A LEVEL
Computer Science
7517/1
Report on the Examination

7517
June 2018

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.

Last year, section B (the unseen programming question) was found challenging by a significant number of students; this year students were better prepared for this type of question.

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, though fewer than in the previous year. Their time would have been better spent trying to get their program code working correctly.

A copy of the Skeleton Program used by the school/college should be included alongside the scripts sent to the examiner whether or not the Skeleton Program was modified. A significant number of school/colleges did not to do this. A few centres attached a copy of the Skeleton Program to each student 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 lost some students marks.

Question 1
Only about 20% of students were able to solve the logic puzzle and obtain both marks, with a significant number of students removing multiple vegetables rather than just one as specified in the question. Some students correctly stated that they would take a vegetable from the box labelled as onions and carrots but then gave an example rather than a general explanation or an explanation of both possible results, e.g. explained what it would mean if the vegetable was a carrot but not what it would mean if it was an onion.

Question 2
Question 2 was about reverse Polish notation (RPN) and stacks. Almost all students were able to convert the simple infix expression into RPN for question 2.1 but less than a fifth were able to get any marks for the more complex conversion in question 2.2. This suggests that students are familiar with RPN but have had limited practice with it.

For question 2.3 the majority of students were able to state one advantage of RPN with no need for brackets being by far the most common correct answer.

Question 2.4 required students to combine their knowledge of RPN and stacks. A number of clearly-written accurate explanations were seen. Some answers were about converting infix to RPN instead of evaluating an RPN expression; another common error was to use multiple stacks even though the question specified that just one stack should be used. Some students did not fully understand how a stack operates and wrote about accessing items that were not at the top of the stack.

Question 2.5 was about the contents of a stack frame. Sometimes answers were too vague to be awarded a mark, “values” was a commonly-seen answer that was too imprecise to be creditworthy.

Question 3
In previous years there have been questions asking students to complete an adjacency matrix based on a diagram of a graph and most students were able to answer question 3.1 this year. This was the first time that an adjacency matrix for a weighted graph had been asked for and some students had clearly not seen this type of question before and only included an indicator that there was an edge between two nodes rather than the weight of the edge between the two nodes; this meant they only got one of the two marks available for this question.

Questions 3.2-3.4 were about graph theory. Question 3.3 was well-answered with students identifying that it was not a tree because there were cycles. The most common incorrect answer was to say that it wasn't a tree because the edges have weights associated with them. Question 3.4 was also well-answered. Answers to 3.2 often showed that students were not as familiar with adjacency lists as they are with adjacency matrices.

For question 3.5 students had to complete a trace of Djikstra's Algorithm. This topic was not on the previous A-level specification and was often poorly answered suggesting many students had not tried to complete a trace for the algorithm before. For question 3.6 many students gave an answer that explained the point of Djikstra's Algorithm (find the shortest route from a node to each other node) rather than what the specific output in the algorithm given in the question would be (the distance of the shortest route from node 1 to node 6).

Question 4
The Halting Problem and Turing Machines were the focus of question 4. Of the four questions in Section A this was the one that students had the least understanding of. A number of answers suggested that students had seen a simulation of a Turing Machine but hadn't fully understood what they had seen. Common wrong answers were that a Turing Machine would need a motor or scroller to move the tape and that a UTM is a Turing Machine that can understand any language. A number of answers talked about the Halting problem being intractable rather than non-computable.

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 only worked for a very limited selection of numbers but showed good exam technique by including their answer even though they knew it did not fully answer the question. A common error was to write the code in such a way that the number 1 was counted as being a prime number.

Question 6
An error on the paper meant that it was not possible for students using the Java programming language to provide an answer for question 6.1. All students (for all languages) were awarded this mark irrespective of what they wrote for their answer.

Question 6.2 asked students to identify a local variable in a method in the `QueueOfTiles` class. There were a number of potential correct answers with `Item` being the most commonly seen. Some students gave an example of a private attribute belonging to the class rather than a local variable in a method in the class.

Questions 6.3-6.5 were about circular and linear queues. Some students stated that a rear pointer would be needed for a circular queue which is true but does not answer question 6.5 as the rear pointer was already present in the Skeleton Program. Some answers for 6.3 talked, incorrectly, about circular queues being a dynamic data structure and linear queues as being a static data

structure. Good answers for 6.4 made it clear that with the queue only being very small in size the overhead of moving all items in the queue up one after deleting an item was negligible.

Most answers for question 6.6 and 6.7 showed some understanding of suitable approaches that could be taken but were rarely precise enough for full marks to be awarded. Some students gave answers for question 6.6 that changed the values of some of the tiles despite the question stating that this should not be done.

Question 7
Question 7 was the first of the questions that required modifying the Skeleton Program. It was a simple question that over 80% of students were able to answer correctly. When mistakes were made this was normally because tiles other than just J and X were also changed to be worth 4 points.

Question 8
Like question 7, question 8 was normally well-answered with almost all student getting some marks and about 75% obtaining full marks. Where students didn't get full marks this was normally due to the conditions on the loop being incorrect which prevented the values of 1 and/or 20 from being valid.

Question 9
For question 9 students had to replace the linear search algorithm used to check if a word is in the list of allowed words with a binary search algorithm. An example of how a binary search algorithm works was included on the question paper but if a similar question is asked in the future that may not be done. A mixture of iterative and recursive solutions were seen. The most common error made by students who didn't get full marks but made a good attempt at answering the question was to miss out the condition that terminates the loop if it is now known that the word is **not** in the list.

Question 10
Students found question 10 easier than questions 9 and 11. Better answers made good use of iteration and arrays/lists, less efficient answers which used 26 variables to store the different letter counts could also get full marks. Some students added code in their new subroutine to read the contents of the text file rather than pass the list as a parameter to the subroutine; this was not necessary but was not penalised.

Question 11
Question 11 asked students to create a recursive subroutine. If students answered the question without using recursion they could still get 9 out of the 12 marks available.

It was disappointing that many students did not include any evidence of their attempt to answer the question. Good exam technique would be to include some program code that answers some part or parts of the question. For instance, in question 11 students could get marks for creating a subroutine with the specified name and calling that subroutine – even if the subroutine didn't do anything. There are many examples of subroutines and subroutine calls in the Skeleton Program that students could have used to help them obtain some marks on this question.

A number of very well-written subroutines were seen that made appropriate use of recursion and string handling. Some good recursive answers did not get full marks because they did not include a check that the word/prefix passed as a parameter was valid before the tile points included in the word were used to modify the score, this meant that all prefixes would be included in the score and not just the valid prefixes. Another frequent mistake came when students wrote their own code to calculate the score for a prefix rather than use the existing subroutine included in the Skeleton

Program that calculated the score for a word – if done correctly full marks could be obtained by doing this but a number of students made mistakes when writing their own score-calculating code.

Mark Ranges and Award of Grades

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

Converting Marks into UMS marks

Convert raw marks into Uniform Mark Scale (UMS) marks by using the link below.

UMS conversion calculator