A-LEVEL
COMPUTER SCIENCE

7517/2: Paper 2
Report on the Examination

7517
June 2019

Version: 1.0

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

Question 1.1

Almost 95% of students correctly identified that a virus checker is an example of utility software.

Question 1.2

For this question part students were required to describe two different types of resource management performed by an operating system. Just over half of the students achieved some marks. The most common reasons for responses not being worthy of marks were that they described a function of the operating system that was not related to resource management or that they simply named a resource management role such as 'memory management' but did not describe what this role was.

Question 2.1

This question was tackled well with the majority of students being able to name two relevant protocols. Around 20% of students described them well enough to achieve all four marks. The most commonly referenced protocols were POP3, SMTP, IMAP and SSH. A small number of responses covered protocols that were not relevant, eg FTP, or that did not operate at the application layer, eg TCP. Responses which stated that POP3 was used to receive email were generally accepted, but it is worth noting that it is more accurate to state that POP3 is used by a client to retrieve email from a server; it is not actively involved in a send/receive process.

Question 2.2

A third of the students correctly identified that the transport layer would look at the port number in the packet to determine which application layer software the data should be passed to. Some responses referred to looking at the header, considering the type of data or considering the protocol used, all of which were not enough.

Question 2.3

For this question part students were required to state the function of the network layer of the TCP/IP stack. The most commonly stated function was that it would add source and destination IP addresses to a packet. It was not enough to state that the network layer added IP addresses to a packet; there needed to be some explanation of what these IP addresses were ie those of the source and destination computers. The second most common response referred to the routing role of the layer. To achieve this mark, students needed to either use the term itself or convey what it meant, such as that at each hop on a route the layer would determine the next hop to take. It was not sufficient to just state that the layer sent the packet to its destination.

Question 3.1

Responses to this question part were often superficial. To achieve a mark it was necessary to explain that with lossless compression the original data could be fully recovered when the data was decompressed, or to identify that with lossy compression some of the original data could not be recovered. Good explanations of how a lossy system might decide which data to remove were also credited.

Question 3.2

Nearly two thirds of students showed some understanding of how dictionary compression worked but they often failed to achieve both marks because their explanations were not sufficiently clear. To achieve both marks, it was necessary to state that a dictionary was built that mapped keys to words from the text and that the words in the text would be replaced with these keys. A minority of students described run-length encoding instead of dictionary compression.

Question 3.3

This question was very well answered, with good responses recognising that a separate dictionary might need to be sent in addition to the compressed data or that the amount of repetition in a small amount of text was minimal and so the potential to compress the data was limited.

Question 4

This question part was extremely well answered, with nearly 90% of students correctly identifying the error in the addition.

Question 5

This extended response question was tackled well: around 60% of students achieved at least half marks. Many students clearly knew the areas of the specification that it covered in detail and were able to write extensively about compilers and interpreters and the fetch-execute cycle. The differences between a compiler and interpreter were well understood. Some students went on to describe advantages and disadvantages of the different types of translator, which was not required, as the question focused on how they worked. The most common misconception was that only an interpreter could locate an error as it would stop when it found it; a compiler is also able to identify and locate errors. It is also worth noting that an interpreter does not generate machine code output. Instead, as it reads and executes a command it calls code within its own program to carry out each command. Many excellent descriptions of the fetch-execute cycles were seen, which covered all three stages of the cycle. Common mistakes were to attribute active roles to passive components, such as the address bus moving to the memory or the Current Instruction Register decoding an instruction.

Question 6.1

For this question part students had to draw an entity-relationship diagram for the database. Most responses achieved some marks, but only around a fifth achieved full marks. The most frequently awarded marks were for the creation of the Appointment and Customer entities and their relationships. Students often failed to include the PetOwner entity which was necessary to link the Customer and Pet entities, as many-to-many relationships cannot be directly modelled in a relational database.

Question 6.2

For this question part students had to identify the relations and attributes that were required to complete the database. Most students achieved some marks with two thirds achieving half marks but only around 10% achieved all four marks. The most frequently achieved marks were for creating the Customer and Appointment entities with the correct attributes. Common errors were to fail to create the PetOwner entity and to miss the Time attribute from the Appointment entity, storing only the Date.

Question 6.3

Most students recognised that speech marks were missing around `Torquay` in the SQL query. Many, but fewer, were able to identify that the condition `Surgery.SurgeryName = Vet.SurgeryName`, which was required to join the two tables, was missing. The most common error made by students was to state that the table names needed to be included before the fieldnames. This was not the case for the fields given which existed in only one of the referenced tables.

Question 6.4

Approximately half of the students had some understanding of the issues around concurrent access and were able to achieve some marks. Good responses recognised that the issue arose when two edits occurred simultaneously, that one update could be overwritten and that which this was would depend upon the order in which the data were saved back. Some answers were given in the context of the veterinary practice, which was acceptable. Many good responses failed to achieve full marks because they were not sufficiently well expressed, for example referring to just data being lost rather than an update being overwritten, or did not make clear enough that two edits were occurring simultaneously – stating only that one edit started after another.

