Course report 2023

Advanced Higher Computing Science

This report provides information on candidates' performance. Teachers, lecturers and assessors may find it useful when preparing candidates for future assessment. The report is intended to be constructive and informative, and to promote better understanding. You should read the report in conjunction with the published assessment documents and marking instructions.

The statistics in the report were compiled before any appeals were completed.

Grade boundary and statistical information

Statistical information: update on courses

Number of resulted entries in 2022: 695

Number of resulted entries in 2023: 665

Statistical information: performance of candidates

Distribution of course awards including minimum mark to achieve each grade

A	Number of candidates	210	Percentage	31.6	Cumulative percentage	31.6	Minimum mark required	93
B	Number of candidates	153	Percentage	23	Cumulative percentage	54.6	Minimum mark required	79
C	Number of candidates	122	Percentage	18.3	Cumulative percentage	72.9	Minimum mark required	66
D	Number of candidates	93	Percentage	14	Cumulative percentage	86.9	Minimum mark required	52
No award	Number of candidates	87	Percentage	13.1	Cumulative percentage	100	Minimum mark required	N/A

Please note that rounding has not been applied to these statistics.

You can read the general commentary on grade boundaries in the appendix.

In this report:

- ‘most’ means greater than 70%
- ‘many’ means 50% to 69%
- ‘some’ means 25% to 49%
- ‘a few’ means less than 25%

You can find more statistical reports on the statistics and information page of SQA’s website.

Section 1: comments on the assessment

Question paper

The question paper largely performed as expected. Feedback from the marking team, teachers and lecturers indicated that it was positively received by centres and was fair and accessible for candidates. The majority of candidates understood what was required and completed the mandatory and chosen optional sections in the allocated time.

The 'C'-level questions at the start of each section helped candidates to focus on the specialist content of each section, before tackling the more demanding problem-solving questions that followed. Although the levels of uptake for the optional sections were similar (54% attempted the 'Database design and development' section, 46% attempted the 'Web design and development' section), the overall performance of candidates opting to attempt questions in Section 2, 'Database design and development', was better than the performance of those candidates who attempted questions in Section 3, 'Web design and development'. This was observed in the questions common to each section as well as the specialist questions.

Project

Revisions to the coursework assessment task published in September 2022 aimed to clarify the project requirements, and what evidence candidates were expected to provide for each stage of the development of their solution. Overall, the revised support and advice was well received, and it helped candidates to focus on the importance of the functional requirements of their chosen project, rather than getting side-tracked with the development of the user-interface of their solution. However, most candidates overlooked the need to comprehensively test all input validation routines. As a result, the application of the marking instructions led to fewer marks being awarded than intended, and an adjustment of 1 mark was made to the grade boundaries for C and A grades.

Section 2: comments on candidate performance

Areas that candidates performed well in

Question paper

Section 1: Software design and development

Question 1: Many candidates identified the correct dimensions of the 2D array, and it was pleasing to see that many more had attempted to indicate the intended data type than in previous years.

Question 2: Many candidates were able to explain why a single linked list was suitable in the situation described. Most candidates achieved full marks in part (b) by making appropriate use of terminology associated with linked lists such as ‘node’, ‘head’, ‘tail’ and ‘pointer’.

Question 3(a): Many candidates correctly identified that a database actor, needed to store appointment details, had been omitted from the UML use case diagram.

Question 3(b): Most candidates provided very good descriptions of the use made of the include and extend relationships by making appropriate reference to the use cases in the UML use case diagram.

Question 4(a): Most candidates made good use of object-oriented terminology, such as ‘instantiate’, ‘object’ and ‘class’ to describe the effect of the code. The assignment of the parameter values in the code to the `nationality` and `username` instance variables was accurately described by many candidates, with some also able to describe the use made of the constructor method to assign a default value of 1 to the remaining variables.

Question 4(b): Most candidates received 1 mark for the correct use of instance variables in their calculation of the average power.

Question 4(c)(i): Most candidates made good use of the object-oriented term ‘array of objects’ to describe the effect of the code, and many were able to describe the assignment of the `newPlayer` object as the first element in the array of objects.

