COMPUTER SCIENCE

Paper 9618/11
Theory Fundamentals

Key messages

Candidates must ensure that they read the question very carefully. Sometimes specific answers are excluded. For example, if a question says, '*Memory management is one operating system task, describe two other operating system tasks*' no marks will be awarded for an answer which describes memory management. Similarly, if the question includes an example of something such as a password, no marks will be awarded for answers that describe something with a different name that operates in the same way such as a PIN code.

In general, two marks will not be awarded for two statements which mean the same thing but which are worded differently. Candidates should be aware of the need to make each point in their answer sufficiently different.

At this level of study it is expected that candidates will respond with more than just general knowledge. Appropriate correct use of the technical terminology (jargon) is expected. Some questions require the application of knowledge and the demonstration of a deeper understanding. These questions usually ask '*how*' or '*why*' and in these situations it is not enough to simply describe a generalised situation. Answers need to be applied to the specific situation outlined in the question.

General comments

The handwriting of some candidates continues to be a cause for concern. If an examiner cannot read what is written as an answer then no marks can be awarded.

Candidates should be aware that when there are numbered spaces or boxes given on the examination paper, only the first answer in each space will be assessed. This also applies if a question asks for a certain number of examples. For example, if a question asks for two situations where something happens, only the first two candidate answers will be considered.

While planning answers is to be encouraged, especially when extended prose is required, candidates should write their plans either on the blank pages in the examination paper or on a separate sheet of paper and cross them out clearly. Writing the answer in pencil and then over-writing it in ink should be avoided. Even when a candidate thinks that it has been completely erased, the pencil image is picked up during the scanning process and the result is a double image which can not only be very difficult for examiners to read but can also lead to confusion in numeric answers.

Comments on specific questions

Question 1

(a) (i) Many candidates were able to correctly convert the binary number directly to hexadecimal. Some candidates who attempted to convert the binary number to denary first, then from denary to hexadecimal, gave incorrect answers because of errors made in the arithmetic calculations involved.

(ii) There were many correct answers to this question. A common incorrect answer was 0110 1100 where candidates had converted the given denary number to 8-bit binary instead of to Binary Coded Decimal.

(iii) Several different methods of conversion were demonstrated in answers to this question, all of which should have led to the correct answer. However, there were also some common mistakes. A number of candidates using the direct conversion method omitted the minus sign on the most significant place value and gave an incorrect answer of 4028. Candidates who used the method of finding the complement often showed the correct unsigned value of 68 but then omitted to include the minus sign.

(b) (i) This question was answered well. Some candidates should understand that when asked to perform binary addition and/or subtraction, converting the values to denary, performing a denary calculation and then converting the answer back to binary will not be awarded any marks

(ii) Almost all candidates could correctly name the error. Describing it proved to be more challenging. Vague answers about space being too small are not precise enough at this level of study. The description would be expected to include at least a statement referring to the number of bits.

Question 2

(a) This E-R diagram represented a standard breakdown of a many-to-many relationship into two one-to-many relationships using a joining table. While there were some good correct responses, many of the diagrams showed the relationships going the wrong way. Some candidates need to ensure that they use a recognised convention for showing the relationships on an E-R diagram.

(b) This question carried two marks and began with the command word ‘explain’, so either two correct distinct points, or one point with an appropriate expansion statement would be expected. The absence of any key dependencies was a common correct answer, some candidates found it more challenging to give a second reason. Simply stating that it is in 1NF and 2NF is not enough at this level. An explanation of what it means if the database is in 1NF or 2NF would be expected.

(c) (i) There were some completely correct SQL scripts. The DELETE FROM statement was more likely to be incorrect than the WHERE clause.

(ii) Fewer completely correct SQL scripts were seen. Many candidates included a second unnecessary table in the FROM clause, which made the condition statements even more complicated. A common error was the omission of any condition about the placements being complete. In questions like this, candidates should be aware of the need to read the question carefully in order to ensure that they cover all the requirements.

(d) This is an example of a question where the use of the correct terminology in candidate answers is very important. The terms, ‘database’, ‘table’ and ‘attribute’ were frequently used incorrectly. Many candidates tried to explain what was meant by referential integrity but found describing the relationships between the primary and foreign keys challenging. Frequently the dependence was described the wrong way round and statements such as, ‘every primary key must have a corresponding foreign key’ were not uncommon. The question also asked how referential integrity applied to the database described in the question. There was very little mention of this. Bland statements just identifying the primary and foreign keys in the tables are not enough.

