Course Report 2025

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 2025 appeals process.

Grade boundary and statistical information

Statistical information: update on courses

Number of resulted entries in 2024: 705

Number of resulted entries in 2025: 767

Statistical information: performance of candidates

Distribution of course awards including minimum mark to achieve each grade

Course award	Number of candidates	Percentage	Cumulative percentage	Minimum mark required
A	195	25.4	25.4	94
B	192	25.0	50.5	80
C	170	22.2	72.6	67
D	115	15.0	87.6	53
No award	95	12.4	100%	Not applicable

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 or equal to 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. Most candidates understood what was required and completed the mandatory and their chosen optional section in the allocated time.

The C-level questions at the start of each section helped candidates focus on the specialist content of each section before tackling the more demanding problem-solving questions that followed. In the question paper, 57.3% of candidates attempted questions in section 2, ‘Database design and development’, and the remaining 42.7% of candidates attempted questions in section 3, ‘Web design and development’.

Project

The introduction of a template to assist candidates in September 2024 resulted in a small improvement in results. Prompts in the template helped to ensure that candidates were aware of what evidence was required at each stage, and also served to remind candidates that requirements introduced at the analysis stage must be tracked throughout the development of the solution.

Despite clear guidance in the coursework assessment task and the template to limit the number of functional requirements, some candidates did not follow the guidance. As a result, the scope of their projects and the volume of evidence they presented was excessive. Having so many requirements makes it difficult, if not impossible, to track each of those requirements through the remaining stages of the development of their solution, and consequently, most of those candidates were unable to access all the marks available.

In addition, some candidates did not follow the template and continued to submit unnecessary evidence, such as scope, boundaries, feasibility studies, and user surveys.

Section 2: comments on candidate performance

Areas that candidates performed well in

Question paper

Software design and development

Question 1: Most candidates were able to state the correct type of linked list and supported their choice with a reason.

Question 2(a): Many candidates correctly spotted that the array details had been arranged in ascending order of company name.

Question 2(c): Many candidates correctly identified errors in the algorithm and described the corrections required.

Question 3(b): Many candidates made appropriate use of OO terminology to explain why the `calcOrderTotal()` method appears in both the `Order` and `Delivery` classes.

Question 3(d): Many candidates were able to provide an accurate explanation of why Line 5 could not be executed.

Question 4(a): Many candidates were able to write the code needed to declare the 2-dimentional array events by correctly identifying the data type and dimensions needed.

Question 4(d)(i): Many candidates correctly stated the correct type of maintenance needed and supported their answer appropriately.

Database design and development

Question 6(a): Many candidates correctly identified the strong/weak entities, and the correct relationship participation needed to correctly complete the ERD.

Question 6(c): Most candidates were able to complete the SQL query correctly.

Question 7(a): Many candidates were able to complete the query by correctly using the SQL BETWEEN and HAVING operators to complete the conditions.

Question 7(c): Many candidates provided an accurate explanation of how the SQL NOT EXISTS operator had been used to generate the required results.

Question 7(d)(i): Many candidates correctly identified the missing method in part A and the correct HTML input or button element needed in part B.

Question 7(d)(ii): Many candidates were able to explain the need for integrative testing when developing a database-driven website.

Web design and development

Question 8: Many candidates were able to describe how the results gained from user surveys could be used at the analysis stage of a project.

Question 9(a): Many candidates provided an accurate description of how the tablet layout could be adapted to produce the A4 paper layout.

Question 9(b): Many candidates were able to explain the need for integrative testing when developing a database-driven website.

Question 10(a)(i): Many candidates provided an accurate explanation of how sessions and session variables had been used in the layout shown.

Question 10(a)(ii): Most candidates were able to write the PHP code needed by making appropriate use of the connection details provided.

Question 10(b)(iii): Many candidates were able to design the section of code used to allocate points to the competitors.

Areas that candidates found demanding

Question paper

Software design and development

Question 3(c): Only some candidates were able to explain that the array was able to store several different types of data because the array was of type `Order` which is a superclass, and all three elements of the array were of subclass types (`Delivery`, `SitIn` and `Collection`) that all fall under the umbrella of the `Order` superclass.

Question 3 (e)(i): Despite demonstrating their understanding of array indexing elsewhere in their responses, many candidates' responses to this question referred to `orderList[1]` being an object of type `Delivery`, which led them to the spurious conclusion that the `addExtraItem()` method cannot be accessed by `Delivery` objects.

Question 3(e)(ii): Only a few candidates were able to correctly suggest how an additional item could be added to an existing `SitIn` order.

