Course report 2024

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 with the published assessment documents and marking instructions.

We compiled the statistics in this report before we completed the 2024 appeals process.

Grade boundary and statistical information

Statistical information: update on courses

Number of resulted entries in 2023: 665

Number of resulted entries in 2024: 705

Statistical information: performance of candidates

Distribution of course awards including minimum mark to achieve each grade

A	Number of candidates	187	Percentage	26.5	Cumulative percentage	26.5	Minimum mark required	93
B	Number of candidates	155	Percentage	22	Cumulative percentage	48.5	Minimum mark required	79
C	Number of candidates	148	Percentage	21	Cumulative percentage	69.5	Minimum mark required	66
D	Number of candidates	111	Percentage	15.7	Cumulative percentage	85.2	Minimum mark required	52
No award	Number of candidates	104	Percentage	14.8	Cumulative percentage	100	Minimum mark required	N/A

We have not applied rounding 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 statistical reports on the statistics and information page of our website.

Section 1: comments on the assessment

Question paper

The question paper largely performed as expected. Feedback indicated it was fair and accessible for candidates; however, a few centres focused on the reintroduction of mandatory integration in the optional sections. Despite this, most candidates understood what was required, and completed the mandatory and chosen optional sections in the allocated time.

In the question paper, 54% of candidates completed the 'Database design and development' section, while 46% completed the 'Web design and development' section.

The overall performance of candidates who attempted questions in 'Web design and development' was better than the performance of the candidates who attempted 'Database design and development'. This was observed in the questions common to each section as well as the specialist questions.

Most questions performed as expected, with the C-level questions at the start of each section helping candidates to focus on the section's specialist content before tackling more demanding problem-solving questions.

However, feedback and statistical data on the integrated questions in the optional sections indicated that some candidates were not prepared for these questions. While these were valid questions, we considered candidates' level of engagement with them when we set grade boundaries. We adjusted the grade boundaries for C and A grades by 1 mark to take account of the new assessment approach.

Project

Revisions to the coursework assessment task published in September 2023 aimed to clarify the evidence requirements for each type of project by providing detailed evidence checklists for candidates. Although there were improvements in the evidence presented in the 'Testing' section, overall, the evidence submitted suggested that candidates did not use the checklists and some candidates continued to provide incomplete evidence of design and implementation.

Many projects that candidates attempted were overly complex, with over 30, 40 or, in a few cases, 50 requirements being identified at the analysis stage. As a result, the volume of evidence candidates presented was excessive, sometimes amounting to well over 300 pages. Having so many requirements makes it difficult, if not impossible, to track each requirement through the remaining stages of development. As a result, these candidates were unable to access all the marks available as their evidence could not be judged to be 'complete' or 'almost complete' in relation to the problem they set out to analyse.

Section 2: comments on candidate performance

Areas that candidates performed well in

Question paper

Software design and development

Question 2(b):	Most candidates were able to correctly indicate the value of the pointer of each node in the sorted list.
Question 3(a):	Most candidates were able to define a 2-D array structure by correctly indicating its data type and dimensions.
Question 3(b):	Most candidates correctly assigned the value to the appropriate element.
Question 5(a):	Many candidates were able to accurately explain the use made of overriding to redefine the `calcArea()` method of each subclass.
Question 5(b):	Although some candidates omitted detail of the default value assigned to the instance variable `area` by the constructor method of the `Rectangle` class, many candidates were able to partially explain the effect of the code provided in 5(b)(i). Most candidates correctly stated the output produced by the code in 5(b)(ii).
Question 5(c)(i):	Most candidates correctly identified that the code would set the fill colour of the roof to red.
Question 5(d):	Most candidates were able to write the code needed to implement steps 4 and 8 of the design.

Database design and development

Question 7(a)(b):	Most candidates showed an understanding of entity types and many were awarded at least 1 mark for describing the relationship participation between the actor and role entities.
Question 8(a):	Many candidates correctly stated 'technical' as the type of feasibility study.
Question 8(b):	Many candidates were able to produce an accurate UML use case diagram to represent the appointments system.
Question 8(e):	Most candidates were able to identify the use made of the `BETWEEN` operator.
Question 8(f):	Most candidates stated the correct output from the subquery and the query.