Question 4(c)(ii): Many candidates provided the correct code needed to complete line 2003 and 2010 of the bubble sort algorithm.

Question 4(d)(i): Most candidates were able to provide an accurate description of inheritance by using the object-oriented terms ‘superclass’, ‘subclass’, ‘methods’ and ‘properties’, with many candidates making reference to the UML class diagram as required.

Question 4(d)(ii): Most candidates were able to identify the use of overriding, an essential component of polymorphism.

Question 5: Most candidates attempted all parts of Question 5, resulting in most candidates being able to access some marks for each question.

Question 5(a)(i): Many candidates correctly identified the need to traverse the readings array and some were awarded 1 mark for identifying the need to check whether a local authority name already existed in the rainfall array.

Question 5(a)(iii): Many candidates correctly identified the need to traverse the rainfall array, and the need to keep track of the number of times an individual local authority appeared in the readings array.

Question 5(b): Most candidates gained marks for the binary search content of their solutions, with many also making use of the results of the binary search when displaying the data.

Section 2: Database design and development

Question 6(a): Many candidates correctly identified the weak `Rental` entity with the correct relationship participation between the `Accommodation` and `Rental` entities.

Question 6(b): Most candidates were able to identify and describe a potential problem with the primary key of the `Student` entity, with many going on to correctly suggest that the problem could be resolved with the use of a surrogate key.

Question 7(c): Most candidates gained 1 mark for correctly identifying the primary key of the `Order` entity, with many receiving a further mark for either identifying and referencing the foreign key, or for the correct use of `CHECK` with `IN` which was needed to implement the restricted choice for the `orderSent` attribute; some candidates were awarded all 3 marks.

Question 7(d): Most candidates correctly identified the use of the `BETWEEN` operator, with some also indicating the need to use the `NOT` operator.

Question 7(e): Most candidates were able to accurately describe the use made of the `HAVING` clause, with many able to describe the results generated by the subquery. Some candidates were also able to describe the use made of the `EXISTS` operator to check whether the subquery had returned any results.

Question 7(f): Most candidates received 1 mark for the correct use of `HAVING` to select products with fewer than five available, and some made appropriate use of `IN` to restrict the selection of product makes.

Section 3: Web design and development

Question 8: Most candidates gained 1 mark for an accurate description of the display size for images being 50% when the screen size was greater than 767 pixels.

Question 9(a): Most candidates provided an accurate description of the use made of the PHP `mysqli_num_rows()` function to ensure that query had returned results.

Question 10(c)(iii): Most candidates made appropriate use of each of the connection details provided in the question, with many demonstrating good knowledge of how those details are used with the PHP `mysqli_connect()` function. Only some candidates received a mark for the correct use of the `mysqli_query()` function, with many failing to make use of the `$query` variable provided in the question.

Areas that candidates found demanding

Question paper

Section 1: Software design and development

Question 4(b): While many candidates ignored the need to implement a method to perform the required calculation, some opted to implement a function that returned the calculated value to the main program, rather than using a procedure. Also, instead of assigning the results of the calculation to the encapsulated instance variable `power`, a few candidates introduced a local variable to store the result of the calculation.

Question 4(c)(ii): Most candidates did not make use of the `getPower()` method to access the encapsulated instance variable that was needed in the comparison at Line 2006. In addition, many candidates ignored the need to sort the league table details into descending order, which resulted in the use of an incorrect comparator in the comparison.

Question 5(a)(ii): Many candidates did not notice that the assignment to temp at Line 4 was an assignment of a single local authority name rather than the assignment of an entire record. As a result, the need to indicate the local field needed in the comparison at Line 6 and in the assignment at Line 10 was omitted from the responses of many candidates. In addition, some candidates demonstrated poor application of the insertion sort algorithm by comparing the temp value with the following value in the array rather than the previous one at Line 6, and by making incorrect use of temp at Line 10.

Section 2: Database design and development

Question 7(a): Many candidates did not identify the adaptive maintenance required to ensure that the website was compatible with the latest versions of web technologies.

