A-LEVEL
COMPUTER SCIENCE

7517/C: NEA
Report on the Examination

7517
June 2019

Version: 1.0

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

Introduction

Centres that entered students in 2019 should read this report in conjunction with the specific feedback sent to them 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 material are available on the e-AQA website secure key material area and also through the TOLS system where there are 5 standardised projects. It is still apparent that a minority of centres did not have a real grasp of the new NEA and they should make sure that they refer to the resources highlighted above and make contact with their NEA advisor.

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 for A-level 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 provide 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. Comments that describe how a student has met assessment points and provide page numbers to support this are really useful. A comment such as 'shows a lot of Group A skills' does not really provide the detail that a moderator requires and it would be really helpful to identify items such as 'complex use of OOP with good use of inheritance (page X) and composition (page Y)'.

Some centres used their own assessment forms whilst others used the AQA Project Log. It is more straightforward for a moderator to agree with centre marks if the administration has been completed correctly and centres are reminded that they are required to provide evidence to support their marking when submitting the sample.

Centres are advised that the sample should be sent by first 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 hole punch the top left corner and use a treasury tag to attach the Candidate Record Form (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 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

As reported last year there continue to be a wide variety of projects submitted from centres showing a good deal of initiative and exploration from students. There were more hardware based projects seen this year along with students attempting to make use of neural networks and other kinds of machine learning. One project of note developed a complex system using genetic algorithms which learnt how to control a computer agent in a game.

Again this year we saw centres with students submitting very similar work and this needs to be considered when students present their proposal at the start of the project. Centres should also be careful about providing structures for groups of students to use as this then looks like provided resources rather than student work. For example, this could be encouragement for all to use a merge sort when this might not be required or to have sections in their documentation which are not really appropriate for a certain project.

A lot of students continue to attempt projects based around a quiz element. The majority still end up as simple multiple-choice quizzes which are then often marked generously by centres. The NEA Guidance booklet provides guidance as to how a quiz can be made suitable for an A-level project. A lot of students also attempt to complete a ‘revision’ based project and they should be careful to make sure that they evaluate against the effectiveness of their project to actually support revision as in many cases this was deemed to not actually be that successful.

Students presenting database projects, of which there are many, can end up scoring highly when these are managed well. Students who just allow data to be added to a table and then provide a few ways of selecting data from this table, or tables, will struggle to score high marks as there is now a required focus on the processing of the data. There are also cases where centres assess simple SQL queries as being Group A for skills when they are actually Group B and this can end up to the level being assessed wrongly for this part of the NEA. This year there were a selection of database projects that were considered to be borderline not A-level standard – the topics tended to be video loaning system, car hire system, library loaning system with a real lack of any real processing with a main focus on login and getting data into the system – centres should be encouraging any student with a database project to be identifying any processing clearly in the analysis section so that advice can be sought as to whether it is A-level standard.

Whilst the below was stated last year it is repeated this year again as it was common to see across centres. Centres are encouraged to make sure that they challenge students to explain their code and to check that design sections match up with what is in the technical solution:

Students using well known frameworks, for example Django, pygame or Unity, must make sure that they do not just follow a tutorial to complete their NEA project. It is evident that certain students do just follow tutorials and this is not recognised/referenced by the student or the centre. Whilst it might be appropriate for a student to make use of an available library or algorithm it is not the appropriate for the focus of the NEA to be a copied tutorial or to borrow a project off a site such as GitHub. Students who do make use of external code, be this a well-known library or just a small project on GitHub, should make this very clear in their design section and when printing off the final solution. There is ample opportunity for a student to discuss what external code might provide for the project, how it will be implemented into their code and provide examples of data coming in and out in their design section. It appeared that sometimes centres were awarding technical skill and completeness marks for sections of the project that were not written by the student themselves. There were also examples of the whole technical solution coming from a YouTube tutorial, online courses or GitHub repositories. Centres are reminded that they should be authenticating that (to the best of their knowledge) the work produced is solely that of each student.

Analysis

A lot of centres are still generous in awarding marks for the analysis section. The top level is looking for a 'fully or nearly fully scoped analysis, presented in a way that a third party can understand'. A lot of students are not providing enough clear detail so that the reader can understand what the problem is about and the objectives that are looking to be met. Scoping can, for example, be providing example questions and answers for a quiz – it was common to see, for example, a spelling test introduced but then no scoping of the kind of words that might be asked and how the difficulty of a word might increase. It is pleasing to see students complete research on topics such as neural networking but there is still a need to scope this into how it will be used in a particular project. A few students just provided copied content about a variety of topics but then did not bring these ideas together into how they would be relevant for their project. The modelling part of the analysis section continues to be often missing or not really relevant for the project type. Whilst a DFD might be suitable for a database project it would be more beneficial for a game to provide a storyline and a few sketches of what a level might look like. What goes in the modelling should allow the reader to have a feel as to where the project is going. It was still common for a project based around the concept of a game to fail to provide enough detail as to how the game would play or be structured.