Web design and development

Question 10(a)(b):	Many candidates were able to write the correct HTML statement.
Question 11(a):	Most candidates stated 'technical' as the type of feasibility study. Candidates who completed the 'Web design and development' section performed better in this question than candidates who answered 8(a) in 'Database design and development'.
Question 11(b):	Many candidates were able to produce an accurate UML use case diagram to represent the website.
Question 11(c)(i):	Many candidates were able to provide the missing pseudocode needed to complete the design.

Question 11(c)(ii):	Most candidates wrote the correct PHP code needed to assign the submitted data to PHP variables.
Question 11(d)(i):	Many candidates correctly explained that a session variable is necessary to allow the entered email to persist on additional pages of the website.
Question 11(e)(i):	Many candidates identified the username for this database connection.
Question 11(f)(i):	Most candidates were able to describe the purpose of the design correctly.

Areas that candidates found demanding

Question paper

Software design and development

Question 1:	Many candidates did not correctly state the three comparisons required to locate 'Sycamore' using a binary search.
Question 2(a):	Many candidates did not identify the value that would be stored in the head pointer.
Question 4(a):	Most candidates did not get full marks for this question, with many incorrectly describing end-user or acceptance testing. Some candidates explained how the persona and test case in the description would be used to carry out final testing of the appointment system.
Question 4(b):	Many candidates did not identify perfective maintenance with an appropriate justification.
Question 5(c)(ii):	Most candidates incorrectly stated that the method had not been defined. Only a few candidates were able to explain that the element `list[8]` is an object belonging to the `Shape` superclass, and therefore were not able to access a method that belongs to the `Circle` subclass.
Question 6(b)(i):	Many candidates did not provide the pseudocode needed to complete the insertion sort algorithm. A few candidates did, however, receive marks for incomplete designs.

Database design and development

Question 8(c):	Many candidates did not accurately describe two benefits of using a surrogate key for the stated entity, with a number making inaccurate claims about the use of a surrogate key.
Question 8(d):	Although a few candidates received 1 mark for the correct subquery, many candidates did not complete the SQL query using the `EXISTS` operator with the subquery.
Question 8(g):	Most candidates did not provide either the `action` or `method` attributes needed to complete the code. This question also had a low response rate compared with the rest of the paper.

Web design and development

Question 11(d)(ii):	While many candidates could explain why a session variable is necessary in 11(d)(i), only some could describe the use made of the session variable on the 'View Climbs' page.
Question 11(f)(ii):	Most candidates did not identify a potential problem with the design at line 5. This question also had a low response rate compared with the rest of the paper.

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

Stage 1: Analysis

Description of problem, UML use case diagrams and project plan for each stage

Most candidates provided accurate descriptions of the constraints on the development, and many used UML use case diagrams correctly to illustrate how end-users and external databases would interact with the system. However, some problem descriptions lacked sufficient detail of the relevant Advanced Higher and integrative concepts for the type of project. The project plans produced by many candidates omitted details of essential subtasks for both the design and implementation stages of the development.

Requirements specification

While most candidates received at least 2 marks for their requirements specification, many projects were overly complex and generated an excessive number of requirements. It was extremely difficult for these candidates to ensure that each requirement was addressed appropriately at each subsequent stage of the development process. In addition, many candidates' end-user requirements focused more on aesthetics than the requirements identified in the UML use case diagrams.

Stage 2: Design

Design of Advanced Higher concepts and integration

For projects that required integration with a database, the design of integration was much improved since last year, with more candidates providing complete and accurate evidence of database design in the form of entity-relationship diagrams and data dictionaries. Refinements of individual processes were used to indicate where connections to the database and database queries would be needed, and query designs were provided to indicate their intended purpose.

Many candidates who tackled software development projects continued to present pages of code, or reverse engineered code that they rewrote and presented as the design of their Advanced Higher concepts, integration and functional requirements. By contrast, most candidates who tackled projects with a focus on database and web development presented small sections of pseudocode that outlined the sequence of actions needed to implement each individual process.

Some candidates who completed software development projects provided a top-level design of their proposed solution that indicated data flow, but in general, many candidates did not include this evidence. When refining the Advanced Higher search or sort algorithm, some candidates applied the algorithm to a 1-D array rather than the Advanced Higher data structure that had been identified in their problem description. In addition, many candidates relied on frameworks that automatically generated the UML class diagram and made no further reference to the array of objects in the remainder of their design. It was clear that these candidates had no intention of applying the Advanced Higher algorithm to this data structure.