Question 7(b): Although many candidates did receive 1 mark for a discussion related to one of the claims made, some did not read the wording of the question and focused instead on providing a very generic description of the use made of a Gantt chart in project planning.

Question 7(g): Most candidates were unable to identify the correct type of testing being undertaken. Although some candidates were able to accurately name and justify the use of end user testing in question 7(g)(ii), only a few were able to accurately name and justify the use of final testing in question 7(g)(i).

Section 3: Web design and development

Question 8: Although most candidates were able to provide an accurate description of the display size for images being 50% when the screen size was greater than 767 pixels, only some candidates were able to fully describe the effect of the CSS `max_width` property. The lack of precision in responses that ignored what size of image would be displayed on a screen width of exactly 767 pixels meant that only some candidates were awarded full marks.

Question 9(b)(i): Few candidates were able to identify the use of the PHP `mysqli_fetch_array()` function, with many simply repeating the `mysqli_num_rows()` function that had been used in part (a).

Question 9(b)(ii): Many candidates were unable to indicate the correct use of the HTML `<tr>`, `<td>`, `</tr>` and `</td>` elements that were needed to display the table data.

Question 10(a): Many candidates did not identify the adaptive maintenance required to ensure that the website was compatible with the latest versions of web technologies.

Question 10(b): Although many candidates did receive 1 mark for a discussion related to one of the claims made, some did not read the wording of the question and focused instead on providing a very generic description of the use made of a Gantt chart in project planning.

Question 10(c)(i): Many candidates did not make use of the HTML `SELECT` element to produce the drop-down list required.

Question 10(c)(ii): Most candidates did not spot the use of the session variable that was needed to share the selected computer type on Screen 1 with the code used to process Screen 3. Despite this, some candidates were able to identify the use made of either `$_GET` or `$_POST` to share the selected computer make on Screen 2 with the code used to process Screen 3, together with the need for the PHP `echo` statement to display this value on the screen.

Question 10(c)(iv): Although a few candidates were able to describe how the query results would be used to populate the value attributes of the HTML `option` element, most candidates were unable to describe any of the PHP that would be needed to display the drop-down menu for Screen 3.

Question 10(d): Most candidates were unable to identify the correct type of testing being undertaken. Although some candidates were able to accurately name and justify the use of end user testing in question 10(d)(ii), only a few were able to accurately name and justify the use of final testing in question 10(d)(i).

Areas that candidates performed well in or found demanding in the project

Stage 1: Analysis

Overall, candidates performed well in the Analysis stage of the project. UML use case diagrams were better than in previous years, with many candidates wisely creating the use case diagram early in the Analysis stage to help them to identify several of the underlying processes that became functional requirements of the solution. Although it is not necessary to do so, many candidates also made use of include and extend relationships to add a layer of refinement to several of the main use cases; for example using include to indicate the mandatory nature of input validation or display the final score in a quiz, and using extend to indicate the optional nature of viewing a web page on a mobile device, attempting a bonus quiz question or updating an email address. In addition, most candidates produced accurate project plans that listed relevant subtasks to be completed at each stage of the development, together with realistic timescales for the completion of those tasks.

Description of the problem

Although there were some excellent descriptions that outlined how all mandatory Advanced Higher concepts were going to be incorporated in the solution, the problem descriptions of many candidates lacked sufficient detail of the relevant Advanced Higher and integrative concepts for the type of project that they had selected.

Project plan for each stage

Many candidates received at least 1 mark for their project plans, but some did not indicate the resources that would be needed at the Implementation stage, with many who tackled a web project omitting to mention the need for a browser. In addition, some candidates

presented a list of subtasks for each stage that was very generic in nature and didn't take account of the specific nature of the development or the requirements already identified.

Stage 2: Design

In general, candidates who had a clear, well-defined set of requirements at the Analysis stage did well in the Design stage of their project development. Candidates who received good marks for their design had considered each of the identified requirements and evidenced them appropriately. Candidates who carefully considered the need to validate all inputs to their solution had thought about what type of validation would be necessary. They indicated a variety of strategies on their user interface designs, including range checks, a presence check, restricted choices, and length checks. In addition, those candidates also took time to indicate planned error messages and, in web projects, necessary redirects.