It was pleasing to see centres respond to the comment in last year's report that interviews of potential users were very superficial and not probing enough. A good strategy is to provide the interview responses in a very structured manner with each question asked, followed by a summary of the responses and then any specific objectives/requirements to be drawn from this question and response.

The quality of the objectives set by students varies in quality. A good idea is to look at the objectives on their own and consider whether, from this list, you get a real sense of what the project is about and some of the complexity that will be required in the solution. Students sometimes introduced complex ideas in the research part of their analysis but then failed to mention anything about this in their objectives.

The analysis section is critical for the project and centres are encouraged to seek advice from their NEA advisor whilst analysis sections are being written.

Documented Design

The documented design is an important section for the student to provide evidence of their understanding as to how they will implement their project. It was common, however, to see poorly structured design sections and this part of the documentation marked generously by centres.

It is most beneficial if the design section starts with an overview. Whilst this might contain a variety of diagrams students should take time to explain what these diagrams are showing. The aim of the overview section is to show how the student has broken down their solution. This is also a good opportunity to discuss any framework that is being used and to discuss how this affects their design.

It was common for students not to design the main complex part of their project and this will limit the marks available. It is not beneficial for a student to heavily design the login section when this is only a minor part of any project. A sketch of the login page along with some brief details about how it will work will suffice which will then give time for the student to focus on the key components.

Whilst it might be appropriate for a student to give a brief introduction to a particular data structure it is considered that it is more important to spend time on how that structure will be used. So, for example, the theory of linked lists might be relevant but the student must then indicate how, where and even why a linked list is going to be used in their solution. Even better if they provide some example data for their project and show how this might be manipulated by algorithms utilising this data structure. So, for example, a chess engine will require the ability to store a representation of a board so it would be beneficial to sketch out a board position and then show how this would be represented in the proposed data structure. Then if there is a min-max algorithm running, perhaps identify how it would evaluate that board and show some of the steps that would be performed. In, for example, a physics SUVAT simulation the student might be expected to sketch out what this might look like but then annotate with examples of current variable values and how these might be changing (and the functions needed to change these and keep the animation running).

It was common for certain centres to include pages of variable names in the design section. For some students this could amount to over 30 pages of variables. This is not really what the design section is for and students would be better served spending more time in trying to explain how the project will work and perhaps identify key data structures and variables, with example usage, but a listing of them all from the solution is not really required.

User Interface sketches are very useful in providing an idea as to how the project will look, especially if they are annotated and with a discussion underneath about how actions are performed. It would also be beneficial to provide example data on these sketches rather than just blank areas.

Students should be encouraged not to focus on providing the pseudo-code and descriptions of well-known algorithms such as merge sort but focus on explaining how these might be used by the project. It would be better for a student to focus on the parts of the project that will require designing. So, for example, a moderated quiz project design might have contained a section on merge sort but not have covered anything about how an answer to a question would be assessed to be correct or not. It can be assumed that the reader of the design section understands merge

sort but would need to know the details such as question generation, display, assessment and tracking.

It is common for centres to arrange this section using headings taken from the specification when this does not really present the design in an appropriate way. The element 'sample of SQL queries' in the specification implies that in the design section there should be samples of SQL queries but not a section with this title. Students would do better to organise their design sections according to the main 'chunks' of their project. So, for example, if the system is to schedule parents' evening slots it would be good to have a sketch of this for UI work, followed with any data structure usage, some pseudocode for how that part of the project will work and the SQL queries required. Some example query data would also be very useful to allow a reader, and potential programmer, to get a real feel as to how that section of the project should work. Then the student should move on to the next 'chunk' of the project.

Completeness/Techniques

It is pleasing to see centres consider more carefully how they award marks for this section. There was a greater usage of the spread of marks and also more detailed comments in project logs as to how these marks had been awarded. A few centres continued to award full marks for the majority of projects sampled and this was not justified when considering the variety of completeness in the projects submitted.