Question 4 (b)(i): Although many candidates did receive one or more marks for their responses, some candidates produced code to implement a bubble sort algorithm rather than an insertion sort. Others incorrectly sorted the array in ascending rather than descending order, and a few candidates failed to use appropriate 2-D array

indexing. In addition, instead of looping from the second to last element of the array, the Python solution of many candidates generated the wrong number of iterations.

Question 4 (b)(ii): Although some candidates were able to accurately describe how before-and-after evidence could be used to demonstrate that data in the array had been sorted correctly, many candidates were unable to do so.

Database design and development

Question 6 (b): Although some candidates were able to state one reason why the Allocation entity would benefit from use of a surrogate key, by simply stating that a surrogate key would be easier or quicker, the responses of many candidates lacked sufficient detail. In addition, a few candidates incorrectly stated that the introduction of a surrogate key would mean that the 'Allocation' entity would now have a unique value for its primary key.

Question 7(d)(iii): Although some candidates were able to describe how end-user testing would be carried out, many candidates incorrectly described how the development team could use personas during final testing.

Web design and development

Question 10(b)(i): Although many candidates were able to describe the purpose of the SQL query at line 37, only a few candidates were able to explain the relevance of arranging the competitor details in ascending order in this particular situation.

Question 10 (b)(v): Many candidates incorrectly used an `INSERT` query rather than an `UPDATE` query.

Areas that candidates performed well in and areas that candidates found demanding

Project

Analysis

Many candidates performed well in the analysis section of the project. Candidates described constraints on the development accurately, and many candidates used UML use case diagrams correctly to illustrate how end users and external databases or files would interact with the completed system. In addition, rather than focusing on aspects of the user interface in their requirements specification, many candidates' end-user requirements were much more focused on the tasks that end users would be able to carry out using the system, which resulted in more specific requirements that candidates were then able to track at each subsequent stage of the development.

Some candidates continue to undertake extremely complex projects that generate an excessive number of requirements. With so many requirements, it was extremely difficult for those candidates to ensure that each requirement had been handled appropriately at each subsequent stage of the development process.

Although many candidates did indicate the resources that would be needed to implement their solution, some candidates tackling a software design and development solution failed to clearly state which programming language that they would use. Also, many candidates' project plans were very generic, simply listing the tasks for each stage as the headings used in the project marking instructions. Also, in some cases, the time scales indicated in those plans were often unrealistic and often omitted any ongoing testing that would be necessary at the implementation stage of the development.

Design

For projects that required integration with a database, the design of integration continues to improve; many more candidates provided complete and accurate evidence of database design in the form of data dictionaries, ERDs, query design and refinements of processes that indicated where connections to the database and the execution of database queries would take place.

Many candidates' user interface design continues to show improvement, with many candidates producing wireframes to indicate the intended layout of input and output and providing descriptions of underlying processes and button and/or menu options.

As a result of having fewer requirements overall, candidates were less likely to omit essential design evidence, and therefore gained more marks for their design in matching the requirements specification.

Many candidates who tackle software design and development projects continue to present revered engineered code that has been painstakingly and time-consumingly edited so it can be presented as the design of their Advanced Higher concepts, integration and functional requirements. By contrast, most candidates who tackle projects with a focus on database design and development and web design and development present small sections of pseudocode that outline the sequence of actions that will be needed to implement each individual process.

Some candidates who tackled procedural programming projects did not provide a top-level design with intended data flow between modules. These candidates also tended to fail to provide a design of the array of records or 2D array that indicated planned data types and dimensions of the data structure they intended to implement. In many cases candidates either omitted this evidence or provided code from their final solution. Similarly, some candidates who completed an object-oriented programming project failed to provide evidence of a UML class diagram indicating the fields and methods of any classes that they intend to implement from scratch. They often presented a UML class diagram generated automatically by the framework used to implement the user-interface and/or game play.

Many candidates who set out to implement an object-oriented solution, submitted a procedural solution that applied an Advanced Higher algorithm to an array of records rather than an array of objects. In most of these cases, the object-oriented content of their solution had been automatically generated by the framework used with the candidate simply making minor edits to some of the methods within the classes created by the framework.

For most candidates whose projects focused on web design and development, the planned use of session variables in the solution continues to be very unclear and, in many cases, contradictory. In addition, 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 structures that were submitted often omitted planned redirects and links indicated elsewhere in the design.

Although many candidates' user interface design has improved, most candidates failed to indicate error messages that would be generated from the system — even when those errors messages had been indicated in the input validation design.

Implementation

Most candidates achieved good marks for the implementation of the constructs and the implemented user interface, with many candidates also gaining good marks for the implementation of integration.

