



A-level **Computer Science**

7517/1 (1A/B/D/E)

Report on the examination

Version: 1.0
June 2025

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

AQA retains the copyright on all its publications. However, registered centres for AQA are permitted to copy material from this booklet for their own internal use, with the following important exception: AQA cannot give permission to centres to photocopy any material that is acknowledged to a third party even for internal use within the centre.

General

Most students were well prepared for this exam and had made good use of the time available between the release of the Preliminary Material and the day of the exam.

The Preliminary Material is released on 1 September and centres should give students access to it soon after that date so students have time to study the material. Centres should make sure they test the Skeleton Program on the computers to be used for the exam to ensure everything will go smoothly in advance of the exam.

Students will not receive marks for screen captures that have not been produced by running their code. There were still students who provided screen captures that were not produced by their code. Their time would have been better spent trying to get their program code to work correctly.

A copy of the Skeleton Program used by the centre should be included alongside the scripts sent to the examiner, whether or not the Skeleton Program was modified. A significant number of centres did not do this. A few centres attached a copy of the Skeleton Program to each student's Electronic Answer Document and sometimes also the exam paper, which is not required.

Some students took screen captures of code rather than copying the code out of the program editor and pasting it as text into the EAD. Whilst this is allowed, it must be discouraged as it makes the code difficult to read, particularly if a dark background colour has been used. Screen captures of code also sometimes resulted in long lines of code only being partially visible when they went beyond the right-hand side of the screen which prevented some students from gaining marks.

Question 1

Question 1 was about data structures and algorithms.

For question part 1.1, just over 50% of students were able to explain why a record in a hash table could normally be found more quickly than in a sorted list. Some students simply stated the time complexity of accessing a record in a hash table which was not sufficient for the mark to be awarded.

Question part 1.2 asked students to describe the steps involved in adding a new record to a hash table. This question part was well answered and 80% of students were able to get some marks, but relatively few obtained full marks. The most common omission was to fail to state what the hash function was applied to.

Most students gave answers to question part 1.3 that showed some understanding of the reason why the time taken to access a record might increase when more records are stored in a hash table but many answers were too vague for the mark to be awarded, frequently stating simply that more collisions occur.

More students were able to explain the reason for the time complexity of the binary search (question part 1.5) than the reason for the time complexity of the linear search (question part 1.4). Students often did not make it clear that the worst case for a linear search would be n comparisons and instead stated that n comparisons were needed. Over half of students were able to state an advantage of a linear search over a binary search (question part 1.6). A similar proportion of students answered question part 1.7 correctly, where the most frequently seen correct answer was that both algorithms had less than exponential time complexity.

Less than 20% of students gave the correct answer of 99 for question part 1.8. The most frequently seen incorrect answer was 10000 (i.e. 100^2), suggesting that students recognised that the algorithm was quadratic time complexity but had not answered the question that was asked.

Question 2

Question 2 asked students to define decomposition.

Nearly 90% of students were able to give a partial definition and obtained one mark for this question. Not many students were able to give a two-mark definition with the idea of “identifiable task” (or equivalent) rarely seen.

Question 3

Question 3 was about the graph data structure.

About three quarters of students could state the purpose of Dijkstra’s algorithm (question part 3.1). Some students did not get the mark as they stated it found the shortest path in a graph, rather than the shortest path between two nodes in a graph.

Most students were able to give two correct answers for question part 3.2, with almost all giving at least one correct answer. Some students believed that all trees had to have a root node, and the most common incorrect answer was to state that there was not a root node.

Question part 3.3 asked students to complete an adjacency matrix. This question was also answered well by most students. The most commonly seen errors were to store a count of how many nodes a node was connected to instead of an indication if there was an edge between two nodes and to indicate that there was an edge from a node to itself for each of the nodes in the given graph.

Almost all students who got the mark for question part 3.4 gave the answer that the start and end nodes were the same. The other possible correct answer was more difficult to describe and was rarely seen.

Question part 3.5 asked students to complete a trace table. It was pleasing to see that most students did attempt this question part. Giving the correct initial values for the contents of `v` and the correct values of `j` were the marks students found easiest to achieve.

Question 4

Most students were able to get some marks on this programming question, with many students producing fully-working code – consistent with achievement on past section B questions.

Many students were able to get about half of the marks on this question by writing program code that got a number and a string from the user, setting up some iteration structures and attempting to remove some characters from the string.

Most students found it easier to remove non-alphabetic characters from the string than to encrypt the string, though this was not always the case and there were some students whose code for the encryption was correct who did not have working code for the removal of the non-alphabetic characters.

Some common errors were to:

- write code to remove only certain, specified, non-alphabetic characters
- have an iteration structure that repeated once too often, normally resulting in an out of bounds error when the program code was executed
- not realise that there are some non-alphabetic characters in the ASCII code range 65 to 122 (for students who used ASCII codes in their attempt to remove non-alphabetic characters).

A wide variety of approaches to this question were seen that allowed students to get full marks.

For the encryption part of the task some students used:

- a two-dimensional list/array to store characters
- remainder division with a one-dimensional list/array
- nested for loops with a string.