Design of Advanced Higher concepts and integration

Most candidates who tackled a software development project did not provide any indication of the intended structure of the 2D array and/or array of records that would form an essential part of the implemented solution. As a result of this, when presenting the design of the selected Advanced Higher algorithm, many candidates simply presented a generic algorithm that made no reference to the data structure that would be processed by the algorithm concerned. Most candidates did not provide a top-level design showing the intended data flow between sections of their solution, together with refinements of Advanced Higher algorithms, functional requirements and input validation.

Most candidates who opted to complete a web project did not provide a site navigation structure to show the planned pages of the site and indicate what navigation would be possible between those pages. The few structures that were submitted were often hierarchical in nature, omitting planned redirects or links between pages that were clearly visible in the wireframes that formed part of the user interface design evidence. In addition, many web candidates did not indicate the intended use of session variables in their solutions.

Some candidates who chose to implement a database, either as the focus of the project or as the integrative component, did not indicate the type of query (SELECT, INSERT, DELETE or UPDATE) in their query design. This lack of detail made it very difficult for markers to match the query design to the implemented code.

Regardless of the type of project selected, many candidates continue to present code, or reverse engineered code, rather than pseudocode as the design of their Advanced Higher concepts, integration and requirements.

User-interface design

Although most candidates received partial marks for this evidence, many of the user interfaces submitted lacked sufficient detail of the intended input validation, instead mentioning it only in vague, generic terms. In addition, some candidates did not indicate the underlying processes that would be performed when selections were made, menu options were clicked, or buttons were pressed.

Most candidates who chose a web project did not submit wireframes to indicate the effect that the mandatory media query would have on all relevant pages of the website.

Stage 3: Implementation

Most candidates received good marks for the implementation of the mandatory Advanced Higher concepts, implementation of the necessary integration and for the implemented user-interface.

Implemented Advanced Higher concepts

Most candidates who chose to implement a database, either as the main focus of their project or as the integrative component, did not indicate the initial contents of the table or tables and the data used to populate them before any queries were executed. In addition, many of these candidates did not submit evidence of the structure of the implemented table or tables to show that it matched the design indicated in the data dictionary.

Although most candidates who implemented an Advanced Higher search and/or sort algorithm did submit screenshots to show the results of how the algorithm performed, most did not submit the 'before' evidence of the data searched or the unsorted data, which was needed to show that the algorithm did in fact work correctly.

Description of new skills and/or knowledge researched and developed

Despite the need for new skills to be related to the functional requirements, the evidence submitted by many candidates for this stage was associated with enhancements to the user interface or was very superficial and trivial in nature.

Log of ongoing testing

Although many candidates did receive partial marks for this aspect of their solution, it was clear that much of this evidence had been added retrospectively and omitted many of the important Advanced Higher concepts, integrative components and functional requirements that were included in the working solution.

Stage 4: Testing

Overall, candidates who had a clear, well-defined set of requirements at the Analysis stage did well at the Testing stage of their development. Most candidates who had numbered their functional and end-user requirements benefitted greatly from this when they came to create a test plan for final testing, as it ensured that none of the requirements were overlooked. With a test plan in place, most candidates achieved good marks for performing each of the planned tests and providing appropriate screenshot evidence.

Test plans and results of testing

Most candidates received partial marks due to incomplete test plans that omitted one or more of the requirements listed at the Analysis stage. Many candidates also did not provide adequate descriptions of the individual test cases that they would use to test each of the requirements that had been listed. Moreover, most candidates did not provide any details of the test data values that were needed to test the mandatory input validation, with very few

mentioning the use of normal, extreme and exceptional values; fewer still made any reference to the expected output in the form of error messages.

It was clear from the evidence submitted by many candidates that they had a much better understanding than last year of the need to describe a persona who would be a typical end-user of the final solution. However, some candidates put so much time and effort into the creation of multiple imaginative personas that they either overlooked the need to list the test cases that would be tested by adopting the characteristics of the personas described, or failed to provide any description of the results of testing those test cases.