Question 3

(a) This question was answered very well. Some candidates need to ensure that they use the correct symbol for an XOR gate and that the circles on the NOT and NAND gates are clearly visible.

(b) This question was not answered well. Some candidates need to improve their understanding of how to interpret truth tables.

Question 4

(a) There were a number of good correct answers to this question. Candidates need to read the question carefully. A common incorrect answer used an XOR instruction when the question clearly asked for a binary shift.

(b) In this question too, candidates need to read the instructions carefully. While there were some good, correct answers, a common incorrect answer was 1110 1111, the result of XOR #12 (B00001100) with the value in the accumulator instead of XOR &12 which was what was required.

(c) This question was answered very well. Almost all candidates were able to correctly complete the AND operation.

(d) This question was answered very well. Many candidates were able to correctly complete the OR operation. A small number of candidates performed the operation OR #100 instead of OR 100, using the denary value instead of the value in the address.

Question 5

(a) (i) This question and the next part asked about functions performed by specific Operating System management tasks. Some candidates found this very challenging and it was not unusual to see a list of the other OS management tasks rather than anything specific concerning the two given in the question. There was sometimes a vague mention of the use of device drivers in relation to hardware management, but this is an area where many candidates could improve their understanding.

(ii) In this part question there was a lot of confusion between the role of the operating system and the role of software such as anti-virus and a firewall. As in the previous part question there was sometimes a vague mention of the use of authentication techniques to restrict unauthorised access, but this is an area where many candidates could improve their understanding.

(b) (i) This question was answered well. Many candidates understood that using library routines would save both programming and testing time.

(ii) Describing a drawback of using library routines was more challenging. There were some good basic statements about compatibility issues or not meeting exact needs, but almost all candidates found it challenging to expand on the drawback identified to achieve the second mark.

(c) This question was answered very well. Easily the two most popular correct answers were prettyprint and expanding/collapsing code blocks.

Question 6

(a) The terms ‘data’, ‘address’ and ‘instruction’ were frequently used incorrectly. Candidates should ensure that they do not simply repeat the name of the register in their description. For example, it is insufficient to say that the CIR holds the current instruction, the statement about decoding and execution is also required for any mark to be awarded.

(b) There were some good complete answers to this question. The most popular drawbacks given were overheating and increased latency due to the need for communication between the cores.

(c) This question was answered well. There were a good number of complete answers describing the differences between DRAM and SRAM. Some candidates need to ensure that when the question asks for the differences between two items, they include statements about both in each individual answer.

Question 7

(a) Descriptions of a static IP address were usually better than the descriptions of the public IP address. Some candidates need to understand that simply putting statements such as ‘a static IP address does not change’ or ‘a public IP address can be seen by the public’ are insufficiently detailed at this level of study. A static IP address does not change every time a device connects to a network and a public IP address allows devices to be visible on the Internet.

(b) Many candidates correctly identified the example given as an invalid IP address. However, the justifications given were often vague and lacking in substance. Answers tended to generalise without saying why it could not be IPv4 or IPv6. Candidates should understand that when asked to choose one answer from three, the justification should explicitly state why it cannot be each of the other two. Imprecise statements such as, ‘it has full stops and a value of 256’ does not say why it cannot be IPv4 or IPv6. A better answer is, ’the groups are separated by full stops so it cannot be IPv6, however, there is a value of 256 which is too big for IPv4, so the address is invalid’.

(c) Candidates found this question very challenging and it was not answered at all well. The question asked how the browser used the URL. Most of the candidate answers went into great detail about the role of the Domain Name System (DNS) but with little or no reference to the role of the web browser and so did not answer the question. Candidates who use previous mark-schemes for revision should be aware that it is unlikely that exactly the same question will be repeated and so they need to apply their knowledge to the question they are answering rather than just repeating the points from previous examinations.

(d) (i) There were many correct attempts at this calculation, but fewer correct answers. Calculators are not permitted in the examination so it is very unlikely that questions will involve complex arithmetic calculations. The numbers used will usually cancel out to leave a straightforward final calculation. Candidates who took this approach were more likely to achieve the correct answer than candidates who tried to calculate, for example, 512 * 2048, without using a calculator. Some candidates should be aware that just 8 bits are required to code 256 colours, so the multiplier for the number of bits should have been 8 and not 256.

