GCSE
COMPUTER SCIENCE

8525/1A-C Computational thinking and problem solving
Report on the Examination

8525/1A-C
June 2023

Version: 1.0

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

Introduction

This year saw the second examination for the 8525 GCSE Computer Science specification, with computer programming being tested in the paper. Most students seemed well prepared for this examination paper and showed a good application of the problem solving and computational thinking skills they have been taught.

Generally, the coding questions were well answered in all three languages. Whilst the indentation grids have been provided for the benefit of all students it is especially important for Python centres to work with their students on their correct use as these are provided to support them in making their indentation clear: in many cases they are not being used as intended and the student's indentation is not clear.

Some students confused the symbols greater than (>) and less than (<) and/or misunderstood the meaning of 'inclusive'.

With programming questions, students should be careful that they are either outputting or returning a value as stated in the question as marks cannot be awarded if this is omitted.

As with the previous specification another key point to take forward from this year's examination paper is that students should always read the question carefully as some student responses did not answer the specific question asked.

General

A small number of students did not attempt the coding questions. Teachers should encourage students to attempt every question as often marks can be gained for straightforward variable assignment and Boolean conditions. There are also design marks available for the coding questions which means that some marks can be gained even if the student's code is incorrect. Students should be encouraged to mark off each item on a list of requirements when they have included it in their answer (these are provided as bullet-points in each question).

AQA has provided resources on their website which show the coding guidance that the examiners use as well as programming and pseudo-code guides to support the teaching of this paper.

Question 1

This question was about algorithms and string handling. It was encouraging to see an improvement in students' responses to these questions, especially concerning concatenation, but only just over 60% of students were able to apply the `length` function correctly.

The definition of an algorithm was well answered. Centres should note however, that the definition given in the specification uses the term 'sequence of instructions', whereas a lot of students are using the term 'set of instructions' which is not specific enough as it does not imply a specific sequence. Although credited at GCSE, it is unlikely to be at A-level so teachers should encourage the use of the correct definition from the specification. Over 75% of students were awarded both marks for this question but nearly 10% did not gain any of the available marks. A common error was to confuse an algorithm with a program and state that a computer would execute the algorithm which is not always the case.

Question 2

This set of questions focused on analysing an algorithm. Question 2.2 asked for the output of an algorithm corresponding to a given input, and less than half of students managed this. This type of question is common on the programming paper and students should be encouraged to work through the algorithms step-by-step to answer such questions. The rest of the questions in the set were successfully answered by over 70% of students.

Question 3

Question 3 looked at the structured approach to programming.

Most students had some idea of what structured programming is but often they could not express this well enough to achieve both marks. Good responses stated an advantage and then expanded on that to explain why. Just over 25% of students gained both marks. Students often stated two different advantages but could not go on to explain either concept further and so did not address the question asked.

Question 4

This question was the first trace table on this paper and the first time that a question has involved the use of format strings (`f""` in Python or `$""` in C# and VB.NET). Teachers are advised to make use of the programming resources that AQA have made available as a lot of students either didn't include the string part of the output or included `{ }` with their outputs. The mark scheme made some allowances for students that traced the algorithm correctly but made errors in the output. Students should also be reminded that when a string is output the `" "` are not necessary in trace tables.

Some students were unclear as to the meaning of the term robust with common incorrect answers to question 4.2 referring to adding comments to code to make it clear or renaming the variables to something more meaningful. Less than 30% of students were able to identify that validation in some form needed to be added to the program.

Question 5

This question involved tracing an algorithm through a flowchart, and this was very well answered with over 75% of the students gaining all three available marks. The most common error was adding `1` to the value for variable `c` rather than adding the other variables together.

Question 6

This question involved completing an algorithm choosing the correct response from a list of 12 words, and it was pleasing to see over 90% of the students gaining at least one of the available marks (most commonly `USERINPUT`), although significantly fewer gained full marks on this question. All the answers to this question were given in the table within the question, yet a small number of students did not attempt the question at all or left blank spaces.

Question 7

This was the first of the language specific questions on the paper that asked the student to write a program. It was pleasing to see that nearly 80% of the students gained four or more of the available six marks on this question. A common error was to miss out the number `6` in the condition for the selection statement. The other common error tended to be in the subtraction of `£5` from the total with students including the `£` sign in a calculation or removing the value from every ticket.

Question 8

Question 8 was testing whether students could describe how a merge sort works and there were some very well written responses with over 80% gaining at least half marks. Those that did well wrote a list of clear steps explaining what happened at each one. They also showed an understanding of the splits happening until the items were in a list of length one and then the merges happening repeatedly until the list was complete and ordered.