Although a few candidates who chose to implement a database did provide evidence of the initial contents of the implemented table(s) and the data used to populate the table(s) before any queries were executed, most candidates omitted this essential evidence. In addition, some candidates submitted incomplete evidence of the structure of the implemented table(s), and therefore failed to demonstrate that the implementation matched the data dictionary.

Although most candidates who implemented an Advanced Higher search and/or sort algorithm did submit screenshots to show the results of the algorithm performed, many candidates did not provide the 'before' evidence of the data searched or the

unsorted data. Candidates must submit this evidence to demonstrate that the implemented algorithm worked correctly.

Some candidates' submissions lacked any evidence of the ongoing testing that would have been carried out automatically throughout implementation as individual features were coded and added to the solution.

Testing

Most candidates achieved good marks for carrying out all tests listed in their test plan and providing screenshot evidence of those tests. In addition, many candidates provided evaluative descriptions of the results of testing by focusing on aspects of their solution that had worked well as they carried out final testing themselves, adopting the persona and carrying out the test cases indicated earlier in the test plan. Importantly, these evaluative descriptions also highlighted aspects of the solution that perhaps did not function or operate as well as the candidates had originally intended.

All candidates are required to produce a comprehensive test plan prior to final testing of their solution. Although many candidates did include all requirements identified at the analysis in their test plans, in many cases, the planned testing was far from comprehensive. In some test plans, several mandatory aspects of the solution had been combined so that they could be 'tested' using one single test — for example, a single test used to (1) test input validation applied when new member details were entered; (2) test the connection to the database; and (3) test an insert query used to add the new member details to the underlying database. In some cases, the comprehensive test plan contained details of fewer than ten planned tests.

Evaluation

Most candidates who referred to each of the requirements identified at the analysis stage received 1 mark for an accurate description of the fitness for purpose of their solution. In addition, many candidates also took time to provide an evaluative discussion of the results of testing.

Many candidates referred to at least one type of maintenance (adaptive, corrective or perfective) in their description of the maintainability of their solution, with those descriptions explaining how features of their solution would aid or hinder that maintenance activity.

In some cases, descriptions of the robustness of the solution made no reference to the limited input validation that had been implemented, with many candidates failing to acknowledge that the limited nature of their input validation meant that their solution was not robust.

Section 3: preparing candidates for future assessment

Question paper

Centres should ensure that candidates are familiar with the Advanced Higher search and sort algorithms: binary search, bubble sort and insertion sort. Candidate responses do indicate that candidates are increasingly aware of the important distinguishing features of each algorithm, but centres should also ensure that candidates have opportunities to practise applying the algorithms to different Advanced Higher data structures, including 2-D arrays, arrays of records and arrays of objects.

Centres should ensure that candidates are familiar with all the types of testing listed in the course specification: component testing, end-user testing, integrative testing, final testing and usability testing based on prototypes. Candidates should be aware of the specific purpose of each type of testing and should be able to accurately describe how each type of testing is carried out.

Centres should continue to encourage candidates to pay close attention to any UML class diagram, together with any sections of related code that may be presented. Candidates' understanding of how those sections of code relate to the class diagram and the business rules outlined in the stem of the question are likely to be examined at different points in the object-oriented programming question. Centres should also ensure that candidates fully understand the relationship between inheritance and polymorphism, as many candidates are incorrectly using the two terms interchangeably: inheritance is used to establish hierarchical relationships between classes, while polymorphism allows objects of those related classes to be treated uniformly — in essence, inheritance is primarily concerned with structure, whereas polymorphism is concerned with behaviour.

In question 3(b), although many candidates correctly stated that the `calcOrderTotal()` method of the `Delivery` class had been inherited from the

`Order` superclass, only a few candidates were able to identify that the inherited method had be edited to account for the additional delivery charge, which did not apply to objects of the `SitIn` and `Collection` classes. Also, in question 3 (c), since `SitIn`, `Collection` and `Delivery` all inherit from `Order`, objects of the three subclass types can all be referenced by a single array of type `Order`, as polymorphism allows them to be treated uniformly. In question 3 (e)(i) however, although `orderList[1]` is an object of type `SitIn`, at Line 6, the object data is being referenced by an array of type `Order`, and therefore only methods that are accessible to objects of the `Order` class can be applied.

Centres preparing candidates to attempt questions in section 2, 'Database design and development', should ensure that candidates are aware of the SQL operators that are listed in the course specification. Some candidates continue to use comparison operators introduced at National 5 level (<= and >=) rather than the Advanced Higher operator `BETWEEN`, and the Higher logical operator `LIKE` rather than the Advanced Higher `IN` operator.