Question 6.5

For this question part students were required to explain how either record locks or timestamp ordering worked to control concurrent access. Students who chose to explain record locks generally performed better than those who chose to explain timestamp ordering. Good responses about record locks recognised that when an edit started on a record a lock would be applied to it and other users would not be able to access or edit the record until the first edit was complete. Some students lost marks by stating the purpose of a record lock rather than explaining how one worked, or by referring to the database being locked rather than a specific record. A minority of students believed that record locks were a security mechanism.

Good responses about timestamp ordering recognised that each transaction would have a timestamp and that, for each record, read and write timestamps would be stored. The database system would then apply a set of rules to determine whether it was safe to process a transaction or not. Many students wrote about transactions being processed in the order of when they were made, which was not sufficient for a mark.

Question 7.1

This question part was not well answered, with only a third of students recognising that the co-domain of the function was $\mathbb{Z}$. A frequently seen incorrect response was $\mathbb{Z}$-1.

Question 7.2

For this question part students had to calculate the result of the composition of two functions. Almost half were able to do this correctly. The most common mistake made was to apply the functions in the wrong order, resulting in a value of 8 instead of the correct answer of 4.

Question 7.3

Partial application is a challenging topic and the responses seen reflected this fact. Good responses recognised that the function would be applied to one of the arguments, which would result in a new function being created which would then be applied to the remaining argument.

Question 8.1

Just over 80% of students were able to correctly complete the two truth tables. The most common mistakes were to write out the truth table for XOR instead of OR and NOR instead of NAND.

Question 8.2

This question was very well tackled, with over two thirds of students achieving all four marks despite the expression being a more complex one than has appeared on previous papers. Some students attempted to simplify the expression before drawing a circuit for it, which was not required. If a student produced a logically equivalent Boolean expression then marks were awarded for a circuit being drawn for that expression, but many students who did this produced an expression that was not equivalent to the one on the question paper.

Question 8.3

The majority of students had some understanding of why the given identity held, but responses often failed to achieve marks due to imprecise statement such as 'to output a 1, both inputs to an AND gate must be the same' – which fails to consider the case of both inputs being 0, or 'for an AND gate both inputs must be 1' – without explaining why both inputs have to be 1, ie for the gate to output 1.

Question 8.4

Most students made a reasonable attempt at simplifying the Boolean expression and achieved some marks, but only a fifth produced a fully correct solution. The most common mistakes were to cancel NOT bars that extended over different parts of the expression and to fail to introduce required brackets when applying De Morgan's law, resulting in an incorrect expression as a result of the precedence of operators:

$\overline{\overline{\overline{B} \,.\, A} \,.\, \overline{B}} = \overline{B}.A.B$ - incorrect cancelling of NOT bars

$\overline{\overline{\overline{B} \,.\, A} \,.\, \overline{B}} = \overline{B + \overline{A} \,.\, \overline{B}}$ - incorrect application of De Morgan as AND higher precedence than OR.

Students should ensure that they show all of their working. This enables the awarding of working marks if the final answer is incorrect. Also, some responses were seen in which students attempted to complete more than one simplification at once and failed to gain marks because they made a mistake through attempting to do too much simultaneously.

Question 9.1

Students showed some understanding of the difference between synchronous and asynchronous communication. Effective responses recognised that devices involved in synchronous communication might use a shared clock to synchronise the transmitter and receiver, whilst in asynchronous communication the clocks of the sender and receiver would be temporarily

synchronised at the start of a communication. Some students wrote about the use of start and stop bits, which was not sufficient for a mark unless it was explained that the start bit was used to synchronise the clocks temporarily. Significant mistakes included confusion with full duplex communication, and symmetric and asymmetric encryption.

Question 9.2

This question was well tackled, with over 90% of students achieving some marks for completing some of the stop and start bits, ASCII code and parity bit. Common mistakes were setting both the start and stop bits to 0, storing the binary value of the integer 4 instead of the ASCII code for '4', and writing the ASCII code in the wrong place in the boxes (often assuming the start bit was the least significant bit of the data value).

Question 9.3

There were some excellent responses to this question part, with the majority of students achieving at least three of the four available marks. Effective responses recognised that Unicode would allow the representation of many more characters than ASCII or explained how this would be beneficial, for example by representing multiple languages in a single character set, but that this would be at the expense of characters taking more bits to store. Responses about majority voting were generally worthy of marks, but could have included more detail. Students often wrote about the improved reliability achieved by replacing a parity bit with a majority voting system, but didn't make the more specific points that a majority voting system would allow some errors to be corrected and could detect multi-bit errors.

Question 10

Most students recognised the structure that the program needed to have, but more mistakes were seen when writing assembly code than in previous years, in both the use of syntax and the logic of the program. Just under half of the students got at least four of the eight marks.

Common syntactical errors were:

- using memory addresses directly in commands instead of registers eg `CMP 102,103`
- using invalid register numbers, or confusing register numbers and memory addresses eg `R103`
- using the command B to identify the target of a branch rather than to make a branch eg using `B loop` as a label
- attempting to combine a comparison and branch into one command eg `BR1EQR2 loop`.

