# A level
# Computer Science

Non-Exam Assessment
Report on the Examination

7517/C
June 2017

Version: v1.0

Further copies of this Report are available from aqa.org.uk

## Introduction

Schools and colleges that entered students in 2017 should read this report in conjunction with the specific feedback sent to the Examinations Officer on the publication of results. 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 NEA Guidance Booklet which is available on AQA's website. Exemplar and standardisation materials are available on the e-AQA website secure key material area. It was apparent that some centres did not have a real grasp of the new NEA even though material had been made available online and through 'Focus on the NEA' and 'teacher standardisation' face to face meetings that were run during the academic year.

A key part of the NEA is consideration as to whether a project is of 'A-level standard'. There were a collection of centres that submitted work marked as 'A-level standard' that, upon moderation, were clearly not at this standard. To assess a project as being of 'A-level standard' the first key place to look is in the analysis section. The key requirements of a user and the objectives set by the student should show clear potential for the technical solution to include skills that involve the complexity required even if a student does not meet these objectives in their final code. It was clear that some students had not set objectives that were challenging enough to be considered at the correct standard and this then affects the marking of the whole project as written in the specification. Students should be encouraged to make sure that the objectives they set initially for their project are challenging. If the student does not meet these objectives they should not be removed from the analysis section but rather be considered during the evaluation part of the project.

AQA provides NEA advisors who can be contacted over the academic year and are happy to discuss issues such as 'A-level standard'. Centres should be aware, however, that many ideas can start off with the potential of being at the correct standard but this should be finally assessed when marking the submitted analysis section.

## Administration

Many centres completed all the administration requirements fully and submitted excellent assessment comments with the projects which are very helpful to the moderation process. Some centres used their own assessment forms whilst others used the AQA Project Log. It is very helpful to the moderator if the administration has been completed correctly.

Centres are advised that the sample should be sent by 1st class post as required in the instructions to examinations officers from AQA. Sending by a courier requiring a signature can delay the sample being received by the moderator.

Samples are packaged up in a variety of ways but the simplest is to use a treasury tag to attach the CRF, Project Log and documentation all together. The use of bulky ring binders or individual wallets is not required and having loose sheets does not help the moderation process. It was also noted that some centres submitted electronic samples on USB or optical media. This is not currently an accepted method of submission and moderators are instructed not to open electronic submissions but to request a hard copy sample from the centre.

The most frequent errors with the administration were:

- Failure to submit a Centre Declaration Sheet with the sample sent to the moderator
- Failure to add up section marks correctly on the Student Record Form (CRF)
- Failure to check that the total mark on the CRF is entered correctly into the electronic submissions system

- Failure to ensure student authentication signatures are on the CRFs
- Failure to submit any supporting evidence of the assessment apart from the individual marks on the CRF

**General**

There was a wide variety of projects submitted from centres showing a good deal of initiative and exploration from students. Centres should be careful, however, when a group of students all complete very similar projects and use a group of similar algorithms to claim complexity. It is important for a student to consider the requirements of a project and not, for example, add in a login system when it is not necessary.

A lot of students attempted projects based around a quiz element. The majority were simple multiple-choice quizzes which were occasionally marked leniently by centres. If a student is to attempt a multiple-choice quiz then to score highly the system must have some inherent complexity built in. Perhaps the questions adapt to previous answers as the system generates the questions between certain limits. Perhaps the system offers a detailed way of tracking each quiz attempted so questions where students are struggling can be identified. To have a quiz system where the questions are hard-coded into the system and therefore can't be adapted or added to is also unlikely to score highly.

Students using VB and Access need to clearly identify in their technical solution what has been generated by the system and the algorithms written by themselves. There were other situations where students should be encouraged to identify between generated code and written code. An example of this is when a student uses a GUI creator and this was seen with Python, Java and Swift to name a few languages.