(ii) This question was not answered well. The most popular incorrect answer for the difference in size was something to do with compression. A small number of candidates correctly realised that there would need to be a file header included.

(iii) This should have been a straightforward question, but many answers were too imprecise to achieve any marks. It is insufficient to just say pixels of the same colour. It should be made clear that the pixels are of the same colour and are consecutive or adjacent to each other. 'Similar pixels' is not enough and many answers described compressing characters or bits when the question clearly stated that it was a bitmap image file.

Question 8

(a) The command word in this question was 'explain', so more than just a basic description was required. Candidates were told in the question that viruses and pharming were types of malware, so a description of a virus which simply stated that it was malware and hence repeated the question was insufficient for any marks. There was considerable confusion between pharming and phishing, with a number of candidates describing the wrong one. This is an area of the syllabus where, at this level, candidates need to take care that they include more than just general knowledge in their answers.

(b) There were a number of excellent, thoughtful answers to this question. Some candidates appeared to mis-read the question and did not understand that the student was connecting, probably remotely, to the school network, but was using their own personal computer rather than a school computer. The stem of the question also said that one ethical consideration was the possible spreading of malware, and asked for three other considerations, so spreading malware was explicitly excluded as one of the candidate answers. Some candidates also need to take care that the three answers given are sufficiently different. For example, 'there is a risk of being hacked' and 'there is a risk of someone accessing their personal information' could be considered to be the same ethical consideration just worded differently.

(c) This question was answered very well and there were some excellent, insightful responses clearly describing both positive and negative impacts of students using Artificial Intelligence to complete their homework and of the longer-term impacts on society as a whole. A popular negative impact was the idea that by using AI students may miss out on learning opportunities that would later impact on their examination grades and future employment prospects. Conversely a popular positive impact was the ability to delve further into the subject and possibly learn more than they otherwise might.

Question 9

Almost all the candidates were able to give a general overview of the differences between a monitoring system and a control system. Few candidates included sufficient detail to achieve all three marks. Statements such as, 'a monitoring system doesn't act, a control system acts' are too vague and imprecise for credit at this level. Answers would be expected to discuss the use of feedback and actuators and to emphasise the autonomous nature of a control system.

Question 10

(a) Many candidates were able to correctly describe the model being built up one layer at a time from different types of material. Few candidates explicitly named the different types of 3D printers, but descriptions of the process were often accurate.

(b) A small number of candidates correctly described the buffer being necessary to overcome the speed mismatch between the computer and the printer, thus freeing up the computer to do other things. Many answers simply stated that the buffer was a temporary store for data. While true, this is not enough to answer the question why it is needed.

COMPUTER SCIENCE

Paper 9618/12
Theory Fundamentals

Key messages

Candidates must ensure that they read the question very carefully. Sometimes specific answers are excluded. For example, if a question says, '*Memory management is one operating system task, describe two other operating system tasks*' no marks will be awarded for an answer which describes memory management. Similarly, if the question includes an example of something such as a password, no marks will be awarded for answers that describe something with a different name that operates in the same way such as a PIN code.

In general, two marks will not be awarded for two statements which mean the same thing but which are worded differently. Candidates should be aware of the need to make each point in their answer sufficiently different.

At this level of study it is expected that candidates will respond with more than just general knowledge. Appropriate correct use of the technical terminology (jargon) is expected. Some questions require the application of knowledge and the demonstration of a deeper understanding. These questions usually ask '*how*' or '*why*' and in these situations it is not enough to simply describe a generalised situation. Answers need to be applied to the specific situation outlined in the question.

General comments

The handwriting of some candidates continues to be a cause for concern. If an examiner cannot read what is written as an answer then no marks can be awarded.

Candidates should be aware that when there are numbered spaces or boxes given on the examination paper, only the first answer in each space will be assessed. This also applies if a question asks for a certain number of examples. For example, if a question asks for two situations where something happens, only the first two candidate answers will be considered.