Common logical errors were:

- not checking if the two values were already equal at the start, meaning that at least one subtraction always occurred
- using branching to distinguish between A>B and B>A but failing to use a branch instruction to skip the execution of the second alternative after the first one
- failing to store the result of the program back into memory location `104` at all exit points from the code.

It is important that students are familiar with the instruction set used for questions and have used it to write programs in preparation for the exam.

Question 11.1

The majority of students achieved either one or two marks for this question part. The most common error was to give the mantissa as 0.0000001 which would produce a smaller number, but that number would not be normalised.

Question 11.2

This question part was reasonably well tackled with almost half of students getting full marks for correctly converting from floating point to decimal. Common errors were to treat the first bit as having the positive value +1 instead of -1 and to convert the number from negative to positive, giving +2.4375 as the answer instead of - 2.4375.

Question 11.3

Approximately half of students were able to correctly represent the given decimal number in floating point and 85% achieved some of the available three marks. The most common errors were to give the exponent as +2 instead of -2 or to fail to normalise the answer. Students should make sure that they show their working so that they can achieve some marks even if they have made a mistake.

Question 11.4

Two thirds of students correctly identified that overflow had occurred and that increasing the number of bits used to represent the exponent would resolve the problem. The most common mistake was to state that additional bits were needed in the mantissa. Some students incorrectly referred to stack overflow which is a different issue.

Question 12.1

For this question part, students were required to calculate how much memory a computer system could access. The question stated that each memory location stored one byte of data and the address bus had 32 bits, so the correct calculation to perform was 2^{32}. A common error was to perform the correct calculation but then divide by 8 to convert from bits to bytes, when in fact the answer was already in bytes.

Question 12.2

In this question part students were asked to explain how a wider data bus could speed up the execution of programs. Markworthy responses recognised that this would be because more data could be fetched simultaneously. Students who just wrote that more data could be fetched or that accessing data would be quicker did not receive the mark.

Question 13

Approximately 90% of students achieved some marks for this question part about the relative merits of bitmap and vector graphics, but few achieved full marks. The most common correct responses related to the scalability of vector graphics without pixelisation and the relatively small memory requirements for vector image files. Other good points related to the difficulty of storing images made up of complex patterns of colour, which could not be computed, as vectors. They also noted the difficulty of representing images from real-life which were not composed of shapes as vectors. Marks were not awarded for the simple statement that a bitmap could use more colours

or that a vector graphic could not represent complex images – highly complex architectural plans are usually stored using vector methods. Quite a lot of students stated that bitmaps could be compressed more than vector images. This is a true statement in most instances, but it is not an advantage of bitmaps; the reason that they can be compressed more is because vector images are already stored in a highly efficient and compact form with little redundancy so do not need to be compressed.

The most commonly cited examples were of a logo for a vector image as it would need to be scaled for different uses and a photograph for a bitmap as it was not constructed in such a way that facilitated representation as a vector.

Question 14.1

This question part was about the Vernam cipher. Nearly 90% of students correctly converted the ciphertext and key to binary and so achieved at least one mark. The majority then knew that the bit patterns needed to be exclusive-ored to compute the plaintext. Common mistakes were to use OR instead of XOR and to either add or subtract the bit patterns from each other.

Question 14.2

Many students seemed to have some understanding of what computational security was but only a quarter were able to express this clearly. Commonly seen incorrect statements were that a computationally secure cipher could never be cracked or could only be cracked by brute force methods. Good responses identified that such a cipher could theoretically be cracked, but that this could not be done in polynomial time so this was not a practical thing to do. Some students failed to achieve a mark because they wrote about decrypting or solving the cipher rather than cracking it.

Question 14.3

For this question part students needed to explain what the key exchange problem was. Good explanations identified that the key would need to be passed from the sender to the receiver and that this could be intercepted during the transmission, which would mean the encrypted message could be deciphered. Over 60% of students achieved at least one of the two available marks. Students often lost marks for imprecise responses or by not directly addressing the question, for example by just stating that if the key were intercepted the message could be deciphered, without linking this to the fact that the key would have to be transmitted in this case so could be intercepted.

Question 14.4

For this question part students needed to explain how a receiving computer would decrypt a message that had been encrypted using asymmetric public and private key encryption, and how the digital signature could be used to verify the identity of the sender. The question part was not well tackled; over half of students achieved at least one mark, but just over 10% achieved all four marks.

More effective responses identified that the receiver would decrypt the combined message and digital signature with the receiver’s private key and the digital signature would then be decrypted using the sender’s public key. The same hashing algorithm would then be applied to the received decrypted message as was applied when the message was transmitted. If the resulting hash

matched the decrypted version of the digital signature (or message digest) then the identity of the sender could be verified.

Common mistakes were to write about public and private keys but not associate these with the sender or receiver, or to be unclear about the calculation of the new hash or what this would be compared to – for example stating that the new hash would be compared to the digital signature, which would not work as the digital signature is encrypted whilst the new hash is not. Responses often included some correct terminology but did not demonstrate that students understood the process.

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.