority of projects were data processing projects, an emphasis on data validation and data security was deemed both relevant and important. However, a more important requirement is, 'Does the solution/ algorithm meet its specification under all circumstances?' For an algorithmic type of project (one that necessarily seeks out one or more suitable algorithm(s) to solve the problem) the emphasis should be on the 'correctness' and effectiveness of the algorithm.

Design is about how to solve the problem in general and in detail. The quality of the design is judged in two ways. Firstly, by the structure of the solution, ie how good is this structure? Secondly, by how well the design is described, ie could the description be used successfully to produce the solution? The first relates to planning and the second to how well the planning is reported.

Some designs were well planned and showed a high level of understanding of the requirements of an A2 project. Many candidates failed to score high marks by not conveying an understanding of the concepts of validation and security beyond a very superficial level, such as field length or presence checks. The high scoring candidates showed a good appreciation of data structures, object-oriented design of classes and objects, file design and normalisation where this was required for their solution. The normalisation process can be demonstrated by showing how the data model develops from the analysis data dictionary and the ERD shown in that section which may contain unresolved many to many relationships to 3rd normal form using standard notation, or by confirming that the entity descriptions derived from the E-R diagram are in 3rd normal form or BCNF using one of the tests, eg every non-key attribute is a fact about the key, the whole key and nothing but the key. Some candidates are still failing to show proof of normalisation and simply producing ERDs, often incompletely labelled, especially for the relationships where labelling would add clarity.

There seems to be a great reluctance to show the HCI design as clearly annotated sketches or by the use of a graphic package or the form designer related to the chosen programming language. This allows the candidate to give clear explanations as to the rationale behind the input and output screens, and how they meet the user needs. Chapters 7.2 and 7.6 of the AQA endorsed A2 text book[1] contain guidance on HCI. The HCI design should also convey guidance in pictorial, annotation and prose forms that will enable the HCI design to be implemented. For example, the rationale may cover the reason for choice of fonts, font sizes and colours for text, foreground and background as well as specifying these. Really good practice was demonstrated by the genuine end user signing and dating the agreed final designs. This provided evidence of end user involvement at various stages of the project. HCI design involves the end user and other potential users. HCI design is necessarily an iterative process.

A number of candidates appeared to produce extensive post implementation designs without even claiming these to be ‘prototypes’. Such ‘prototypes’ should have corroborated end user involvement by being signed off and dated.

Very few candidates gave clear algorithms relating to the data transformation to be programmed by the candidate. These algorithms should be in a form that could be coded in any language and need to concentrate on the complex parts of the system that the candidate is going to code. Frequently, the ‘sample’ was of very basic functions. We are not very interested in the algorithms required for login, form navigation and password changes etc. Algorithms in pseudocode or structured English must be given if a high mark is to be scored. Those that were given were generally poor with the notable exception of some centres whose methodical approach made moderation of this section relatively simple.

Frequently ‘algorithms’ were post implementation code extracted from the full code listing, not in pseudocode or structured English as required. The algorithms presented in this section are one of the places that Moderators inspect closely at an early stage to confirm the potential level of complexity of the project. It is not appropriate to omit design algorithms and forward reference those given in the System Maintenance section because the purpose of the Design section is to allow a competent third party to implement the required system.

[1] 'AQA Computing A2' by Kevin Bond and Sylvia Langfield, published by Nelson Thornes ISBN 978-0-7487-8296-3

Candidates were penalised heavily for doing this because the design is simply not feasible without design algorithms.

If candidates are going to need to embed SQL queries in their programming code because they will be creating part-packaged/part-coded solutions, it is appropriate to include the designs for these queries in the design section. Similarly, DDL script designs for table creation should be included here as appropriate.

Security, integrity and test strategy are problem areas as many candidates are still not addressing these in the context of their system, but presenting general theory rather than applying it to their particular projects where relevant.

Technical solution

This section is also closely linked with the level of complexity as well as the level of technical competence displayed. Many candidates programmed a solution at the level at which they were capable, but again there was a great deal of evidence of much time being spent on coding and only when this was completed or when time was running out were the Analysis and Design sections finally attempted. The complexity demonstrated in the Technical Solution - eg appropriate use of OOP - is one way that the complexity of the problem as defined at Analysis can be confirmed. A technical solution that has some embedded SQL or that creates objects at runtime, for example, does not in itself make the problem being solved complex. However, if at the Analysis stage doubt is present as to the appropriate level of complexity, then evidence from Design and Technical Solution stages should be used. A 'best fit' method of assessment as opposed to a 'tick box' approach is required during assessment.

