A-LEVEL
COMPUTER SCIENCE

7517/C Non-Exam Assessment
Report on the Examination

7517/C
June 2023

Version: 1.0

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

GENERAL FEEDBACK

It was again pleasing to see the wide variety of projects being submitted by centres and the number of different environments being used. From various simulations, games of many genres, business database applications to machine learning it is clear that students are taking on a wide variety of computer science concepts and centres are commended for allowing this to happen.

There are still a lot of students building login systems for projects and applications that really do not require the idea of users and logins. Students would be better rewarded by spending more time thinking about the complexity of their main project area – for example a 'polynomial calculator app' that had users and logins for no particular reason – there was no data held against a user login and it would only run on one personal device. In a similar way there are still database projects where a student will then merge sort a returned result from an SQL query when on the A-level specification we do cover the 'order by' part of a query and this would seem more appropriate. Students would be expected to use standard methods for implementation.

Many centres are administering the project effectively over the long period of time normally offered to students. The use of project logs continues to help provide the marking rationale and really helps during the moderation period. There is a continual movement towards testing evidence being provided via video evidence. Those that commentate over the video tend to produce something that is more useful.

There continues to remain a concern over the authenticity of student implementations and centres are encouraged, whilst supervising their students, to probe understanding of code produced by students and to ensure that any links to tutorials or sources used are clearly referenced in either the documentation or on the CRF form. It was pleasing again, this year, to see some centres clearly responding to this concern with marks being reduced in the skills and completeness sections where appropriate and occasionally setting both these marks to zero when there was no evidence of student work. Where students have clearly written the documentation themselves (analysis, design, testing, evaluation) it does seem appropriate to award marks to these sections but zero completeness and skills when the project code clearly comes from a tutorial or perhaps a GitHub repository with minimal changes from the student.

ANALYSIS

Many students complete a good analysis to start off their project and it is clear that centres also have conversations around the setting of the objectives with many making use of exemplar projects and their NEA adviser.

Some students do fail to scope out their project 'fully' and therefore leave questions in the mind of the reader. How many people will the system work for? What kind of items does the shop sell? What is exactly covered by KS3 maths angles topic? What equations might be needed to understand the movement of air around an obstacle? What does the school layout actually look like if we are aiming to build a shortest path app?

It was pleasing to see a group of students use prototyping for their initial modelling, whilst others looked at how APIs could be used showing example requests and responses.
When completing the analysis section imagery can be really useful in helping the reader have a better feel for the project, for example images from researching current or similar applications, from textbooks/specifications/menus of a restaurant/documentation to help scope out the project and from annotated sketches in the initial modelling section.

In the requirements gathering section the stronger students ask probing questions relevant to the project area to gain insights that will help set objectives and scope out the project. Weaker students tend to ask more general questions such as 'do you like playing games?' and 'what kind of games do you like to play?' and fail to then ask questions that will actually generate relevant requirements.

For some students objectives are more about 'how' something will be created rather than 'what' the project will complete. So 'use an SQL database' is perhaps not a good objective but 'implement customer, stock and order tables using an SQL database' would be considered better.

The initial modelling is a part of the analysis section that many students still miss. This could be annotated level designs for a game, a sketch of a projectile moving linking into the equations necessary to draw it, an initial analysis of the fields and tables that might be required for a database, example calls and responses from an API. It was also pleasing to see a few centres encourage students to complete some prototyping.

DOCUMENTED DESIGN

The documented design, as with the analysis, tend to be the areas where marks are changed during moderation. The aim of the design is that there is detail present so that a competent programmer could take the design and produce an implementation. There is no expectation to design out everything but a student should be encouraged to design out some of the more complex areas of the project. There are still student designing out parts of a project such as login and passwords and then do not cover the main part of their project which might be, for example, asking questions and marking them and then producing reports for the user.

Many centres have moved to setting out design in 'chunks' referring to key areas of the project. In a particular chunk there may be UI design, data structure discussions, any relevant database queries and algorithm design through structured English / flowcharts or pseudocode. Centres that still encourage a 'sample SQL queries' section tend to have designs that do not hang together well as a page of sample SQL does not help the reader fit it back into the relevant part of the project. Chunking designs really does help these sections hang together well.

It was common this year, especially with games, to find little information about how the game would output to the screen. If a student included a couple of sketches from a level/scene and then linked in how the internal data and algorithms were being used to draw this on the screen they could provide a very clear design. For a chess project, if they provided a sketch of the board and then showed how the board data would be stored internally again the design became much clearer. For a projectile simulation, if they sketched out a scene and then provided the current internal data and using the formulæ / algorithms they could then show how the next scene would be drawn.

Students should be encouraged to link diagrams into documentation with some text of explanation. There were also more diagrams this year that were not readable due to the size of the text. Student should check their documentation when printed to make sure that all parts are readable.

SKILLS and COMPLETENESS

Most centres assess these sections well. Centres must be mindful of code that is not written by the student and it was pleasing to see a group of students clearly highlight code that was not written by themselves.

When assessing completeness, the effectiveness of the system must be assessed. This year we saw a selection of 'London Underground Route Planners'. Some students setup a graph and used random data with minimal stations, whereas one student clearly travelled the London Underground and gathered timings for each stop. Groups of students also made use of the Transport for London API to gather 'live' data. Where a student just brings in Dijkstra and applies it to random data this can't be considered complete as a solution – this is a student seeing Dijkstra in the skill list and then just implementing a simple example rather than linking it to the main aim of the project.
For skills to be really credited they need to be part of the solution and embedded into it and not just core A-level algorithms thrown in with no attachment to the project.

When providing the technical solution (code) students should be encouraged not to screenshot from 'dark mode' output as this becomes very hard to read when printed. They should also double check the size of the code when printed to make sure it is readable.

TESTING

There are many students who carefully test the implementations. Where they make use of a clear test plan, modular testing with screenshots and then provide a video demonstrating the whole project against the objectives it can be a clear way to secure the top level marks.

Videos with commentary provide much clearer evidence than does a soundless video.

Students should think about how to show that everything is working – so those that show a database window open with the application can clearly demonstrate that their system is manipulating the database.

EVALUATION

Evaluations tend to be completed well. Some students will just 'assess an objective' rather than 'evaluate an objective' with a Yes/No checkbox. It is important for students to reflect honestly on the project as a whole and then move onto considering each objective. Those that score well clearly identify issues in the project and discuss reasons for these issues and how they could be improved in the future.

User feedback is often found in the evaluation section with those who ask users to complete a checklist with comments against the objectives performing well. Sometimes user feedback is more general and often not that reflective on the actual project itself commenting on aspects such as the look rather than how tstudehe project actually performs.

Mark Ranges and Award of Grades

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