Stage 5: Evaluation

Some candidates received 1 mark for descriptions that referred to the robustness of their final solution by accurately referring to the input validation that had been added to the code, and the results of testing that highlighted any issues with this aspect of their solution.

Fitness for purpose

The evaluation of fitness for purpose submitted by many candidates lacked sufficient detail at Advanced Higher level. In some cases, descriptions were incomplete with one or more of the requirements being ignored, while in other cases the list of requirements was simply presented as a table with a column of ticks to indicate that the requirement had been achieved; in a few cases, solutions were said to be fully fit for purpose and ticks had been added to the table when, in fact, the Implementation and Testing evidence showed that there were errors in the solution.

Maintainability

Descriptions of maintainability submitted by many candidates lacked sufficient detail at Advanced Higher level and bore more resemblance to responses at National 5 level. Very few candidates focused on types of maintenance and considered features of their solution that would aid that maintenance.

Section 3: preparing candidates for future assessment

Teachers and lecturers should note that we are keeping the following modifications to assessment in the Advanced Higher Computing Science course in session 2023–24:

- We have removed computer systems.
- Candidates will have the option to complete either the 'Database design and development' or the 'Web design and development' section of the question paper. Questions in each option will require a degree of integration with the other section. This is set out in pages 15 and 16 in the updated course specification. This reflects what teachers and lecturers have told us they are already teaching to support integration in project work.
- The exam will remain reduced from 80 marks to 55 marks, with a shorter duration of 2 hours.

An updated Advanced Higher Computing Science Course Specification and a new specimen question paper can be found on the Advanced Higher Computing Science page of SQA's website.

Question paper

Most candidates attempted to indicate the intended data type for the 2D array in question 1. Teachers and lecturers should continue to encourage candidates to indicate the intended data type for any software design and development data structure they are asked to define or declare. Where responses are presented in languages that make use of dynamic data typing, candidates may wish to add internal commentary to their code to ensure that the intended data type is clear.

Teachers and lecturers should ensure that candidates are familiar with all three types of maintenance introduced at Advanced Higher level: adaptive, corrective and perfective. Candidates should understand the purpose of each type of maintenance and know when each type of maintenance would be necessary. They should also know who would be responsible for any development costs associated with the maintenance.

Teachers and lecturers should ensure that candidates are familiar with all types of testing expected in an Advanced Higher Computing Science question paper: component testing, end-user testing, final testing, integrative testing and usability testing based on prototypes that make use of the described personas. Candidates should understand at which stage in the development process each type of testing is used, who is involved in each type of testing (members of the development team or end users) and what role each participant plays.

Overall, candidates appeared to have coped well with the problem-solving questions that required them to design algorithms for unseen tasks and processes, such as those in questions 5(a)(i), 5(a)(iii) and 5(b). Teachers and lecturers should continue to encourage candidates to attempt these more challenging questions, as statistical evidence shows that most candidates receive partial marks for correct aspects of their design, even when those designs may be incomplete or do not provide a fully working solution.

Teachers and lecturers should continue to ensure that candidates are familiar with the required object-oriented terminology used to explain the operation and effect of code that is written in the SQA Reference Language. They should advise candidates to pay close attention to any UML class diagram in the question paper which lists the methods available to the main program code, and also pay close attention to any class code that may be presented. In their responses to question 4(b), where candidates were expected to code a method to simply calculate a player's power rating, it was clear that many candidates did not notice the `getPower()` method that was already available in the `Player` class. In addition, since the UML class diagram indicated that all properties in the class were private, any code used in the calculation must refer to relevant instance variables and, once calculated, the assignment must be made to the encapsulated `power` variable. Teachers and lecturers should ensure that candidates are aware of how overridden methods can be used to demonstrate polymorphism when they are applied to an array of superclass objects.