While planning answers is to be encouraged, especially when extended prose is required, candidates should write their plans either on the blank pages in the examination paper or on a separate sheet of paper and cross them out clearly. Writing the answer in pencil and then over-writing it in ink should be avoided. Even when a candidate thinks that it has been completely erased, the pencil image is picked up during the scanning process and the result is a double image which can not only be very difficult for examiners to read but can also lead to confusion in numeric answers.

Comments on specific questions

These comments should be read in conjunction with the published mark scheme for this paper.

Question 1

Most of the candidates were able to correctly match the visual check to data entry and the parity byte check to data transfer. Correctly matching the other two verification methods proved to be more challenging.

Question 2

(a) Many descriptions of open source and shareware licences were correct. Some candidates need to make sure that they include a full description to achieve the mark. For example, just writing that the source code is available in open source software is not enough, there also needs to be something about how the software can be used. Answers for copyright frequently described actions of plagiarism rather than the use of copyright. Describing a generic software licence was challenging for almost all the candidates. Many answers were restricted to just a commercial software licence.

(b) There were some very good answers to this question, with some thoughtful and insightful reasons why a code of conduct is desirable. A small number of candidates gave answers that were more suited to a code of practice, care should be taken to differentiate between the two.

Question 3

(a) (i) This was a question where the correct use of the terminology was important. Candidates who used the terms 'base address' and 'offset' were able to succinctly state what was meant by relative addressing. Trying to clearly describe the mode of addressing without using these terms was challenging.

(ii) This question was answered very well. Almost all candidates were able to identify two special purpose registers.

(b) The first instruction, LDM #98, was usually correctly executed, although a few candidates loaded the contents of address 98 rather than the denary value. The other two instructions were more challenging. Some candidates need to improve their understanding of indexed addressing. A common incorrect answer was 5, where the contents of the Index Register had simply been added to the contents of address 100.

(c) Many candidates found this question challenging. When explaining about clock speed vague statements such as, 'increased clock speed means more tasks are done faster' is just not detailed enough at this level of study. Answers need to include something about the number of F-E cycles and hence the number of instructions processed per unit time. Answers for cache memory frequently confused primary and secondary memory. Candidates should be aware of the need to use the correct terminology in their answers.

Question 4

(a) This is another example of a question where the use of the correct terminology in candidate answers is very important. The terms, 'database', 'table' and 'attribute' were frequently used incorrectly. Some candidates also need to understand that when describing a many-to-one (or a one-to-many) relationship it is necessary to ensure that the tables at each 'end' of the relationship are identified. Some candidates included an E-R diagram of the relationship between the tables in their answer, this was a good idea, but if they do this, candidates should ensure that they use a recognised convention for indicating the degree of the relationship on their diagram.

(b) There were some good, correct SQL scripts. Some candidates need to take care that they do not use the code from a programming language rather than SQL code when defining the data type of the additional attribute.

(c) Here too there were a number of good, completely correct SQL scripts. The most common error was the use of the SUM function instead of the COUNT function in the SELECT clause. Some candidates need to take more care when copying values given in the question. There were some interesting spellings of the ship's name.

(d) Many candidates found this question very challenging. There was a lot of confusion with Integrated Development Environments (IDEs), many answers described the purpose of an IDE in creating code rather than the purpose of the Developer Interface in a Database Management System (DBMS). Again, the incorrect use of the terminology did not help. There is a need for candidates to clearly differentiate between the tables in a database and the database itself and between a database and the DBMS.

Question 5

(a) Almost all candidates were able to give one characteristic of a Local Area Network (LAN). Expanding their answer for the second mark was more challenging. There seems to be a misconception amongst some candidates that all LANs are wired.

(b) There were some good answers to this question, most candidates connected the devices in a star topology. Some candidates misunderstood the layout of the devices and connected them in either a bus topology or a mesh topology rather than a star topology

(c) Some candidates found this question challenging and there was considerable confusion with packet switching from paper 3 where candidates described the best route being selected rather than a direct connection from sender to switch and switch to receiver. Some candidates included a correct reference to the communication being unicast rather than broadcast.

(d) Many candidates correctly stated that Carrier Sense Multiple Access with Collision Detection (CSMA/CD) would be used. Some candidates should understand that CSMA/CD is a protocol, that is a set of rules, so while the CSMA/CD states what actions should be completed, it is the devices in the network that actually carry out the relevant tasks.