For most candidates who completed a web project, the planned use of session variables was unclear and, in many cases, contradictory. For example, wireframe designs indicated personalised welcome messages while the refinement of the login process either made no mention of the session variables needed to display those messages or suggested that different session variables would be used. Although most candidates did indicate the effect that media queries would have, either in the form of wireframes or in a written description, some candidates did not indicate when the media query would be triggered. Similarly, many candidates did not provide a site navigation structure to show the planned pages of the site and indicate what navigation would be possible between the pages. The few navigational structures that were submitted often omitted planned redirects and links indicated elsewhere in the design.

User interface design

Many candidates received good marks for the design of their user interface. Candidates used wireframes to indicate the intended layout of input and output, and described the underlying processes executed when selecting buttons and menu options.

Although many candidates did attempt to show the input validation that would be carried out as part of their user interface design, most candidates omitted the details needed to show the error messages that would be produced by that validation.

Design matches requirements specification

As a result of including so many requirements at the analysis stage, many candidates did not provide sufficient evidence to show that they had considered all requirements at the design stage.

Stage 3: Implementation

Implemented Advanced Higher concepts, integration and user interface

Most candidates gained good marks for the implementation of the Advanced Higher concepts and the implemented user interface, with many candidates also gaining good marks for the implementation of integration.

Most candidates who chose to implement a database did not provide evidence of the initial contents of the implemented table or tables, and the data used to populate them before any queries were executed. In addition, many candidates failed to submit evidence of the structure of the implemented table or tables to show that the implementation matched the design indicated in the data dictionary.

Although most candidates who implemented an Advanced Higher search and/or sort algorithm submitted screenshots to show the results of how the algorithm performed, they did not provide the 'before' evidence of the data searched or the unsorted data, which is needed to show that the algorithm worked correctly.

Log of ongoing testing

Some candidates failed to provide any evidence of the ongoing testing that would have been carried out automatically throughout the implementation stage. Where evidence was provided, it was often incomplete and omitted many of the important Advanced Higher concepts, integrative components and functional requirements that had been indicated at the analysis and design stages.

Stage 4: Testing

Most candidates gained good marks for carrying out all tests listed in their test plan and providing screenshot evidence of them. However, a few candidates provided no description of the results of testing based on the test cases provided in their test plan.

Candidates who had included requirements at the analysis stage that focused on aesthetics, such as ‘clutter-free layout’ and ‘easy to use’, encountered difficulty generating evidence of final testing, which they are expected to carry out and evaluate themselves.

Stage 4: Evaluation

Fitness for purpose, maintainability and robustness

Most candidates who referred to each of the requirements identified at the analysis stage received 1 mark for an accurate description of their solution’s fitness for purpose.

In many cases, the descriptions of fitness for purpose lacked sufficient detail at Advanced Higher level and lacked any qualitative evaluation of the solution. Similarly, many evaluations of maintainability did not explain how features of the solution would aid (or hinder) future maintenance tasks, and many evaluations of robustness included inaccurate claims that all input had been validated when there was clear evidence in the solution that this was not the case.

Section 3: preparing candidates for future assessment

Teachers and lecturers must ensure that they are familiar with the current version of the course specification. It is essential that teachers and lecturers are aware of the requirements for integration within the question paper that this document outlines.

Question paper

Teachers and lecturers should ensure that candidates are familiar with the Advanced Higher standard algorithms, binary search, bubble sort and insertion sort. In particular, candidates should be aware of the distinguishing features of each algorithm and be able to demonstrate their understanding of them by applying them to given lists of data values. Similarly, candidates should be familiar with the operation of single and double linked lists, and able to demonstrate that understanding by applying it to linked lists consisting of values and pointers.

Teachers and lecturers should ensure that candidates are familiar with the correct terminology for, and are able to recognise and accurately describe, component testing, end-user testing, integrative testing, final testing and usability testing based on prototypes. Candidates should be able to accurately describe the use of personas and test cases in relation to final and usability testing.

Teachers and lecturers should ensure that candidates are familiar with the correct terminology for adaptive, corrective and perfective maintenance. Similarly, centres should also ensure that candidates are familiar with the correct terminology for economic, time, legal and technical feasibility studies.

Teachers and lecturers should advise candidates to pay close attention to any UML class diagram that lists all the methods available to the main program code. Although program code will be presented, it is likely that the code will be incomplete. It is vital, therefore, that candidates understand the relationship between the UML class diagram and the code presented. Teachers and lecturers should also ensure that candidates are aware of how polymorphism applies to an array of superclass objects. Although individual elements of a superclass array may reference a variety of different types of objects (including superclass objects and objects of different subclasses), those superclass array elements can only access methods that have been defined for the superclass.