Centres with candidates attempting questions in Section 2 (Database design and development) should ensure that candidates clearly indicate the strong and weak entities in their entity-relationship diagrams. Centres should note that Appendix 3 of the course specification has been updated to use a double-edged rectangle to indicate the existence of a weak entity, with use of a single-edged rectangle indicating the presence of a strong entity. The use of O and | to indicate the optional or mandatory nature of both sides of the relationship between related entities is unchanged. This is how we will present entity-relationship diagrams in question papers, but we will still accept other approaches that candidates may take.

Candidates attempting questions in Section 3 (Web design and development) must be familiar with all of the PHP coding and mysqli functions required at Advanced Higher level. Candidates are expected to understand the purpose of each mysqli function and know when it would be used. To better prepare candidates to tackle unseen problem-solving questions that cover this content, teachers and lecturers should ensure that candidates have opportunities to read and explain unfamiliar PHP code, together with opportunities to select and apply appropriate mysqli functions to solve unfamiliar problems.

Centres preparing candidates for Section 3 should also ensure that candidates understand the use of session variables as a mechanism to enable data to persist across several pages of a website. The use of session variables is typically associated with user data, possibly gathered when logging into a site or, in the case of question 10(c)(ii), gathered from selections made by the user when navigating the site. Session variables let the server know that requests originate from the same user and allow the site to display user-specific data. Session variables are needed whenever it is necessary for data to persist across at least three separate pages; when data only needs to be shared across two pages, the use of `$_GET` or `$_POST` is sufficient.

Project

Coursework assessment task

Centres must ensure that they are using the correct version of the coursework assessment task. It provides candidates with good advice about the evidence they are expected to submit at each stage of the project. Unfortunately, a number of centres continue to refer candidates to outdated project guidelines. Candidates subsequently spend time completing work and generating evidence such as scope and boundaries, feasibility studies, user surveys, lists of inputs, processes and outputs, and progress diaries. This is unnecessary and will receive no marks.

Use of frameworks or libraries

Although fewer projects were reliant on frameworks or libraries than in previous years, it was evident that a few candidates had based their projects on templates or tutorials. When candidates rely on frameworks, libraries, templates or tutorials to build their solution, they are often unable to generate the evidence needed to gain good marks at the Design and Implementation stages of the development. Teachers and lecturers should encourage candidates to consider projects that would enable them to demonstrate coding skills gained as part of the Advanced Higher course and focus on the functionality of their solutions, rather than the usability of the interface, which is not assessed within the Advanced Higher project.

Presentation of evidence

When preparing their project evidence for submission, candidates should be encouraged to use the headings in the marking instructions within the coursework assessment task to organise and present their evidence, and to number all pages in their submission. Code must be printed in a font that is large enough to be easily read by markers. Marks can only be awarded if the Advanced Higher concepts, integration, and listed requirements can all be identified in the implemented code. To assist with this, candidates should be reminded to highlight their code to indicate where these features of their solution can be found. When screenshots of code are presented, these must be large enough to enable markers to easily read the code and, where possible, use of a black background should be avoided. Teachers and lecturers should encourage candidates to check that they have been consistent in referring to the requirements identified at Analysis; at the end of the Design and Implementation stages, candidates should check that all requirements are accounted for and that no new requirements have been introduced.

Description of the problem

In their problem descriptions, after a brief introduction to the problem they are solving, candidates should indicate the type of project being undertaken before making specific reference to each of the mandatory Advanced Higher and integrative concepts listed on the relevant page of the coursework assessment task document. Problem descriptions must give details of the intended use of each concept and clearly indicate what input validation will be needed; at Advanced Higher level, it is not enough to simply state that a concept will be used, and input validation will happen.

Requirements specification

Centres should remind candidates that all inputs to the final program must be validated, and as this is a mandatory requirement of all Advanced Higher projects, details of the intended input validation must be listed in the functional requirements at the Analysis stage. Teachers and lecturers should encourage candidates to number each of their requirements so that they can easily refer to them in later stages of the development; this is very important. Candidates must ensure that all requirements are accounted for at the Design stage, that they have been coded during Implementation, that they have been tested at the Testing stage and that they are evaluated at the Evaluation stage. Candidates who present a clear, well-defined list of requirements at the Analysis stage generally receive good marks in all stages of the project development.