(e) A threat was usually correctly identified and a suitable prevention method given although there was some confusion between phishing and pharming. The descriptions, however, were often insufficient for credit. For example, frequently when the threat was identified as malware, the description would either describe a particular form of malware, such as spyware, rather than the overarching threat, or the word 'malware' was repeated in the description. Some descriptions were too vague and imprecise for credit at this level. For example, when the threat was identified as hacking, the description stated that it allowed the computer to be hacked. In this case the description does not provide any further information than the name of the threat.

Question 6

(a) The stem of this question told candidates that the sound file had been compressed by reducing the sampling rate and asked for a justification of the type of compression. While most candidates correctly identified the compression method as lossy many of the justifications simply described lossy compression in generic terms without any reference to the sound file or the sampling rate.

(b) This question was answered very well. Some candidates need to take more care when copying from the question paper, in this type of question spelling and punctuation would be expected to be exactly as shown.

(c) (i) This question was also answered very well.

(ii) This question was not answered very well. Quite a few candidates had not attempted the question. Many candidates need to improve their understanding of parity block checks.

(d) There were many correct attempts at this calculation, but fewer correct answers. Calculators are not permitted in the examination so it is very unlikely that questions will involve complex arithmetic calculations. The numbers used will usually cancel out to leave a straightforward final calculation. Candidates who took this approach were more likely to achieve the correct answer than candidates who tried to calculate, for example, 1000 * 2000, without using a calculator. A common mistake, even when the initial calculation was correct was the omission of the division by 8 to convert bits to bytes.

Question 7

(a) This question was answered very well. There were many completely correct calculations. Some candidates, however, should understand that converting the two binary numbers to denary, adding the denary values together and then converting the answer from denary back to binary is not an acceptable method of working.

(b) This question was also answered very well.

(c) (i) There were many correct answers to this question too. A common incorrect answer was 0110 1000 where candidates had converted the given denary number to 8-bit binary instead of to Binary Coded Decimal.

(ii) Here too, here were many correct answers. A common incorrect answer was A4, where instead of treating the denary number 104 as a whole and converting it to hexadecimal either by dividing by 16 or via a conversion to binary, the first two digits,10, had been converted to hexadecimal A and the final digit, 4, converted to hexadecimal 4.

Question 8

(a) (i) There were some very good answers giving the benefits of cloud computing. Some candidates need to take more care when describing access to cloud storage. It is insufficient to say that cloud storage can be accessed from anywhere. It can only be accessed if there is an internet connection available, so the benefit should be written as, 'cloud storage can be accessed from anywhere with an internet connection'.

(ii) Answers for the drawbacks of cloud storage were also good. Easily the most popular choice was the requirement for internet connectivity, closely followed by issues of security and privacy, although some candidates seemed to think that storing any data in the cloud automatically meant that it would be hacked.

(b) (i) The question asked how the point of touch on a touchscreen is converted to a selection from the restaurant menu. Many candidates were able to explain how the point of touch would be registered on different types of touchscreens. Candidates generally then found it more challenging to explain how that point of contact was converted into a menu selection. Many of the explanations simply repeated the wording in the question.

(ii) The most popular correct choice of utility software was backup. The most popular incorrect choice was de-fragmentation software.

Question 9

(a) This question was answered very well. Some candidates need to improve their understanding of the difference between the symbols for an XOR gate and a NOR gate.

(b) This was a slightly different question about a truth table which some candidates found difficult. Row 2 was usually identified as incorrect, but identifying the other two incorrect rows proved to be quite challenging.

Question 10

(a) This was a question that needed to be read carefully. The question asked candidates to explain how a programmer can **first** use an interpreter and **then** a compiler to develop a computer program. Some candidates misread the question and gave generic descriptions of the operation of each translation program which was not what was required. Some candidates also need to understand that statements such as, 'he would use the interpreter when developing the program to…' are not acceptable. The question says that the program is being developed. Care should be taken to ensure that the meaning of each statement in an answer is explicit. Much better is, 'he would use the interpreter while coding the program to…'.

(b) There was a lot of confusion here with Freeware. Almost all candidates wrote that Free Software was free of charge, which is not necessarily true. Free software is free of restrictions, and may be free of charge, but is not always so.

(c) This question was not answered very well. Many candidates need to improve their understanding of the use of digital signatures for authentication.

