AQA

GCSE

COMPUTER SCIENCE

8525/1A/1B/1C Computational thinking and programming skills
Report on the Examination

8525
June 2022

Version: 1.0

Introduction

This year saw the first examination for the new GCSE Computer Science specification, with computer programming using actual programming languages being examined within the paper for the first time. On the whole students seemed well prepared for this examination paper and showed good application of the problem solving and computational thinking skills they have been taught.

The coding questions were generally well answered, especially considering the difficulties of writing code on paper. This was true for all three languages.

One area that centres should address is the way in which students write down their identifier names. They MUST be written as single strings with no spaces/gaps between words in multi word identifiers (see Question 8 commentary). It may help if students use meaningful single word identifiers. We've been forgiving in this first series, but centres do need to stress the importance of writing identifier names correctly, or marks may not be gained.

Students need to have a better understanding of basic mathematical operators and terms, such as distinguishing the greater than and less than symbols, as well as the meaning of the term 'inclusive'.

With the programming questions (in the same way as the algorithm questions on the previous specification) students should be careful that they are either outputting or returning a value as stated in the question. Marks will not be awarded for outputting a value to be returned or vice-versa.

As with the previous specification, another key point to take from this year's examination paper would be that students read the question carefully - it was clear from some candidate responses that this was not always happening.

Question 1

This first set of questions focused on some of the key programming terms. Although it was multiple choice it was pleasing to see over 80% of the students getting these questions correct, the exception being question 1.4 where relational operators were examined.

Over 50% answered Question 1.6 correctly which was the first of the programming questions and tested the students' understanding of the term definite iteration and their understanding of loop counters.

Question 2

The first part of this question asked students to answer questions on an algorithm and over 85% gained at least one mark. However, the second part was not answered as well with only around 25% of the students identifying that a comparison only has two possible values; the most common incorrect response was that there were 5 possible values.

In the final parts of this question students needed to identify the most suitable data types for certain variables. They should be encouraged to use the actual data types available within their chosen programming language rather than generic terms such as ‘real’, and to ensure that their answers show distinct types.

Question 3
This was the first of the questions where the students were asked to write program code for the answer. About half of the students were able to answer question 3.1 correctly, but a note should be made about concatenation in Python; although in an output statement a comma can be used to output one string directly after another, if you do this when assigning to a variable then it does not perform a concatenation.

Therefore, answers in the following form could not be credited:

```
displayMessage = carReg," is not valid"
```

Question 3.2, which was an alteration to an existing calculation, was answered very well with over 70% gaining the available mark. Arithmetic errors were the biggest mistake made here.

Question 4

As with the previous specification, students are still using efficiency as an answer without qualifying what they mean. Students often said that ‘B has fewer steps’ without adding that this would therefore make it more ’time efficient’ or ‘reduce the number of calculations required'.

One common misunderstanding was that since Program A used an iteration structure it must be more efficient, when in fact the opposite was true in this case.

Question 5

The responses to this question showed that students do not fully understand the difference between syntax and logic errors, as less than 25% of students identified and were able to fix the logic error on line 7, with most students correcting the syntax error on line 6 instead.

Nearly 70% of students were able to identify the data structure in question 5.3 but integer alone was a very common wrong answer.

Question 6

This question is one of the definitions given in the specification but some of the more common wrong answers included removing details from a program or removing unnecessary code. It is important to focus on the problem and not the code.

Question 7

It was pleasing to see that over 50% of the students gained the full five marks on this question. Students often lost a mark under the *‘Maximum 4 marks if any errors’* limitation, because they made their answer more complicated than it needed to be, eg by adding a conditional loop, and while their answer covered all mark points, they subsequently introduced an error that affected the logic of the program. Another common error was to not print out the email address with the message “Match”, even though the word ‘and’ was highlighted in bold within the question.

Question 8

Although over 40% of students gained full marks on this question a significant number of students used the word ‘OUTPUT’ rather than the correct name in their chosen language (`print` in Python, `Console.Write`/`WriteLine` in C# and VB.NET). Students often used overly complex identifier names (such as `numberOfItemsSold` and `numberOfYearsEmployed`) and in the process often introduced errors by inserting gaps in the middle of the identifier name or by spelling mistakes. Students should be encouraged to use the shortest meaningful identifier names they can such as `items` and `years` in this case. Another common error was to use a variable for the bonus, but then to forget to output its value as required by the question.