A project that only meets objectives and skills that are around high GCSE level should only be entering level 2 at best. Whilst the student might be meeting their objectives consideration must go into the requirements of the actual project and potential users and also would be expected of an A-level student.

There are projects on e-AQA and TOLS where the student has met their objectives but the completeness mark has remained in level 2 and centres are encouraged to look at these. A common place for this to happen is with quiz systems and 'teaching'/'revision' systems. If, for example, a student introduces the idea of A-level students using a program to understand Djikstra's algorithm as part of their introduction to the analysis stage then this must be considered when marking the completeness. The student might have met all of their objectives in terms of working out Djikstra's algorithm and displaying it but if the system does not help a user understand the algorithm then it can't be marked as fully complete.

When marking the techniques it is beneficial for either the centre or student to clearly identify the skills used in their coding. A student can do this by placing a cover sheet in front of their code with a list of key subroutines, skills used and page numbers where Group A/B skills can be identified.

Students must also clearly identify code they have written as distinct from code which is borrowed, comes from a library, textbook or is auto-generated by a framework. Any borrowed, library or similar code should have already been introduced and referenced in the design section.

There were more students submitting code that was hard for a moderator to read this year. It is not useful to have screenshots of code when this comes from a 'dark mode' setting as the code is hard to read from the dark background. There were also a group of students assuming that very small fonts could be easily read by a moderator. Centres and/or students should also make sure that the whole code of the solution is available to the moderator as for a few projects it was clear that certain sections were missing and this complicated the awarding of the completeness and techniques marks.

Testing

It is pleasing to see more centres submit evidence of testing using video. For certain projects this can really help provide evidence that objectives have been met and actually be less time consuming for the student compared to providing a large amount of screenshots. Students should be encouraged not to focus on too much testing of the UI of their project and aspects such as login and registration. When a project has a main focus on some sort of processing this should be where the majority of the testing is focused. It was evident this year that students who provided a commentary on their video provided evidence that was easier to follow and understand.

It is also pleasing to see students begin to realise the importance of performing some manual checks for the processing of their solution. So when a depth-first search, for example, is performed this is shown to be correct by also manually performing it. Whilst it is not appropriate to check every action of the solution manually it is appropriate to perform this once or twice to show that their code is performing as expected.

Similarly with database projects it is pleasing to see students realise that it is appropriate to also show the contents of the tables whilst performing some tests so that it can be evidenced that the program and the database are interacting correctly.

It was nice to see a project where a multi-player networked game was created and the testing clearly showed the server element running on one computer and clients running on other devices in the computer lab.

If using screenshot evidence students should be encouraged to annotate their screenshots to explain what is going on – this is important for tests that are failing. It is also useful for students to try and include evidence of the data being stored by showing, for example, evidence from the database itself or using breakpoints to produce values from an IDE.

There are students who focus more on testing the objectives and perhaps forget to also test the system as a whole. So, for a quiz, work through a whole test to show the functionality of the system ending with a score and perhaps an entry into relevant database tables and updating of tracking data. Students who submit video testing can often show a whole system test across the video and then use timestamping to indicate where specific tests have also been performed.

This year it was also common to see more unit testing being performed with some of this being automated using testing systems. Students who complete unit testing in the design section, which is perfectly acceptable and to be encouraged, should remember to refer to this in their testing section of the documentation.

If students are submitting video evidence for their project, which is to be encouraged, then a way of accessing this video for the moderator should be very clear. Students should be encouraged to use a large, clear font when placing any link into documentation and look to remove any hyperlink (underlining). A few centres used URL shorteners and this helped take some complex links and made them simpler (especially for shared Google folders). A few centres either setup a YouTube channel for all of their projects or placed all videos into a shared folder and provided clear details of this with an enclosed sheet in the sample – this was very helpful.

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 over it. 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. To gain the top level a student is required to consider some improvements in detail and whilst not necessarily producing final code it would be appropriate for a student to take at least one idea and consider the changes and challenges this would present if it were to be implemented.

It was evident this year to see students not reflecting on the project as a whole. It would be beneficial for an evaluation to start with an honest and fair reflection on their solution compared to the ideas set out in the Analysis section. To gain a level 4 mark a student should have reflected on the whole project as well as the individual objectives.

This year, a few centres had students evaluating their objectives by supplying either a screenshot from the solution running or copying code from the implementation. This is not really evaluating an objective. Whilst it might show an objective being completed we are expecting a student to think about the objective and consider how, or indeed if, they met this objective and whether, upon reflection, they feel this has been done well or perhaps could be improved.

Mark Ranges and Award of Grades

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