Question 11

Many candidates identified the system as a control system. However, the justifications given were often vague and lacking in substance. Answers tended to generalise without including sufficient detail to achieve all three marks. Statements such as, ‘it is a control system because it is opening the doors’ or ‘it is a control system because it is not a monitoring system’ are too vague and imprecise for credit at this level. Answers would be expected to discuss the use of feedback and actuators and to emphasise the autonomous nature of a control system.

COMPUTER SCIENCE

Paper 9618/13
Theory Fundamentals

Key messages

Candidates must ensure that they read the question very carefully. Sometimes specific answers are excluded. For example, if a question says, '*Memory management is one operating system task, describe two other operating system tasks*' no marks will be awarded for an answer which describes memory management. Similarly, if the question includes an example of something such as a password, no marks will be awarded for answers that describe something with a different name that operates in the same way such as a PIN code.

In general, two marks will not be awarded for two statements which mean the same thing but which are worded differently. Candidates should be aware of the need to make each point in their answer sufficiently different.

At this level of study it is expected that candidates will respond with more than just general knowledge. Appropriate correct use of the technical terminology (jargon) is expected. Some questions require the application of knowledge and the demonstration of a deeper understanding. These questions usually ask '*how*' or '*why*' and in these situations it is not enough to simply describe a generalised situation. Answers need to be applied to the specific situation outlined in the question.

General comments

The handwriting of some candidates continues to be a cause for concern. If an examiner cannot read what is written as an answer then no marks can be awarded.

Candidates should be aware that when there are numbered spaces or boxes given on the examination paper, only the first answer in each space will be assessed. This also applies if a question asks for a certain number of examples. For example, if a question asks for two situations where something happens, only the first two candidate answers will be considered.

While planning answers is to be encouraged, especially when extended prose is required, candidates should write their plans either on the blank pages in the examination paper or on a separate sheet of paper and cross them out clearly. Writing the answer in pencil and then over-writing it in ink should be avoided. Even when a candidate thinks that it has been completely erased, the pencil image is picked up during the scanning process and the result is a double image which can not only be very difficult for examiners to read but can also lead to confusion in numeric answers.

Comments on specific questions

These comments should be read in conjunction with the published mark scheme for this paper.

Question 1

(a) This question was answered well. Many candidates were able to draw two correct lines from each change. Some candidates need to ensure that they read the question carefully, it asked for two lines from each box, some responses had only one line from some boxes, others had three.

(b) There were some good, complete explanations of how ASCII represents the text. Some candidates should understand that there is a need for precision in their responses. It is not enough to say, for example, that each character is represented by a binary or denary number. It must be made clear that this number is unique for each different character.

(c) Many candidates were able to give two differences between the ASCII and Unicode character sets. Easily the most popular correct answer was the greater range of characters in Unicode which enabled other languages to be represented.

Question 2

(a) This question was not answered very well. The first and last rows were often correct, but there were very few completely correct answers seen. Many candidates need to improve their understanding about the length of different IP addresses. Very few of the third and fifth rows were completed correctly.

(b) This was another question that needed to be read carefully. The question asked for the answer in 12-bit binary. There were many answers in only 8-bits.

(c) This question was answered well. Some candidates performed a left shift instead of a right shift.

(d) This question was answered very well. Almost all candidates correctly converted the binary integer to hexadecimal.

Question 3

(a) This question was also answered very well.

(b) This question was answered well. Some candidates need to ensure that they use the correct symbol for an NOR gate and that the circles on the NOT gates are clearly visible.

Question 4

(a) The question asked for the advantages of a wireless network compared to a wired network. Some candidates need to take care that their answers do not apply equally well to wired networks. For example, stating that an advantage of a wireless network is that multiple users can connect at the same time is not enough as this is also true of a wired network. At this level of study too, vague, general answers such as, ‘a wireless network is easier’ will not gain any credit. A reason why it is easier needs to be included. A better answer is, ‘a wireless network is easier to set up because there is no cabling required’.

(b) Similarly, here when candidates were asked for a drawback of wireless networks. Care should be taken that the drawback given does not also apply to wired networks. Just ‘wireless networks are slower’ is not enough for credit. A better answer is, ‘the transmission speed of wireless networks can be slower’.