Question 9

This question tested the use of substrings, an area that has proved problematic on previous papers. It was pleasing to see that over 65% of students were able to identify the substring in question 9.1. However, when it came to the length of the array/list, only around 35% were able to say that there were 2 items in the list.

Where a question asks for the output from a piece of code or an algorithm, or in a trace table, students need to give the exact output. If a string is being output, there should not be quotes around it as these would not be output (unless the code specifically did so).

Question 10

Approximately 5% of the students did not attempt the trace table, which is disappointing to see, but over 45% gained all the marks available. A common mistake was to complete a normal division instead of an integer division. For question 10.3, many answers still gave another letter such as `x` instead of something meaningful to the algorithm. This is something that students need to work on.

Question 10.4 asked about two differences between the way the two subroutines worked and less than 10% of students were able to give the two clear differences with most of them focusing on the conditions themselves. A lot of students also reused the example given or gave a reason such as ‘one uses repeat and one uses while’. Students should be familiar with the concept of conditions at either end of an iterative structure even if they are not supported in the language being used.

Question 11

Question 11 focused on the different elements of subroutines. For question 11.1, 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. Those that did answered it well with over 85% gaining 4 or more of the available marks. The ones most commonly missed were the returning of the value and the calling of the subroutine.

The second part of the question focused on local variables and there was an increase in the understanding from the previous series with around 60% being able to explain what a local variable is.

Over 50% were able to give two or more reasons for using a subroutine which again shows an improvement on previous papers.

Question 12

Question 12 focused on another algorithm and contained the second trace table for this paper.

This was well answered, with a quarter of students showing that they can trace through code to gain full marks and over 70% of students gaining at least one mark which is encouraging. Only around 30% of the students were able to identify that it was a bubble sort with one common misconception being that due to a variable being called `temp` that the algorithm was to do with temperature. Another common incorrect answer was a description of the word 'algorithm'. In question 12.3 it was pleasing that nearly half of the students were able to identify that the algorithm would go out of bounds.

Question 13

Question 13.1 was a straightforward programming question and over 30% of the students gained all the available marks. One area that caused mistakes was the misunderstanding of the word *inclusive,* as students calculated the boundaries incorrectly.

A lot of students used a REPEAT UNTIL structure, when their language does not have one. Also, the use of composite conditions such as `0 < position <= 100` should be discouraged, as some students became confused by the logic, and students were still using them when their chosen language did not support them. Note that in C# this type of statement gives a syntax error, while in VB.NET may not give the intended result: if `position` has the value `-25`, in VB.NET `0 < position <= 100` has the value `True` since `25 < position` is `false`, which is then 'converted' to `0` for the test `0 <= 0`.

Question 13.2 used a list/array to hold the cards and once again showed that students struggle to use indexing correctly. Whilst less than 10% picked up all the available marks, over a quarter managed to get at least four marks, which shows that they understood what the question was asking. A common issue for students was that they did not realise that to count five pairs, they only needed four comparisons. Similar mistakes were made while setting the conditions for the loop, looping either one time too many or one time too few. While it is perfectly possible to solve this question using only while loops (with no 'IF' selection), it's also a lot more difficult and only a few students who went down this route achieved full marks.

Question 14

Question 14 made use of 2D arrays/lists which in previous years' papers had not been answered very well. It was encouraging this year to see that nearly 40% of students were able to complete the algorithm correctly in question 14.1 and over 90% were able to gain at least one mark.

In question 14.2 students were asked, for the first time in this specification, to create a program for a subroutine and pass in a parameter. It was tackled very well with around 15% of students gaining full marks and over 75% gaining at least one mark. Marks were commonly lost by poor indexing methods and for also incrementing variables in the wrong place.

Mistakes were also made when students returned their number of asterisks rather than outputting them.

Final note

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: in question 14.2 for example a mark could be gained for declaring a subroutine and a second for passing in the given parameter. 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. A tip for students might be to tick off each item on the list when they have included it in their answer as the list of requirements is very concise

Mark Ranges and Award of Grades

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