Design approaches

Teachers and lecturers should advise candidates that code presented at the Design stage as evidence of the design will receive no marks. As the SQA Reference Language is a programming language with its own syntax, this advice also applies to use of SQA Reference Language at the Design stage. Rather than using pseudocode to design their solution, many candidates spend a lot of time attempting to reverse engineer their implementation code, adding line numbers and rewriting every program instruction using structured English. To avoid this retrospective generation of pseudocode, teachers and lecturers should encourage candidates to consider what processes they will need to implement their list of requirements at the Design stage, before any code is produced. They should show these processes in a top-level design that also shows any necessary data flow between those processes. For software development projects, candidates should ensure that input validation routines and any additional functionality stated at Analysis have been indicated in the design, together with refinements of all Advanced Higher algorithms used. For web projects, candidates should provide a top-level design for individual pages. The design should indicate the main processes that will take place on the page, together with any planned use of session variables.

As part of their design evidence, candidates who attempt a software development project must indicate the intended structure of any 2D array or array of records that will be used in their solution. This design should indicate the dimensions of the array and the data type or types that will be needed. Similarly, candidates who opt to complete a web project must provide a navigation structure of the planned website. This should show all planned pages and indicate links between pages, together with any redirects that may be necessary. The site navigation structure may also be used to indicate the sharing of data between pages by showing where session variables will be used.

Evidence that mandatory and functional requirements are fit for purpose

Teachers and lecturers should remind candidates of the importance of producing evidence to show that all mandatory and functional requirements are fit for purpose. In software development projects, candidates must provide evidence to show that any Advanced Higher algorithm coded in the solution is working correctly. Although most candidates remember to include a screenshot of the sorted output produced by the sort code, they forget about the need to evidence the unsorted data — without this, markers cannot award full marks. Similarly, evidence of a working binary search is incomplete unless it includes a screenshot showing the full list of values that have been searched by the code. For all projects that

implement a database, candidates must submit evidence to show that the structure of the implemented table or tables matches the structure indicated in the data dictionary produced at the Design stage. This evidence can be in the form of the SQL code or screenshots of the implemented table showing the necessary structural details. In addition, these candidates must also present evidence to show that all queries are fit for purpose. Teachers and lecturers should advise candidates to include evidence of the initial values stored in tables before any of the implemented queries are executed, for example a list of quiz questions or a list of stock items. This advice applies even if the tables initially have no records, for example a member table with no records or a high score table with no entries. This evidence could be the SQL code used to populate the table or screenshots showing the initial values or empty table. Candidates should also take screenshot evidence to show that all implemented queries are fit for purpose. They should ensure that:

- evidence for INSERT queries shows additional records in the table
- evidence for UPDATE queries shows that existing records in the table have been edited
- evidence for DELETE queries shows that records have been removed from the table
- evidence for SELECT queries shows that results returned have been retrieved from the underlying table

Ongoing testing

Throughout the implementation of their solution, candidates must maintain a log of ongoing testing. Although it is not necessary for candidates to log every minor syntax error or mismatch that they encounter, they should note when important components are added to the solution. For example, whenever they implement a mandatory Advanced Higher concept or functional requirement, candidates should automatically run their code to check that this additional functionality is working correctly. They should record evidence of that testing in the log of ongoing testing. Depending on the sequence of implementation, it may be necessary for candidates to add temporary print lines or stubs and driver code to test certain sections of the solution; examples of this would be very good evidence to include in the log. Where possible, candidates should be encouraged to make use of breakpoints or watchpoints to pause the code during execution to check what values are being stored internally. Screenshots of this would be very good evidence that could be used in the log of ongoing testing, or as evidence of final testing. A brief note of any issues encountered as components are implemented should be added to the log, together with the full details of any references that were used to resolve the problem. If candidates resolve issues themselves, they should note this in the log.