High scoring candidates used advanced features of the programming language appropriately. It is helpful if the **complex** sections of the code written by the candidate are highlighted by the candidate or the Local Assessor. This confirms that the candidate understands what they have done.

The project is not made complex just because a candidate has used Java, C++ or Python for a fully coded data processing project involving minimal processing of a sample database. Similarly, writing an iPhone application using an application generator and incorporating published library code is not 'Complex'; it is the degree of technical competence of the candidate-written code and the creativity involved that is important.
A number of centres again allowed candidates to use the data source configuration wizard of VB.Net or the Express editions of the language to fill a dataset with data ie they used the TableAdapter feature. This is effectively a package-generated code to establish database connectivity and does **not** confirm complexity of the solution. There are other more appropriate non-wizard methods for database connectivity that should be used to gain high marks.

Only some Local Assessors made clear statements on the Project Log Sheets or other centre-generated documentation such as, 'A fully working robust system,' 'most processing objectives achieved,' etc. Those that did significantly assisted the moderation process.

Many of the candidates who used a programming language produced well structured listings. High scoring candidates used advanced features appropriately. It is helpful if such code actually written by the candidate is highlighted by the Local Assessor and, in particular, the use of advanced features such as user defined data/record structures need to be clearly indicated.

There is no longer a specific reference to parameter passing in the specification; this is assumed as demonstrating high levels of technical competence where parameter passing is appropriate. Solutions are not Complex simply because parameter passing has been used although this was used as the justification for perceived level by at least one centre.

Some candidates achieved a high mark in this section and went on to gain a high mark overall, but others seemed to spend the majority of their time on their coding to the neglect of the other sections that account for the remaining 55 of the 75 marks available.

If a part-packaged/part-coded approach has been adopted, it is important that validation is coded by the candidate using the programming language and that SQL queries are fully embedded within the code as discussed in the Teachers' Standardising Meetings. Similarly, the reports also need to be programmed, not generated by wizards. A small minority of candidates again simply customised a database package with minimal amount of self-written code. This is not in keeping with the philosophy of the new specification. Creating a solution which links a spreadsheet package eg MS Excel with Delphi or VB still counts as a part-packaged/part-coded system and the appropriate assessment criteria need to be applied. In simple terms: 'Package' does not just mean a database.

The Computing specification does **not** require a separate ICT-style implementation guide for the Technical Solution. Evidence for the mark in this section comes from the Testing and System Maintenance sections and possibly the User Manual where screen captures may be the only evidence of the working system if other sections are not complete.

System testing

Testing is one of the few sections that is assessed without reference to the perceived level of complexity. Testing by many of the average and weak candidates was trivial. It is **not** appropriate to have multiple tests for login passwords and/or to prove that all the buttons navigate to other forms and/or print reports. It needs to be emphasised that what is required is proof that candidates' coding works to produce accurate results and meets its specification.

The higher scoring candidates showed evidence of carrying out a thorough test plan effectively by producing a table of expected results and screen captures to demonstrate that these results were achieved; the screen captures were well annotated and were cross referenced to the table entries; they were **neither** heavily cropped **nor** simply labelled with a figure number. Ideally they might be corroborated by the Local Assessor or end user. Perhaps there should be no more than two screen captures to an A4 page to make them visible. Shadow script employed on screens printed four to a page made the information virtually illegible in one instance.

Few candidates used boundary testing effectively. In some cases boundary data was non-existent because of the type of problem being solved, but these instances need to be clearly stated and justified. Similarly, erroneous testing may not be possible because of the use of slider bars etc in the implementation. Sampling is needed for this section to avoid coursework of potentially massive length. Some candidates incorrectly interpreted extreme testing as extremely high or extremely low erroneous data. The following definition may prove helpful: 'Boundary values are those that are just **inside**, **on** and just **outside** the range of allowed values'. In this case boundary testing maps onto extreme testing. High scoring candidates often clearly identified if the data was erroneous, extreme (boundary) or normal. In simple terms, candidates need to demonstrate that the really clever parts of their system worked correctly, not that they could trap an incorrect password, accept a correct password and change a password etc.