Question 9

This was the first time that a record structure has been tested on the new specification and the first time that it has been done using an algorithm. Teachers are again advised to use the resources that are available to them through the AQA website as it was clear that a lot of students were not prepared for this type of question with only 40% gaining the marks on these questions. The specification does state:

> Use a systematic approach to problem solving and algorithm creation representing those algorithms using pseudo-code, program code and flowcharts.

A pseudo-code guide is one of the resources available and students should be prepared to answer questions in each of the three ways.

An answer such as

```
antMan ← Film('Ant-Man', '12A', 2015, True)
```

does not strictly update a record but instead gives it a completely new (record) value, whereas

```
antMan.beingShown ← True
```

does just update the record.

Question 10

Question 10 was a trace table for an algorithm that was stated in the question as having an error, although over 75% of students could identify what the error was fewer than 40% gained more than one of the five available marks on question 10.1. The most commonly obtained mark was for the (i) column going from `0` to `2` but students struggled to output the values correctly in the nested loop. Often students who scored four marks had missed the mark for `count` going all the way to `6`.

Question 11

This was a second algorithm with an identified error in the question. Students found fixing this error in question 11.2 difficult with less than 15% of students being able to correctly change < to a >=. Commonly students identified that they needed to use an = **or** identified that they needed to change the sign from < to > but did not identify both changes.

Question 11.1 was similar to that in question 2.2 but students performed better on this question.

Question 12

Question 12 tested the students' understanding of searching algorithms. Tracing through an list/array often causes problems for students so it was pleasing to see over 50% gaining two of the four available marks and over 30% gaining all four marks in question 12.1. Commonly students would add in extra numbers or list the animals meaning their outputs were incorrect.

In question 12.2, the linear search algorithm was answered well overall but over 10% of students did not attempt the question. Design marks are always available on this type of question which can be awarded even if the syntax is incorrect. There was a mark available for assigning user input to a variable which should be accessible to all students.

Over 50% of students were able to access four of the seven available marks which is an improvement on the previous paper for similar questions. A common mistake was when indexing values in the `fruits` list/array or outputting values within the loop which meant that they were printed multiple times rather than just once. Although the question explicitly stated students were not to use built in search functions, some students did just that when checking if a value was in the list/array.

With question 12.3 only 50% of students could identify that the list needed to be sorted for a binary search with a common misconception being that a list of strings cannot be ordered or the binary search only works on lists of numbers.

Less than 5% of students gained full marks on question 12.4, which again asked students to write in pseudo-code: the comments for question 9 also apply here.

Question 13

It was disappointing to see that nearly 25% of students did not attempt this question which was examining their ability to index two different characters in a string. Those that did answer did well with over 60% gaining at least two of the six available marks and over 35% gaining at least four. Commonly students did not show understanding that the value that had been input as a string so missed quotes around values or tried to do arithmetic operations with the input.

When using Boolean conditions in a statement students often did not check the value against each criterion, for example, in VB.NET:

```
If letter = "A" Or "B" Or "C" Then
```

instead of

```
If letter = "A" Or letter = "B" Or letter = "C" Then
```

Again, these are errors that we have seen before and across all three languages.

On the 8525/1C exam paper an error appeared in the answer space: the code students were asked to extend was slightly different to that in the figure given in the stem of the question. Examiners were briefed and the mark scheme extended to accommodate whichever version of the code students used in their answer to ensure that the marks awarded to students were not adversely affected. The great majority of students responded to the code in the figure, and analysis of student responses indicate no negative impact on student performance on this question.

Question 14

This was a second complete-the-algorithm question but in this question there was no choice of words from which to select. Over 30% of students gained at least one of the available marks which was pleasing to see in a 2D array question. Many responses had at least one part not attempted or placed `i` and `1` in the wrong positions.

Question 15

This was the fourth programming question and students appeared more willing to attempt this one. Nearly 60% of students gained at least five of the available eight marks. The condition for checking whether the bill was paid was handled well. The most common mistake made, as with previous years, was students not using iteration to keep getting user input. This limits the marks they can gain in the question. Another common mistake was missing the output for how much was left to pay.

Across all three programming languages (but most commonly in Python) we are seeing students using the word `OUTPUT` instead of a `print` statement which cannot gain the mark for the programming.

Question 16

Question 16 used a random number generator for the first time in a programming question and it is an area centres should revisit. Some students randomly assigned values to their dice rolls rather than use the code that was provided for them. However, the question was well handled in general with nearly 30% of students gaining at least nine of the available marks.

The selection statements to complete the game were handled well in most cases, but the errors in student answers concerned areas already mentioned. For **Mark D** students were asked to output both dice rolls and this was missed in a lot of responses although it was clearly asked for in the question and shown in the sample output. The iteration (as with question 15) was not included by some students and in cases where it was included, only one of the conditions was used to control the loop.

Mark Ranges and Award of Grades

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