For the removal of non-alphabetic characters part of the task some students:

- compared a character with a list containing all of the allowed characters
- used a regular expression to check if a character was allowed
- checked if the ASCII code for a character was in one of the two allowed ranges of ASCII codes
- used a built-in function to test if a character was alphabetic.

Question 5

This question was about the `ConvertToRPN` subroutine.

Most students were able to give at least one advantage of using RPN to represent expressions (question part 5.1). The most frequently given correct answer was that the expressions are simpler for a computer to evaluate. For question part 5.2, most students stated that a dictionary consisted of key-value pairs but only about a third could give the full explanation.

About half of students gave a correct answer for question part 5.3. The most common incorrect answer was list.

Question 6

This question was about the `CheckIfUserInputValid` subroutine.

Almost all students obtained full marks for question part 6.1.

Students found question part 6.2 more challenging. Some students recognised that the FSM would not match with some strings that contained operands with an even number of digits but did not state that this was only for operands other than the first.

For question part 6.3 over half of students could give a partially correct explanation about which strings would be accepted but only about 20% were able to get both marks.

Question part 6.4 was discounted and all students were given the mark for this question part. This was because the question asked students to explain the purpose of the metacharacter `$` in the regular expression in the `CheckIfUserInputValid` subroutine in the Skeleton program. However, the specification does *not* require students to be familiar with this metacharacter. We apologise for this oversight in the question setting.

Question part 6.5 was well-answered. The most common mistake was to confuse the purpose of the `*` metacharacter with the purpose of the `+` metacharacter.

Question part 6.6 asked students to give two properties of a set that are not true for a list. More students recognised that lists were unordered than that they have no duplicate values.

Less than 20% of students were able to explain why a regular expression could not check an infix expression was valid if it was allowed to contain brackets for question part 6.7. A common incorrect answer was that this was because brackets are used in regular expressions, implying that it would not be possible to tell if the bracket was part of the infix expression or not.

For question part 6.8 over half of students were able to write a partially correct BNF rule that covered some of the expressions that it needed to match with but less than 2% of students were able to write a BNF rule that matched with all of the required infix expressions.

Question 7

This question was about the `EvaluateRPN` subroutine.

For question part 7.1 most students incorrectly described an expression that would not be accepted by `CheckIfUserInputValid`. The most frequently seen correct answer was an expression that contained division by zero.

Over half of students got a mark for question 7.2 but explanations were rarely detailed enough for both marks to be awarded.

Question 8

This question was about the vector data structure.

About a third of students correctly identified a variable in the Skeleton Program that could represent a vector. `Targets` was the most frequently seen correct answer.

Question 9

Question 9 required students to write program code that changed the player's score based on two penalty positions in the list of targets. About a third of students obtained full marks on this question.

The most common mistakes made were:

- to use positions 1 and 2 of `Targets`, instead of 0 and 1
- to multiply the value in position 0 by 10, instead of adding 10
- placing the new selection structures in an incorrect place in the code
- writing the code in a way that meant it was not possible for both of the two penalty positions to result in a change in the score in the same turn.

Question 10

Question 10 required students to adapt the Skeleton Program so that expressions that used the exponentiation operator could be used.

Over 80% of students were able to get half of the marks for question part 10.1, but few obtained full marks.

Most students adapted the Skeleton Program so that exponentiation could be used but did not attempt to write code to make exponentiation be right-associative. These students normally got 5 marks for question part 10.1 but were unable to get the mark for question part 10.2.

There were a significant number of students whose attempt at right-associativity was to calculate the second operand to the power of the first operand instead of the first operand to

the power of the second operand. This was not correct but did work correctly for the test students were asked to do for question part 10.2. These students normally only got 4 marks for question part 10.1 but did get the mark for question part 10.2.

Question 11

Question 11 required students to write a new subroutine that would list all the values that appeared multiple times in the list of targets. About a third of students obtained full marks on this question. A wide range of creative approaches were seen in the answers to this question including:

- using a set to hold duplicates found meaning that further instances of that value would not be added
- sorting a copy of the `Targets` list to make finding duplicates easier
- using a dictionary based on the `Targets` list where the key-value pairs are the target and a count of how often that target has been found in the list so far.

Some of the common mistakes seen were:

- to not ignore the -1 values in the list of `Targets`
- writing code that would only work if there were up to three instances of a target in the list
- writing code that would either go beyond the upper bound of the list or not check the last item in the list.

Question 12

Question 12 required students to write a new subroutine that would check if it was possible to get a target using exactly three of the numbers allowed. As was the case with question 11, a wide range of different approaches were seen in working solutions to this question.

About 25% of students did not attempt this question, even though it was possible to get a mark for simply creating the new subroutine.

Some students used 16 selection structures, instead of nested iteration, to obtain two operators. Where this was done accurately, students were awarded the marks but many students who used this approach missed out one or more combinations of the two operators.

Some of the common mistakes seen were:

- to not ignore the -1 values in the list of `Targets`
- to not check that the three positions used in `NumbersAllowed` were different
- to write code that would change the contents of the `Targets` list eg calling `CheckIfUserInputEvaluationIsATarget` with `Targets` rather than a copy of `Targets`.

Mark Ranges and Award of Grades

Grade boundaries and cumulative percentage grades are available on the [Results Statistics](#) page of the AQA Website.