(c) There were many completely correct answers to this question. Some candidates need to take care if expanding the acronyms that they do so correctly.

Question 5

(a) While there were some good correct responses, many of the diagrams showed the three one-to-many relationships going the wrong way. Some candidates need to ensure that they use a recognised convention for showing the relationships on an E-R diagram.

(b) There were some good, complete SQL scripts seen. Some candidates need to take more care when copying attribute names that are given in the question. A number of candidates used data types from programming languages rather than SQL data types. This should be avoided.

(c) The SELECT clause and the joining of the two tables was often completed correctly. Some candidates found the condition and the ordering more challenging.

(d) This was a question that required application of candidate knowledge and needed to be read carefully. It asked candidates how the DBMS could be used to ensure the security of the customer data, so answers needed to describe how the data was protected, not just how the chosen security method is implemented. Many candidates were able to correctly name the method, but descriptions were overall too vague and generalised for credit. For example, a popular correct method of ensuring security was the use of passwords, but a description such as, 'so only those who know the password can log in' is far too generalised. It does not describe how the use of passwords protects the data. A much better answer is, 'so that unauthorised access to customer data is prevented'.

Question 6

(a) There were many correct answers for the first two rows. Some candidates need to improve their understanding of the operation of the XOR instruction with a hexadecimal operand.

(b) The question asked for the roles of the Memory Address Register (MAR) and Memory Data Register (MDR) in the fetch-execute (F-E) cycle. Some candidates misread the question and gave a description of all the steps in the cycle not just those involving the two given registers. There is still a tendency for candidates to write about the registers performing all sorts of actions rather than holding or storing data and instructions, which indicates a lack of understanding of what a register actually is.

(c) Some candidates found completing the names of the instruction groups in this question very challenging. These candidates need to improve their understanding of the grouping of assembly language instructions.

Question 7

(a) There were some thoughtful and insightful answers to this question with many candidates describing both positive and negative impacts of the use of Artificial Intelligence in this way.

(b) There were many correct attempts at this calculation, but fewer correct answers. Calculators are not permitted in the examination so it is very unlikely that questions will involve complex arithmetic calculations. The numbers used will usually cancel out to leave a straightforward final calculation. Candidates who took this approach were more likely to achieve the correct answer than candidates who tried to calculate, for example, 4000 * 3000 * 30, without using a calculator. There was some confusion with the divisors when candidates converted the answer to gibibytes rather than gigabytes. A common mistake, even when the initial calculation was correct was the omission of the initial division by 8 to convert bits to bytes.

(c) This question asked for a method of data verification that can be used when transferring data. Many candidates read just the first part about data verification and described methods of verification used on data entry not data transfer. Some candidates who chose a parity byte check seem to think incorrectly, that the parity bit is automatically set to 1 if the number of ones in the byte is odd and zero if the number of ones is even, no matter whether odd or even parity has been agreed.

(d) There were a few good answers to this question, but many candidates found applying their knowledge to the given situation challenging. Vague answers such as, 'a hard disk stores more data', or 'a hard disk is cheaper' are insufficient for credit at this level of study. Answers need to include why the increased storage, or the cheaper cost would be considerations in this situation.

(e) This question was not answered at all well. Many candidates found it very challenging to interpret the coded algorithms and determine which data verification check was being implemented.

Question 8

(a) This was another question where candidates were being asked to apply their knowledge. Many answers described the operation of a compiler and/or an interpreter without including any reasons as to why it might be a disadvantage to use a compiler during program development. Candidates should ensure that their responses match the question on the paper and are not simply generic statements about, in this case, translators.

(b) This question was answered well. Many candidates understood that using library routines would save both programming and testing time. Some candidates found it challenging to include sufficient detail or additional points to achieve all the marks.

(c) Many candidates were able to name two software licences. An Open Source Licence was a popular correct choice. There was a lot of confusion between Free Software and Freeware. Almost all candidates wrote that Free Software was free of charge, which is not necessarily true. Free software is free of restrictions, and may be free of charge, but is not always so. Freeware on the other hand, is usually free of charge, but may not permit modification and re-distribution.

(d) This question was answered well. Many candidates were able to correctly name three other Operating System Management tasks. Some candidates should take care that if one of the management tasks is given in the question it will not also be credited as an answer.