Test plans

Candidates' test plans should list each of the functional and end-user requirements that were identified at the Analysis stage. They should provide a description of the test needed for each requirement, together with details of any test data values used. When testing input validation code, test data values should include normal, extreme and exceptional values. Having generated their test evidence, many candidates choose to present the screenshots for individual tests in an additional column of their test plan. As a result, these screenshots end up being very small, and it is often extremely difficult for markers to read the details being presented. Teachers and lecturers should encourage candidates to present the evidence for each test in the test plan below the test plan itself. Numbering each individual

test in the test plan and labelling the matching test evidence accordingly would ensure that test evidence was readable, and markers were able to reward candidates appropriately for well-structured test plans with well-presented test evidence.

Evaluation

In their evaluations of fitness for purpose, candidates must discuss whether the final solution matches all the requirements listed at the Analysis stage. If issues were encountered during the Implementation or Testing stages, candidates should refer to these in their evaluation. To enable candidates to discuss the maintainability of their solution using terminology appropriate to Advanced Higher level, teachers and lecturers must ensure that all candidates are familiar with the three types of maintenance introduced at Advanced Higher: adaptive, corrective and perfective. Evaluations of maintainability should refer to aspects of their code that candidates feel would help (or indeed hinder) specific types of maintenance that may be necessary in the future. When discussing the robustness of their solution, candidates should refer to the input validation routines incorporated within the solution, and also to the results of testing of those aspects of the solution.

Appendix: general commentary on grade boundaries

SQA's main aim when setting grade boundaries is to be fair to candidates across all subjects and levels and maintain comparable standards across the years, even as arrangements evolve and change.

For most National Courses, SQA aims to set examinations and other external assessments and create marking instructions that allow:

- a competent candidate to score a minimum of 50% of the available marks (the notional grade C boundary)
- a well-prepared, very competent candidate to score at least 70% of the available marks (the notional grade A boundary)

It is very challenging to get the standard on target every year, in every subject at every level. Therefore, SQA holds a grade boundary meeting for each course to bring together all the information available (statistical and qualitative) and to make final decisions on grade boundaries based on this information. Members of SQA's Executive Management Team normally chair these meetings.

Principal assessors utilise their subject expertise to evaluate the performance of the assessment and propose suitable grade boundaries based on the full range of evidence. SQA can adjust the grade boundaries as a result of the discussion at these meetings. This allows the pass rate to be unaffected in circumstances where there is evidence that the question paper or other assessment has been more, or less, difficult than usual.

- The grade boundaries can be adjusted downwards if there is evidence that the question paper or other assessment has been more difficult than usual.
- The grade boundaries can be adjusted upwards if there is evidence that the question paper or other assessment has been less difficult than usual.
- Where levels of difficulty are comparable to previous years, similar grade boundaries are maintained.

Grade boundaries from question papers in the same subject at the same level tend to be marginally different year on year. This is because the specific questions, and the mix of questions, are different and this has an impact on candidate performance.

This year, a package of support measures was developed to support learners and centres. This included modifications to course assessment, retained from the 2021–22 session. This support was designed to address the ongoing disruption to learning and teaching that young people have experienced as a result of the COVID-19 pandemic while recognising a lessening of the impact of disruption to learning and teaching as a result of the pandemic. The revision support that was available for the 2021–22 session was not offered to learners in 2022–23.

In addition, SQA adopted a sensitive approach to grading for National 5, Higher and Advanced Higher courses, to help ensure fairness for candidates while maintaining

standards. This is in recognition of the fact that those preparing for and sitting exams continue to do so in different circumstances from those who sat exams in 2019 and 2022.

The key difference this year is that decisions about where the grade boundaries have been set have also been influenced, where necessary and where appropriate, by the unique circumstances in 2023 and the ongoing impact the disruption from the pandemic has had on learners. On a course-by-course basis, SQA has determined grade boundaries in a way that is fair to candidates, taking into account how the assessment (exams and coursework) has functioned and the impact of assessment modifications and the removal of revision support.

The grade boundaries used in 2023 relate to the specific experience of this year's cohort and should not be used by centres if these assessments are used in the future for exam preparation.

For full details of the approach please refer to the National Qualifications 2023 Awarding — Methodology Report.