A-LEVEL
COMPUTER SCIENCE

7517/1: Paper 1
Report on the Examination

7517
June 2019

Version: 1.0

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

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 that they have time to study this material. Centres should make sure they test the Skeleton Program on the computers to be used for the exam and ensure everything is going to go smoothly in advance of the exam; there were a small number of centres contacting AQA on the day of the exam regarding the data files not opening (normally caused by them not being saved in the correct place), how to access the Preliminary Material and what an EAD is.

Students will not receive marks for screen captures that have not been produced by running their code. There were still many 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 working 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 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

Most students were not able to describe any improvements that could be made to the bubble sort algorithm provided. Most answers were either wrong or too vague to be creditworthy. Many answers showed understanding that some passes might not be needed as the list could already be sorted, but did not describe how the algorithm could be adapted to detect that a list was already sorted. Fewer students recognised that after each pass another item in the list is now in the correct place and does not need to be included in subsequent passes.

For question 1.2, a number of students just rephrased the question. They stated that sorting a list does not become an intractable problem when the size of the list is very large rather than explaining why this is the case. About 50% of students were able to get at least one of the two marks available for this question.

The most common correct answer to question 1.3 was to state that heuristics could be used. Expansions to this point often showed limited understanding of what this meant.

Question 2

Question 2 was on Turing machines. Nearly half of students got full marks for the Turing machine trace in question 2.1. Some students did not show the position of the read/write head and could not gain the marks allocated for this. Very few students were able to work out the purpose of the Turing machine or the purpose of the transition from S5 to S6. Question 2.4 asked for an

explanation of what a Universal Turing machine is; this question has been asked before but only around 43% of students were able to get a mark.

Question 3

Approximately 92% of students completed the logic puzzle in question 3.1 correctly; the most common mistake was to include a letter more than once in the table in the EAD.

Questions 3.2-3.4 were about abstraction. More students knew what representational abstraction was than abstraction by generalisation. A significant number of students put the same answer for both 3.2 and 3.3 suggesting that they remembered that abstraction could mean removing unnecessary detail from a problem, but that they did not remember the definitions of the other forms of abstraction. The most common answer for question 3.4 that did not get the mark was to state that the contents of the cells were no longer shown, even though this was stated in the question.

Very few students obtained both marks available for question 3.5. The most common correct answer was that an adjacency matrix is a more appropriate choice for a graph with many edges compared to the number of nodes. Around 40% of students realised that the bottom half of the matrix would be needed for a directed graph; the most common wrong answer was a weighted graph, a topic that appeared on last year's exam paper.

Question 4

Question 4 was about regular expressions and was a topic that students understood well. In question 4.3, some students did not understand that a sequence of characters has a higher order of precedence than the or (`|`) operator and took `1|01+` to mean `(1|0)1+` rather than `1|(01+)`.

Question 5

Most students were able to get some marks on this programming question, with about a third producing fully-working code. Some students wrote programs that did not work correctly when both words contained a particular character but there was more of that character in the first word than in the second word. Other common errors were to make a mistake with the criteria on an iterative structure so that the last character of a word was missed and, for students who removed a character from the second word rather than counting characters, to remove all instances of a character from the word rather than just one when a match was found.

The two most common approaches taken that led to working solutions were:

- to count the number of times each character appeared in both words and check that the count, for all characters, was less than or equal to the count of that character in the second word
- to remove an instance of a character from the second word for each instance of that character in the first word.

A number of creative answers to this question were seen that showed good problem solving and programming skills. Examples included using built-in permutations functions to create a list of all the permutations of the second word and check if the first word appears in that list or not, and creating dictionaries to store counts of characters in the two words.

Question 6

The majority of students were able to state one reason why a binary file might have been chosen but few were able to give two. The most common correct answer was that the file size would be smaller.

Question 7

This question was about the `GetIndexOfItem` subroutine.

Nearly half of the students answered question 7.1 correctly. Some answers talked about the item not being found in the inventory rather than the item not being in the game. About half of the students also answered question 7.2 correctly, although some students just wrote the condition on the selection structure in words and did not explain how it would be known if the search should be completed using the name or the ID.

Two-thirds of students gave the correct answer for question 7.3. The most common correct answer for 7.4 was that the wrong index could be returned if there were two items with the same name, but few answers went on to say and the wrong item was the first one in the list. Other creditworthy ideas were if two items had the same ID and if there was an item with an ID of -1 in the list of items.

In question 7.5 only half of the students were able to write anything worth a mark for how a hash table could be used. The idea of the result of the hashing algorithm giving the position in the array to look at was the most frequently awarded mark. While some students got a mark for talking about a mechanism for checking for collisions, very few got the mark for how it would be known that the item being searched for was not in the array. The most common correct answer for question 7.6 was that there were not that many items in the game so there would be limited benefit to using a hash table.

Question 8

Generally, students were not well-prepared for this topic. It was the first time a question about sets had appeared on Paper 1 and it was clear that many had not revised this topic. Questions 8.1-8.3 involved applying knowledge of sets to the items in the game. Over half of the students got one mark on question 8.1, normally for using intersection with set B. Questions 8.2 and 8.3 were rarely answered correctly. Question 8.4 was a straightforward question about the difference between proper subsets and subsets but not many students were able to describe the difference accurately. A common misconception was that a subset could contain items not in the set it was a subset of, but that a proper subset could not.

Question 9

Many students obtained full marks on this question with over 80% of students getting the marks for questions 9.1, 9.2 and 9.4. Fewer students were able to state the identifier of a user-defined subroutine that used exception handling and sometimes students were unable to clearly express why it is good practice to use local variables.

Question 10

Question 10 was the first of the questions that required students to modify the Skeleton Program. It was a simple question that nearly 90% of students were able to answer correctly. When mistakes were made this was normally because students did not have any understanding of how to generate random numbers, even though there was an example of this in the Skeleton Program. A minority of students generated a random number from an appropriate range but then set up condition(s) on the selection structure in a way that meant there was not an equal chance of each of the two messages being shown.

Question 11

Like question 10, question 11 was normally well-answered with almost all students getting some marks and about 55% obtaining full marks. The most common error made by students was to set the variable used to store the number of items in the inventory to 3 instead of counting the number of items in the inventory, meaning that the code would only work correctly if there were 3 items in the inventory.

Question 12

For question 12, students had to add a new command 'drop' to the game. Over 85% of students got at least one mark for this question with about 15% getting full marks. Some students only completed part of the question and implemented the drop command but not the functionality to deal with fragile items. The most common omission by the more able students was to obtain the index of the item being dropped but not to deal with the situation when -1 was returned by a call to `GetIndexOfItem`, meaning that their code would try to access the -1th position in `Items` if the user tried to drop an item that did not exist in the game.

Students should be encouraged to attempt the longer programming questions as there are some marks available for relatively simple parts of the code. For instance, there was one mark in this question for creating a subroutine called `DropItem`, even if the subroutine did not do anything. 10% of students did not attempt this question but would probably have obtained some marks if they had done so.

Question 13

70% of students got a mark on this question but relatively few obtained full marks. The aspect that caused the most difficulty was sorting the three values rolled by the player, with some answers not working correctly if two of the dice rolls resulted in the same value. A number of students did not read the question carefully and sorted both sets of dice rolls, not just the player's.

Use of statistics

Statistics used in this report may be taken from incomplete processing data. However, this data still gives a true account on how students have performed for each question.

Mark Ranges and Award of Grades

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