Overall, candidates coped well with the problem-solving questions that required them to design algorithms for unseen tasks and processes such as those in question 6. 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.

For 'Database design and development', teachers and lecturers should ensure that candidates can accurately describe the benefits of using a surrogate key to replace a compound key. When preparing candidates for section 2 of the question paper, they should ensure that candidates are familiar with the skills, knowledge and understanding needed to design and implement HTML forms. There is an example of this requirement in question 7(f) of the specimen question paper.

For ‘Web design and development’, teachers and lecturers should ensure that candidates are familiar with all the PHP coding and mysqli functions required at Advanced Higher level. Candidates should understand the purpose of each mysqli function and know when it can be used. At the same time, they should ensure that candidates have opportunities to read and explain unfamiliar pseudocode used to design server-side processes that would be implemented using these mysqli functions. When preparing candidates for section 3 of the question paper, centres should ensure that candidates are familiar with the skills, knowledge and understanding needed to design and implement SQL queries. There is an example of this requirement in question 10(d)(i) of the specimen question paper.

Project

Coursework assessment task

Centres must ensure that they are using the correct version of the coursework assessment task. This includes evidence checklists for each type of project. These provide details of the evidence required at each stage of the project. Unfortunately, some centres continue to refer candidates to outdated project guidelines and sample projects. Candidates who use the appropriate documents are less likely to spend time generating evidence that receives no marks, such as scope and boundaries, feasibility studies, user surveys, lists of inputs, processes and outputs, and progress diaries.

Project selection

When candidates are selecting projects, teachers and lecturers should advise them to be realistic about the extent of what they can achieve working independently. They should ensure that candidates do not embark on overly complex projects that generate an excessive number of requirements. Projects with a limited number of requirements will help candidates to keep track of them and ensure that they can access all marks in each stage of the development. For session 2024–25, the coursework assessment task has been updated to state that projects must have no more than 6 end-user requirements and 24 functional requirements.

The Advanced Higher Computing Science course is a general course that introduces candidates to advanced coding skills in three areas: software design and development, database design and development, and web design and development. Although candidates must focus on one area of the course and integrate with one other area, the course is not intended to be a specialist web development or games development course. Teachers and lecturers should discourage candidates from developing solutions that a team of professional developers would normally carry out.

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 in the Advanced Higher project. Some candidates opted to integrate software design and development with web design and development. Given that all IDEs provide a central interface to edit, compile and debug code, and that there are no marks available for the user-friendliness of the solution, there is no need to send program output to a separate web interface. Since most of these candidates relied on a framework to handle the complexities associated with the connectivity across the two technologies, they did not provide any design evidence to show that they understood how the connectivity would be achieved. Additionally, since the technical details of the implementation were largely hidden by the framework used, it was often difficult to award full marks for implementation of the connection and integration.

Centres should remind candidates that all inputs to their solution must be validated, and since this is a mandatory requirement of all Advanced Higher projects, details of the intended input validation must be provided in the functional requirements at the analysis stage. Checking whether a user is already registered is user authentication and not input validation; the input validation that is required in the Advanced Higher project should build on the validation techniques introduced in the National 5 and Higher Computing Science courses.

While user authentication is an importance process carried out by many systems, it is an additional requirement of the Advanced Higher project that does not meet the need to validate all input values.

Presentation of evidence

When preparing their project evidence for submission, candidates should use the subheadings in the evidence checklists in the project assessment task to organise and present their evidence. To make it possible for markers to track each requirement at each stage of the development, candidates must number all pages in their submission. Code must be printed in a font that is large enough to be easily read by markers who can only award marks if they identify Advanced Higher concepts, integration and listed requirements in the implemented code. To assist with this, candidates should highlight their code to show where to find these features. Screen shots of code must be large enough for markers to read them easily, and, where possible, candidates should avoid using a black background. Where screen shots of the user interface are used to support evidence of testing, it should be possible for markers to read input values and output messages in those screen shots without needing to magnify the page. For session 2024–25, a template for gathering evidence will be available on the Advanced Higher Computing Science subject page. This is designed to help candidates present their evidence and to track their requirements through each stage of the development.