Centres preparing candidates to attempt questions in section 3, 'Web design and development', should continue to ensure that candidates have opportunities to read and explain unfamiliar pseudocode used to design server-side processes that can be implemented using the PHP mysqli functions listed in the course specification.

Project

Centres must ensure that they are using the correct version of the coursework assessment task. Version 3.0 was issued in August 2025, alongside mandatory templates that candidates must use to gather their evidence. These new templates will be mandatory for all project submissions in session 2025–26, and provide details of the evidence that candidates must submit to fully demonstrate the skills assessed at each stage of the project.

When projects are being selected, centres should advise candidates to be realistic about what can be achieved working independently within the constraints of the course. The course introduces candidates to advanced coding skills in three

separate areas and allows candidates to consider how coding in those separate areas may be combined to form a single, integrated solution. Unlike the National 5 and Higher coursework tasks, which are very prescriptive in nature and are very focused on the content examined at that level, the Advanced Higher project is the first opportunity that candidates have to be creative, and for many, it is the first opportunity for candidates to truly demonstrate the extent of their development skills. However, it is **essential** that centres help candidates to understand that one purpose of the project is to focus on the tasks involved at each stage of the software development process while demonstrating the breadth of their understanding of the **mandatory** content across all three of the Computing Science levels. This means that all candidates should be able to combine input validation skills gained at National 5 level with skills gained at Advanced Higher level that enable them to integrate coding skills from two different areas of the courses.

In addition, candidates tackling a 'Software design and development' project at this level should demonstrate their understanding of the Advanced Higher-standard algorithms by applying procedural programming skills gained at Higher level to one of the Advanced Higher data structures (array of records or a 2-D array), or applying the object-oriented programming skills gained at this level to an array of objects and using object-oriented techniques to search or sort the array data. Candidates tackling a database or web design and development project at this level should be able to demonstrate the querying or web skills gained at lower levels of the course with the querying and web skills gained in the mandatory content of the Advanced Higher course. Centres should advise candidates to consider an agile-type approach to the development of their project: in the first iteration, the requirements focus solely on the mandatory concepts, as well as integration and the generation of all the evidence required for submission of a successful project. If time permits, later iterations of the development would enable candidates to have more freedom, be more creative and add more interesting features to their working solution — with the added bonus of not needing to generate evidence of the later iterations.

Since a mandatory requirement of all Advanced Higher projects is that all inputs to the final solution must be validated, centres should advise candidates that they are expected to apply the input validation skills gained at National 5 level by applying

some form of input validation routine. This could be a length check, a presence check, range check or a restricted choice. The intended input validation of all input to the system should be indicated on the wireframes used to show the planned user interface design. Where candidates opt to use drop-down features of the UI or buttons to implement a restricted choice, this should be clearly described along with all other underlying processes.

When preparing their project evidence for submission, candidates must use one of the four mandatory templates to organise and present their evidence. To make it possible for markers to track requirements at each stage of the development, each template uses page numbering in the footer of the document, and this must **not** be removed prior to submission. In addition, all candidates should add the complete code for their solution to the appendix at the end of the template.

Markers must be able to easily read the code, and they can only award marks if the concepts, integration, and stated requirements can all be identified in the implemented code. When screenshots of code are presented, these must be large enough to read the code and, where possible, use of a black background with coloured text should be avoided. Where screenshots of the user interface are used to support evidence of testing, markers should be able to read input values and output messages in those screenshots without the need to magnify the page.

Analysis

Centres should encourage candidates to use a UML use case diagram early in the analysis stage. Once completed, the UML use case diagram should provide candidates with a high-level understanding of how end users will interact with the system and the tasks that end uses should be able to perform. Those user interactions identified in the UML case diagram should appear as end-user requirements in the requirements specification, which means that there is no need for candidates to carry our user surveys for this purpose.

In the requirements specification, candidates must limit themselves to the stated maximum number of end-user and functional requirements, and the functional requirements **must** include all of the mandatory concepts and integration stated for

each type of project. The new templates contain pre-populated functional requirements for Advanced Higher concepts, integration and space for candidates to include up to a maximum of eight additional functional requirements (including input validation). A penalty will be applied to projects that exceed this. Centres should advise candidates that a functional requirement should not result in multiple sub-processes (for example, several independent queries) at the implementation stage. By reducing the number of requirements, candidates should be better able to track each requirement and ensure that it is treated appropriately at each subsequent stage of the development. In addition, reducing the number of requirements and focusing on the mandatory Advanced Higher concepts, integration and input validation should enable candidates to demonstrate their skills and understanding of the tasks that are involved in each stage of the software development process.

