AS LEVEL

COMPUTER SCIENCE

7516/1: Paper 1
Report on the Examination

7516
June 2022

Version: 1.0

General

Most students were well-prepared for the exam, and appeared to demonstrate good time management. There was an evident familiarity with the preliminary material, and most questions relating to this material were meaningfully attempted.

Students will not receive marks for captures of screens that have not been produced by running their code. Students should also make sure, when pasting in screen captures, that they are readable. The height of a capital letter should be at least 2mm when the EAD (Electronic Answer Document) is printed. This is a skill that could be practiced before the exam, using an EAD and a past paper.

When the evidence requested is an amended subroutine, the entire edited subroutine should be added to the EAD. Full credit can often not be awarded if the examiner cannot ascertain the location of any new code. The code should be copied and pasted as text into the EAD, rather than as a screen capture.

When a question requests an identifier of a variable or a subroutine, no other code should be included. If the answer to a question is the subroutine name `DisplayMenu`, an answer of `DisplayMenu()` would not be considered creditworthy. The parentheses are not considered part of the identifier, nor would the data type be considered part of a variable's identifier.

Question 1

Most students were able to secure either four or five marks on the trace table question. Where students secured four marks, the mark that was dropped was the final mark. This required the result of the algorithm to be given.

Question 2

Forty-eight percent of students were able to secure all six marks on this question. The most common misconception was assuming that each event could only be assigned one label, even though the question stated that *Some of the cells in the table may need to be assigned more than one label*. This assumption would result on only one mark being dropped, so most students scored well.

Question 3

This question was completed well by students with the majority of students achieved a high mark. The most common mistake was related to the requirement to generate a random integer between 1 and 6 inclusive. Some students were unable to write the correct syntax for random integer generation. Others wrote syntactically-correct code that produced an integer in the wrong range, such as between 0 and 6 inclusive.

Where students were unfamiliar with the syntax for random integer generation, subsequent marks were still secured. Commonly, variables were used to store integers that were not randomly generated, but these variables were subsequently processed correctly. This enabled students to demonstrate their programming abilities in spite of an early error or omission.

Question 4

The majority of students gained both marks.

Question 5

The majority of students gained both marks.

Question 6

The majority of students gained the mark for question 6.1

A partial understanding was demonstrated by the majority of students on question 6.2. The element of the data structure in question fulfilled multiple roles, but in many cases, only one role was identified. Students should always consider the number of available marks when answering a question.

Question 7

Many students secured both marks for this question, with only twenty-seven percent failing to pick up any marks.

The question required changes to be described that would apply to a single subroutine of the skeleton program. Some students, in describing changes outside this subroutine, dropped a mark. Others described the same change in two different ways and only gained one mark.

Question 8

Most students gained one of the two available marks for this question. Decomposition was often correctly defined as the breaking down of a problem into sub-problems, but with no further insight presented.

Some students incorrectly described decomposition as though applied to a solution rather than to a problem. Definitions of many key terms are available in the specification, including decomposition.

Question 9

Many students failed to secure full marks in question 9.1, with responses that were insufficiently specific rather than incorrect. The most common way in which a mark was dropped was to correctly define definite iteration, and then to present indefinite iteration as the opposite of definite iteration. Where a mark is available for providing a definition, a standalone definition should be provided.

Questions 9.2 and 9.3 required students to identify subroutines that contained examples of definite and indefinite iteration. The majority of students secured both of these marks.

Question 10

Most students secured the first mark of 10.1 by identifying that an exception stops a program from crashing. Many fewer students recognised that exception handling is relevant to run time errors. 'Errors', in isolation, did not demonstrate a sufficient level of understanding.

The majority of students were able to identify a subroutine that performed exception handling, as required by question 10.2. Of those who correctly identified the subroutine, most were able to identify a relevant circumstance that might cause an exception.

The most common reason for dropping a mark on question 10.2 was to give more than just the subroutine's name. For example, `LoadPuzzleFile` was creditworthy, whereas `def LoadPuzzleFile` or `LoadPuzzleFile(PuzzleName, Puzzle)` was not.

Question 11

Many students dropped the mark on question 11.1 by providing an incomplete answer. Only a very small number of answers were incorrect, but many more were incomplete. Stating only that a subroutine is a named section of code was insufficient on its own, and was a commonly-given answer.

The majority of students secured the mark for 11.2.

Question 11.3 was another question for which responses scoring zero were more likely to be incomplete than incorrect. Students were asked to describe rather than state, showing understanding rather than knowledge. As such, 'reduces testing time' was insufficient, whereas 'reduces testing time due to less code to test' secured the mark.

Both marks of question 11.4 were secured by the majority of students. The most common reason for dropped marks was giving more than just the subroutine's name. For example, `ResetDataStructures` was creditworthy, whereas `ResetDataStructures()` was not.

Question 12

This question was the first that required changes to be made to the skeleton code. On the whole, responses to this question were of a good quality, with most students securing most marks.

The most commonly dropped mark related to the requirement for new code to be at an appropriate place within the subroutine. Validation had already been included to check that inputs could be converted to integers. The new code, checking whether those integers related to protected coordinates, should have been added after this validation code.

Question 13

This question required additional validation to be added to the skeleton code. A value should not be permitted in a cell if it already exists in the same row, column, or sub-grid. Most students secured some marks on this question, but only a minority picked up all six. The marks relating to the sub-grid were secured by fewer students than the marks relating to the row and column.

Determining whether a value already existed within the same row or column was a relatively straightforward task. This could be accomplished using two simple loops. The most common way of checking a sub-grid required a combination of the modulo operator and nested loops. Other students opted for a nine-part selection structure, which was equally creditworthy.

Question 14

This question required the addition of 'undo' functionality to the skeleton code. Design marks were applicable to this question, so syntactically incorrect code could be credited. The majority of students secured at least two of the three design marks.

Most students created the subroutine correctly, and were able to make a valid call to it from elsewhere in the code. Beyond this, the majority of students conformed to one of two groups. One group gained no further marks, and the other group secured all or most of the remaining marks.

Marks dropped by more able students included replacement of a number with an empty string rather than a space. Another common error was assuming that the number of 'undo' moves would always be a single digit value. Were the user to enter `-10`, indicating ten 'undo' moves, this would result in one 'undo' and an ignored zero.

Mark Ranges and Award of Grades

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