It was common to see grid-type controls on forms or reports showing tables of data with no real consideration as to the value of the data being presented or what it would look like with a large amount of data. So, for example, if a system is being set up for 200 students from a school and there is a grid to display all of these records some consideration should be placed into paging of the data or a means of filtering. Or, for example, a system to record attendance of pupils at a centre with no way of filtering will soon become very cluttered with just a simple 'SELECT * FROM register'. A system that will not be able to cope with the amount of data indicated in the initial requirements of the system cannot be considered complete when assessing the project.

Another popular area for projects was identifying a system that would 'teach' or help students 'revise a topic'. In the requirements set there would be ideas such as 'showing steps' or 'calculations should be made clear'. When considering the completeness mark this requirement needs to be considered as a group of projects were marked fully complete when a student using it would not come away having been 'taught' or having 'revised a topic'. So, for example, a system looking at A* path following might quickly draw the solution onto a maze but not allow the students to step through the solution or see how it was produced. If the requirement was for a student to be able to use the system to understand A* then this system is not complete.

**Analysis**

There is encouragement that a student should gather details for the project from users via a dialogue of some form. Some of the interviews seen were very detailed and clearly gained relevant information for development of the project. Unfortunately, it was also common to see very short interviews which gathered no real requirements for the project to be assessed highly by centres. Students should be encouraged that in the interview it would be beneficial to ask probing questions

to find out the real requirements of the user(s) and not just the kind of colours to be used or whether they like playing games.

The analysis should contain a list of the objectives set by the student for their technical solution. It was pleasing to see many students provide a detailed list of objectives that indicated both the requirements to be met and the complexity that this might involve. Students who submitted vague and brief objectives would struggle to pick up high marks in the analysis section and it would also be common for the rest of the project to suffer slightly. Weak objectives also make awarding the completeness mark hard as consideration must also be placed into what an A-level student would be expected to achieve.

The analysis section is to contain some modelling of the proposed system and it was pleasing to see students complete this in a variety of ways. Those projects that needed data processing usually included some discussion of the data required and DFD or ER diagrams. Students looking to produce a game sometimes struggled with the modelling section and also left the reader not understanding what their idea actually was. Students completing gaming projects could consider sketching out some ideas for the game and discussing the game flow as part of their modelling section.

**Documented Design**

It was pleasing to see good students carefully structure out the design of their technical solution. Effective use of diagrams to provide an overview of the whole system, key data requirements being identified and explained along with a breakdown of the complex parts leading to pseudo-code and/or code snippets would lead to a high mark.

It was also common, however, to see a more random attempt at the design documentation including just pasting code across with no detail as to the design process or how it would link into the main system. So, for example, just providing stock algorithms for merge sort and binary search does not help the reader understand the design of the system.

Having a section titled ‘sample of SQL queries’ is not very beneficial in providing a reader an understanding as to how the system will work. Students would do better to design out a particular form/page and then discuss the algorithms required for that part of the system including the SQL queries to be used for that part.

Students should be encouraged to think about the data to be used by the system. In a quiz system, for example, it would be beneficial to provide examples of the kind of question(s) to be asked. For a simulation it would be good to see how the formulas are to be used alongside, for example, a sketch of the trajectory of the projectile being modelled. For a game a student could sketch out the grid or level and talk through, for example, the movement of any enemies. It was common to see algorithms appear without the reader having any real understanding as to how these fitted into the system and a few sketches or examples of the data to be used would help. So, for example, one student produced an excellent Sudoku solver which had some complex pseudo-code in the design section. This code was hard to understand but a few sketches of particular board layouts showing how the individual functions would perform would have really helped.

It was common to see many students make use of well-known algorithms such as the merge sort. Just providing the pseudo code for this algorithm is not going to help their documented design mark. If the student talks about how this algorithm is going to be used by the system and integrated then this is beginning to pick up some credit. If the only pseudo-code or algorithm design a student attempts is based around merge sort, quick sort, binary search or other well-known algorithms

without any attempt at looking at other parts of the system then the student should not be scoring highly in the documented design section.

**Completeness**

This new section for the NEA was over-marked for many projects. When marking the completeness section consideration is given to the requirements of the project and the objectives that come from the analysis section.