Analysis

Centres should encourage candidates to use a UML use case diagram early in the analysis stage to help them to identify the end-user requirements they will need in their solution. There is no need for candidates to carry out user surveys to help identify them. Once completed, the UML use case diagram should provide candidates with a high-level understanding of how end-users of the system will interact with it, and the tasks they should be able to perform. Identifying the use cases needed to complete the diagram should be the first step in helping candidates to identify several of the back end functional requirements that will be needed to support the end-user requirements. In addition, the UML use case diagram should also show how the system will interact with external databases and files, which should help candidates to identify additional functional requirements.

In the requirements specification, candidates should specify all end-user requirements identified in their UML use case diagram. It may be helpful for candidates to use the heading 'End-users of the system should be able to' and follow this with the list of tasks that end-users should be able to perform using the completed system. Tackling the end-user requirements in this way would avoid all end-user requirements that are concerned with the aesthetics and usability of the solution, which candidates can find difficult to test and evaluate. Centres should remind candidates that the functional requirements specified in the requirements specification must include all mandatory Advanced Higher and integrative requirements listed in the coursework assessment task.

Design

As part of their design evidence, candidates who attempt a software development project are expected to indicate the intended structure of any Advanced Higher data structure they will use in their solution. This data structure may be a 2-D array, an array of records or an array of objects. In addition, the design of the Advanced Higher search or sort algorithm must clearly indicate the planned use of this Advanced Higher data structure; generic algorithms copied from class notes or applied to a 1-D array, or parallel 1-D arrays, are not appropriate.

Implementation

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 has 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 tables, or screenshots showing the initial values or empty tables. Candidates should also include 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 are expected to 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 are expected to note when important components are added to the solution. For example, whenever a mandatory Advanced Higher concept or functional requirement is implemented, it is expected that candidates would automatically run their code to check that this additional functionality is working correctly. Evidence of that testing should be recorded in the log of ongoing testing, even when no issues were encountered. Depending on the sequence of implementation, it may be necessary for candidates to add temporary print lines or stubs/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 — screen shots of this would be very good evidence that could be used in the log of ongoing testing or as evidence of final testing. Candidates should add to the log a brief note of any issues encountered as components are implemented, together with the full details of any references that were used to resolve the problem. If candidates resolved issues themselves, or encountered no difficulties implementing a particular component, they should record this in the log.

Testing

Teachers and lecturers should remind candidates of the need to provide a description of at least one persona for the purposes of final testing. The persona should be a realistic character that represents a typical end-user of the completed system. The description of the persona should provide background details such as name, age and occupation, and, importantly, should indicate their level of IT skills and knowledge.

Along with the persona, candidates should also provide test cases. These should outline tasks that typical users of the system would carry out. Although it is not necessary to do so, it is useful to include the test cases as part of the formal test plan. This not only ensures that the tests are completed, but also generates evidence of testing that can be included in the submission and helps candidates to report on the results of testing using the test cases.

Candidates should carry out final testing of the solution themselves. There is no need for candidates to carry out end-user or usability testing. They should provide screen shot evidence of each test in the test plan, and where appropriate, it should indicate input values stated in the test plan together with predicted validation error messages.

Evaluation

In their evaluations of fitness for purpose, candidates are expected to discuss the extent to which their solution matches all the requirements listed at the analysis stage. 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 are expected to refer to the input validation routines incorporated within the solution, and to the results of testing of those aspects.

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.

Every year, we evaluate the performance of our assessments in a fair way, while ensuring standards are maintained so that our qualifications remain credible. To do this, we measure evidence of candidates' knowledge and skills against the national standard.

During the pandemic, we modified National Qualifications course assessments, for example we removed elements of coursework. We kept these modifications in place until the 2022–23 session. The education community agreed that retaining the modifications for longer than this could have a detrimental impact on learning and progression to the next stage of education, employment or training. After discussions with candidates, teachers, lecturers, parents, carers and others, we returned to full course assessment for the 2023–24 session.

SQA's approach to awarding was announced in March 2024 and explained that any impact on candidates completing coursework for the first time, as part of their SQA assessments, would be considered in our grading decisions and incorporated into our well-established grading processes. This provides fairness and safeguards for candidates and helps to provide assurances across the wider education community as we return to established awarding.

Our approach to awarding is broadly aligned to other nations of the UK that have returned to normal grading arrangements.

For full details of the approach, please refer to the National Qualifications 2024 Awarding — Methodology Report.