Design

All candidates who set out to produce a procedural programming solution should include a top-level design of the solution that indicates the intended data flow between modules of their solution. Currently, most candidates fail to include this evidence. In addition, these candidates should also provide a design of the Advanced Higher data structure (array of records or 2-D array) that indicates the planned data types and dimensions of any data structure that they intend to implement.

Similarly, rather than presenting a UML class diagram that has been generated automatically by the framework used to implement the user interface and/or game play, all candidates who set out to produce an object-oriented programming solution must present a UML class diagram that indicates the fields and methods of any classes that they intend to implement from scratch. This UML class diagram should clearly indicate the fields and methods that will apply to any objects that become elements of the array of objects indicated at the analysis stage. If the array of objects is shown in a separate class, then that class should indicate the search and/or sort method(s) that will be applied to the array of objects. In addition, although these

candidates do not need a top-level design, they are expected to design each of the methods that they intend to implement from scratch.

Candidates who set out to produce an object-oriented solution should also ensure that they apply the Advanced Higher algorithm to the array of objects they have created.

Implementation

Centres should remind candidates that they must provide evidence to show that any 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, many 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 the code has searched.

Centres should also remind candidates that they must submit evidence to show that the structure of the implemented tables matches the structure indicated in the data dictionary produced at the design stage. This evidence can be in the form of the SQL CREATE code or screenshots of the implemented table showing the necessary structural details. In addition, candidates must also present evidence to show that all queries are fit for purpose. 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) must be presented and clearly labelled to indicate that these data values are in fact the initial values. 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 INSERT code used to populate the tables or screenshots showing the initial values or empty tables. To demonstrate that their queries are working, candidates should also include screenshot evidence to show that each implemented query is fit for purpose:

- evidence for INSERT queries should show additional records in the table
- evidence for UPDATE queries should show that existing records in the table have been edited

- evidence for DELETE queries should show that records have been removed from the table
- evidence for SELECT queries should show that results returned have been retrieved from the underlying table

Throughout the implementation of their solution, candidates are expected to maintain a log of ongoing testing. As they are coding their project solution, candidates should automatically run any new code to check that that it is working correctly, and the log of ongoing testing should reflect this. Candidates should note that ongoing testing has been carried out as the mandatory concepts and integration are added to the solution and, also, when functional requirements are implemented. Candidates should record evidence of ongoing testing in the log, even when they did not encourage any issues. 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 — screenshots of this would also make very good evidence that could be used in the log of ongoing testing or as evidence of final testing. Candidates should add a brief note of any issues encountered as requirements are implemented to the log, 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 make a brief note of this in the log.

Centres who teach Python must ensure that their candidates are aware of the difference between the implementation of an array of records and the implementation of an array of objects — this is especially important when candidates are taught to implement arrays of records at Higher level as a plain or ‘empty’ class that has data fields, and the only method is a constructor. There is evidence in the problem statement and subsequent design work, test evidence and evaluations of candidate submissions that many candidates are confused about whether an array of records or an array of objects has been implemented. The essential difference is that the blueprint for an array of objects is a class definition that includes both data

fields **and** methods that can be applied to manipulate the data held in those fields, whereas an array of records is a plain class with fields and a single method which is the constructor.

Testing

Centres should encourage candidates to consider how to comprehensively test their solution. Their test plans should clearly indicate when each functional requirement is being tested and importantly, the test plan should provide details of how it will be tested. For many candidates, it may be useful to consider details of their test plan as early as the design stage. By doing so, candidates will not get bogged down with or clouded by any implementation issues that may arise later. Instead, they should be able to think clearly about how to gather the evidence needed to demonstrate, for example, that their sort does in fact arrange player and score details in descending order of score, that an insert query correctly add the details of the player and their score to the database or that a session variable with the player's name is passed correctly to the page displaying a personalised welcome message. The test plan should also indicate input values that will be used to comprehensively test all input to the system by indicating normal, extreme and exceptional values that will be used.

Candidates should carry out testing of the end-user requirements of the final solution themselves by adopting the realistic persona of a typical end user of the system described, and carry out each of the test cases that have been listed. Testing of the functional requirements in the final solution should generate screenshots as evidence for each test in the test plan. Where appropriate, these screenshots should indicate the input values stated in the test plan, validation error messages, evidence of breakpoints, underlying database tables, and output showing results of processing.

Appendix: general commentary on grade boundaries

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

For most National Courses, we aim 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, we hold 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 our 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. We 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.

For full details of the approach, please refer to the Awarding and Grading for National Courses Policy.