AQA

A-LEVEL
COMPUTER SCIENCE

7517/1 Paper 1
Report on the Examination

7517
June 2022

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 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 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 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 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.

For this examination, advance information had been provided to students about the topics that would be assessed in section A of the exam paper (questions 1 to 4). No advance information was provided for sections B, C and D.

Question 1

Question 1 was about the time complexity of algorithms.

Most students were able to recall the time complexity of the three stated algorithms and almost all got at least one correct.

Question 2

Question 2 was about data structures, particularly stacks and queues.

Question part 2.1 was not well answered, with many answers showing some understanding of the process of removing an item from a circular queue but lacking the precision required to get many marks. Common errors were to write about removing from the rear of the queue and to describe a linear queue with all items moving “up one” after deleting an item. Students who did know the correct process sometimes omitted the need to check for an empty queue before attempting deletion. There were a number of students who described a linear queue instead of a circular queue or whose answers removed items that were not at the front of the queue.

Part 2.2 was better answered, though few students could describe three differences. The most frequently seen answer was about static data structures having a fixed size.

Parts 2.3 and 2.4 were answered correctly by the majority of students, slightly fewer understanding the pop operator than the peek operator.

Part 2.5 asked students to describe how a stack could be used to reverse the order of the items in a queue. This was a question that has been asked before and most students were able to at least partially describe the process. The most common omission was not to add the items back into the queue after they are removed from the stack.

Question 3

Question 3 was a logic puzzle that most students were able to get some marks for. Some students simply said that there would be a contradiction when answering parts 3.1 and 3.3:whilst true this was not sufficient detail for marks to be awarded. The most common mistake when answering part 3.3 was to assume that statement 5 could not be true because it said the same as statement 6.

Question 4

Question 4 was about the graph data structure.

Part 4.1 asked students to indicate the statements that were true about Djikstra’s algorithm. Over half of students got all full marks and most students got at least one mark.

Part 4.2 asked for a definition of recursion, a question that has been asked before. Almost all students were able to provide a suitable definition.

Part 4.3 was a short algorithm trace. Only about 20% of students were able to complete the trace successfully. Common errors were to have multiple values being returned and to have additional values for the `Count` variable.

Most students were able to complete the adjacency matrix for part 4.4.

Part 4.5 was a more complex trace than the one for part 4.3 and included the use of recursion. However, the mean percentage mark for this question part was higher than for the shorter trace from part 4.3 as many students were able to get marks on this question for completing the early parts of the trace successfully. This was intended to be one of the more challenging question parts on the paper: about 10% of students were able to get full marks for completing it.

Parts 4.6 to 4.8 assessed student understanding of the algorithms given in question 4. More students were able to recognise that G was a depth-first traversal for part 4.7 than were able to identify the correct answers for parts 4.6 and 4.8. There were some students who managed to get the marks for 4.7 and 4.8 even though they had not completed the trace for part 4.5 correctly.

Question 5

Most students were able to get some marks on this programming question, with about a third 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 iterated through a string and identified all the vowels, even if their attempt at swapping the order of the vowels was limited or non-existent.

Some students wrote programs that had an iteration structure that omitted checking the last character in the string entered by the user, meaning that their code did not work correctly if the last character was a vowel.

Other common errors were to combine multiple conditions incorrectly when checking for a vowel, writing code that would not work if there were no vowels or only one vowel in the string and trying to access a position in an array/list/string that did not exist.

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

- to go through the string and add vowels found in the string to a list, reverse the order of the list and then go through the string again replacing each vowel with an item from the list of vowels

- to go through the string adding consonants to a new string until a vowel was found, then to go from the end of the string backwards until a vowel was found and concatenate that vowel with the string up to the vowel found; this process then repeated (skipping over vowels already processed until the end of the string was reached.

A number of creative answers to this question were seen that showed good problem solving and programming skills. Examples included adding vowels to a stack as they were found in the string and solutions that had a recursive subroutine that called itself with the substring between the first and last vowels as a parameter

Question 6

Question 6 was about object-oriented programming in the skeleton program.

Part 6.1 was answered correctly by most students. The most common wrong answer was aggregation. Part 6.2 was also answered well, with `Breakthrough` being the most common wrong answer.

For part 6.3, a significant number of students described the *private* specifier instead of the *protected* specifier. Answers showed that more students understood the *public* specifier than the *protected* specifier. A common incorrect answer was to talk about the permissions people would have to modify parts of the code.