Maintenance

Many centres used 'Prettyprinter' type software to produce the code listings and most candidates had used appropriate, self-documenting identifier names for variables, constants, types and subprograms. Many candidates also annotated their code appropriately. Frequently the procedure and variable listings (including scope as appropriate) were completely absent. This would make maintenance of the system difficult. However, in a high scoring project, there may be too many procedures and variables to all be documented separately. Unless the code listings exemplified a high degree of self documentation to a degree that minimised the need for substantive variable and procedure lists, candidates were penalised. A list of the main procedures and important variables, with some explanation of their importance and role, should be sufficient, backed up by well written code. Useful information about the main variables should also be contained in the design data dictionary.

Some candidates who had programmed in Java often used the Javadoc tool to create some of the documentation for this section, but others stated that it generated a lot of paper and included just a sample.

Not all candidates produced algorithms or a suitable alternative and only a small number of high scoring candidates produced them at a level that would enable **their** systems to be effectively maintained. If the algorithms originally proposed in Design are omitted or have been changed during Implementation, then they need to be updated here. If they have not been changed, a clear reference back to the algorithms in the earlier Design section must be given. This is allowable.

Please do not include any package-generated code. This was especially relevant for those candidates who had used the data source configuration wizard of VB.Net or the Express editions of the language to fill a dataset with data. This is effectively package-generated code to establish database connectivity and does **not** confirm complexity of the solution.

A similar problem occurred with a centre whose candidates had all used PyQT to generate the HCI required by their Python applications. Moderators are not expected to trawl through the submitted code to decide what was written by the candidate and what was effectively wizard generated. The centre is expected to identify clearly all of the candidate-written code with a highlighter or similar. Another centre whose candidate had used Javax Swing to create the HCI for a Java application had done this.

User manual

Some candidates produced very good User Manual documentation. Many showed screen captures in the documentation, but too many of these were too heavily cropped and the use of shadow text made it impossible to read. This was particularly the case with error messages where the data causing the error needed to be clearly visible as well as the message. Many failed to convince that the system worked, as there was very little data to guide the user. Some candidates failed to provide a guide to the complete system as the subject specification requires.

A detailed table of contents for this section should be included.

Not all candidates provided appropriate installation instructions for the different types of user/administrator of their system, especially for Client-server systems.

Nevertheless, most candidates incorporated screen captures with appropriate explanations, but perhaps they should have considered the needs of their users in more depth. Some

candidates had provided their user manual for the user to assess during acceptance testing and this was signed off in some cases. This was again good practice.

Error messages linked to screen captures, trouble shooting and recovery procedures generally needed to be more extensive. Sometimes, there was very little data in the system when either testing it or writing the user manual.

Appraisal

Most candidates referred to the objectives set out in their Analysis section, but not all fully evaluated how these objectives had, or had not, been met. Some simply stated, 'Yes,' or 'No' against each objective. Assuming that candidates had produced a detailed list of SMART objectives in agreement with their end user in the Analysis section, there was easy scope for comparison and discussion of outcomes versus objectives.

There were only a few centres that had candidates with significant evidence of genuine, authenticated user feedback. This needs to take the form of a dated and signed letter on headed notepaper or a **genuine** e-mail from the end user. Local Assessors should authenticate user feedback by communicating with the end user and signing and dating this section of the report, especially where it is obvious that the candidate has just typed this into the report. The best feedback evidence had criticisms as well as praise for what had been achieved by the candidate so as to allow analysis and derivation of possible improvements etc. Some of the feedback presented by candidates was unconvincing to say the least. Even if genuine feedback had been presented, many candidates still failed to analyse it and then use it as the basis for discussing possible future developments/ improvements. Many possible extensions (if indeed present) appeared to be entirely candidate driven.

Quality of Written Communication

This was usually quite good, but a detailed Table of Contents was not always present at the front of the report. Most centres accurately assessed this criterion, with the general view being that acceptable use of English, a well-structured report divided into the sections detailed in the specification, and the appropriate use of word-processing facilities could score the full three marks.

It is again disappointing that many candidates failed to use a word-processor appropriately; many project reports were missing such basics as headers, footers and word-processor-generated page numbers and a table of contents. Page numbering was hand-written in some cases. It would be useful if candidates were able to reprint the column headings on subsequent pages of the test plan.
It is not really appropriate to award full marks for this section when candidates put lots of important components of the documentation in appendices, especially if these were not accurately referenced in the sections where they should have been situated. This was exacerbated if there was no overall Table of Contents at the front of the project or a fully completed Project Log Sheet. Minor spelling and grammatical mistakes that did not detract from the meaning were not penalised this year.

Grade boundaries and cumulative percentage grades are available on the **Results Statistics** page of the AQA Website.

UMS conversion calculator www.aqa.org.uk/umsconversion