When assessing completeness, centres were recognising that objectives had been met and using this to help guide the mark but sometimes a key requirement of the project had been missed and this needs to bring any mark down.

For example:

- When a student identifies a project that implies multi-user access across many machines but then a single local desktop application is developed it is hard to justify full marks in completeness. How would multiple users access and use the system without coming to this single machine?

- When a student identifies the need for a 'mobile' application but then produces a system that would not work on a mobile device.

- When the requirement of a system is to 'teach' or allow a topic to be revised. A student might meet all of their simpler objectives but this overall requirement, perhaps identified in the project background, still needs to affect the completeness mark.

Students who produced carefully written objectives that also included the essence of any requirements could go onto score highly in this section if they successfully produced their technical solution.

**Techniques Used**

Identifying technical skills was performed well by most centres. However, students need to think carefully about the requirements of their project and focus more on this than use well known algorithms just to add 'complexity'. Using a merge sort on a file that will only have 3 records is not really appropriate. In a similar way writing a merge sort to work on data brought back from an SQL query is perhaps not that effective when SQL has an ability to sort records.

Implementing a Caesar cipher to encrypt passwords is also not that effective, especially when most programming languages have access to encryption routines that can easily be used.

It was clear that some centres had provided a group of 'complex' algorithms for students to use. These would all appear in the design documentation as pseudo-code with no real information as to how they fitted into the requirements of the project and sometimes not even appear in the actual technical solution.
It was pleasing to see the number of projects making good use of OOP and using this effectively in their project. Students should be aware that they do not need to make a class for everything but should think carefully about the functionality that they require. A class for every separate question in a quiz probably demonstrates a slight lack of understanding rather than a good complex OOP model.

Databases continue to be involved in a lot of project and a variety of technologies were seen with a large group making use of MySQL and SQLite. Projects where time has been spent in the design section thinking about the correct data model could lead to good systems. To score highly, however, the project needed to work with the data and not just store it. Database projects that just used INSERT, UPDATE and very simple SELECT statements were not awarded Group A marks. Those, however, that made good use of cross-table queries or actually took the data returned from the database and then processed it to produce analysed results could score highly.

**Testing**

There was a wide range in the testing evidence presented by students. It was encouraging to see centres submit video evidence for projects as these could not only give clear evidence that objectives were met but also demonstrate the system being tested as a whole.

It is clear that a lot of students trust that their code is correct rather than taking time to work through a few sample scenarios in another way to prove their code is correct. It was pleasing to see, for example, a fractal program tested by generating fractals both on the student's system and also an online system to perform a comparison. Another student with an n-body particle system set up simulations to match certain well-known behaviours and demonstrated that their system created a similar model. Certain students took time to manually work through calculations to prove the algorithms were also performing correctly. It was common, however, for students just to take any output from a system to be correct and it would be appropriate for at least some tests to be proved a bit more thoroughly.

When using screenshots to evidence tests it would be beneficial for a student to also provide some explanation as to what it being shown. This might only be necessary for a sample but would be very appropriate when a test fails or shows a slight problem.

Good testing spent time to make sure that the system produced was robust. Students would put a reasonable amount of data into the system to see how it behaved. One student had a Raspberry Pi to control a boiler and ran their system over a fortnight to collect a good range of temperature values and also to demonstrate that it would follow weekly schedules. Other students, however, would test their system with 3 maths questions for a quiz or 10 students for one morning of an attendance recording system. These did not show how robust the system would be and meant top marks were not available.

**Evaluation**

Students with good objectives set in their analysis section could go on to score well in the evaluation section. They should, however, be careful to paint a fair evaluation of their project. If certain objectives have not been met very well then it is more beneficial to actually reflect on this rather than gloss it over. In the same way user feedback should also be honest as it was common for a project to receive glowing user feedback when it was obvious from the technical solution and testing that there were serious limitations to be commented upon.

## Mark Ranges and Award of Grades

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