Answers for part 6.4 often lacked precision and showed limited understanding of overriding. To gain the mark answers needed to make clear the link to inheritance.

Question 7

Question 7 was about the `CardCollection` class in the skeleton program.

Part 7.1 was well-answered with most students explaining that one of the cards would be duplicated/lost.

Answers to part 7.2 often stated that all values in a set have to be unique. This is a property of a set but was not relevant to the list in a card collection as each card is unique. Good answers made it clear that a set could not be used as it would not allow there to be an order to the cards.

The most commonly achieved mark point for 7.3 was that the result of the hashing algorithm indicated the location that the card should be stored at. The mark points about dealing with clashes/collisions and applying the hash algorithm to the card number (because it is unique) were seen less frequently. Good answers for part 7.4 made it clear that a hash table would be suitable as it would provide direct access to the needed data in constant time.

Question 8

Question 8 required students to demonstrate understanding of a few different parts of the skeleton program.

Parts 8.1, 8.2 and 8.4 were answered well, but fewer students were able to state the actual data value for part 8.3. Some of the identifiers suggested for part 8.2 did not get a mark as it was not clear that they were about the hand e.g. `NoOfCards`. Some responses to part 8.3 gave generic answers about the contents of a stack frame and did not apply to the `SetupGame` subroutine.

Question 9

Question 9 was about using files in the skeleton program. The majority of students gave the correct answer.

Question 10

This question was about regular expressions. Not many students were able to write an accurate regular expression, with most not using the correct symbol for alternation (|).

Question 11

Question 11 required students to write a validation routine to check that data entered by the user met specified criteria. It was the first of the questions that required students to modify the skeleton program.

Almost all students got some marks for this question with over half getting full marks. The most common mistakes were to incorrectly combine the two conditions or to not include an error message. While there were some students who got full marks for developing a correctly working solution that used recursion instead of iteration, there were not many. Most recursive solutions did not do anything with the value returned by the recursive call meaning that the recursion would terminate but the game would no longer work and the player's hand would not be updated following their choice of `D` or `P`.

There were some students who included screen captures not produced by their code. These are never given marks, and their time would have been better spent trying to get their program code to work correctly.

Question 12

For question 12 students had to calculate and display some information about the contents of the deck.

About 40% of students were able to get full marks for this question and most students got some marks. Some students lost marks by not being careful with the conditions they used when looking for a tool card; examples included counting one of the types of tools twice instead of counting each of the three types of tool, combining multiple conditions incorrectly and having an iteration structure that missed out the last card in the deck.

Some students found alternative ways to count the number of tool cards including finding the cards that had a score greater than 0 or checking if the card had a `ToolType` or `CardType` property.

Some students had overly cropped their screen capture for part 12.2 and so did not provide all the evidence needed to be awarded the mark for testing despite having fully-working program code.

Question 13

For question 13 students had to implement a one-use blasting cap that could be used to complete a challenge on the current lock. Most students were able to get some marks on this question with about 15% getting full marks.

Good answers recognised that one would need to be subtracted from the value entered by the user to indicate the position to use the blast cap on.

The most common mistakes made by students who made a reasonable attempt to answer this question were

- giving the variable used to keep track of the blasting cap's use its initial value in a place that meant it would reset after each turn
- having a condition that compared the position entered with the value one less than the number of challenges instead of with the number of challenges and so changing the status of a different challenge to the one specified by the user.

Some students did not get the mark for part 13.2 as they only completed part of the test, most frequently omitting the attempt to use a blasting cap for a second time.

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 variable and giving it an initial value in an appropriate place. Over 15% of students did not attempt this question but would probably have obtained some marks if they had done so.

Question 14

For question 14 students had to create a new type of card – a trap card. Most students were able to get some marks on this question though very few were able to get full marks.

Most good answers seen either created a list containing the indexes of the challenges that had been met or checked if there was at least one met challenge and then repeatedly selected a random challenge until a met challenge was selected.

The most common mistakes made by students who made a reasonable attempt to answer this question were to:

- inherit from Card rather than DifficultyCard
- write code that could sometimes select an unmet challenge
- write code that changed (or could change) the status of multiple challenges
- omit one of the changes needed in GetCardFromDeck.

Again, students should be encouraged to attempt the longer programming questions as there are some marks available for relatively simple parts of the code.

Mark Ranges and Award of Grades

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