COMPUTER SCIENCE

Paper 9618/21

Fundamental Problem-solving and Programming Skills

Key messages

This paper focuses on the practical application of computational and algorithmic thinking skills. This involves analysing and comprehending the requirements of a given scenario, as well as devising and presenting a solution. Candidates are expected to have developed these skills and demonstrate their ability to apply them effectively in the presented scenarios in order to achieve high marks.

The subject incorporates numerous technical terms and expressions with precise, defined meanings that must be used accurately. Answers are required to be specific and exact, and candidates should ensure they understand the meanings of the command words used in the paper to tailor their responses accordingly. It is essential for candidates to carefully read and comprehend each question to accurately grasp what is being asked, and to avoid assuming that similarities to previous questions indicate identical answers. Furthermore, candidates should be mindful that simply echoing phrases from the question will not result in gaining marks.

Candidates should attempt all exam questions, including those involving pseudocode, as these still offer accessible marks. Many skipped such questions entirely. When programming, consider whether to use a procedure or function, what type of loop is appropriate (count-controlled or conditional), which selection structure suits best (`IF` or `CASE`), and what variables should be used, along with what available functions, and operators may be of use before writing the code.

General comments

If any answers are crossed out, ensure that the new responses are written clearly so they can be easily read and correctly marked. Many candidates tried to fit their answers into the margins of the answer lines, which made them hard to read. Candidates should be encouraged to use a continuation booklet if needed.

Candidates need to read the question carefully in order to understand what is required before forming their answer. They should not assume that just because they think they recognise a question from a past paper that the requirement is identical. An example is **Question 1(c)**, where many candidates suggested general IDE features and not the specific ones required by the question.

Familiarity with fundamental programming concepts is vital. Lack of understanding is often illustrated by the misuse of the `LENGTH`() function in attempting to find the length of an array, it should only be used to find the length of a string. Another common mistake is the use of `OUTPUT` in place of `RETURN`. Many candidates often used a series of prompts and inputs within the body of the subroutine as well as parameters that already contained the relevant values.

The functions and operators that are available for use in pseudocode answers are described in the Insert which accompanies the paper. Candidates should be aware that the use of language-specific functions or methods that do not appear in the Insert will not gain credit.

There seems to be some confusion over the use of conditional loops. A ‘hybrid’ construct was seen quite frequently, where a count-controlled loop is placed inside a conditional loop. These will not correctly address the ‘end or exit the loop when something happens’ mark point and they may iterate indefinitely.

Candidates would usually not be awarded marks related to the loop construct for these types of answer.

Taking the MS from **Question 8(a)** as an example:

Correct:

```
Count ← 0
Index ← 1
WHILE Index <= 5000 AND Count < Max
   IF Loan(Index).CandidateID = ThisCandidateID AND__
                    Loan(Index).OnLoan = TRUE THEN
      Count ← Count + 1
   ENDIF
   Index ← Index + 1
ENDWHILE
```

Incorrect:

```
Count ← 0
WHILE Count < Max
   FOR Index ← 1 TO 5000
      IF Loan(Index).CandidateID = ThisCandidateID AND__
                      Loan(Index).OnLoan = TRUE THEN
         Count ← Count + 1
      ENDIF
   ENDFOR
ENDWHILE
```

Other common misunderstandings of pseudocode seen:

Field syntax

The dot notation syntax was often used incorrectly. A significant number of candidates attempted to convert the array index and field name into two indices of a 2D array, often replacing the field name with a numeric value.

Correct: `Loan(Index).CandidateID`

Incorrect: `Loan(Index, 1)`

Global versus local

If a scenario says that a variable is defined as global, then it should **not** be declared as a local variable within the module.

Parameter versus `INPUT`

If a scenario states that a value is passed as a parameter, then it should not be assigned a value using `INPUT` within the module.

Concatenation of numeric values

`Using & operator` to concatenate an `INTEGER`

Count-controlled loops

Omitted assignment statement, e.g.,

FOR Index 1 TO LENGTH(NumString)

instead of

FOR Index ← 1 TO LENGTH(NumString)

Comments on specific questions

Question 1

(a) Perhaps a bit more thought-provoking than the usual initial question, this gained a range of marks from zero to 4; the most commonly awarded marks being the first two rows. Row three gained moderate marks but row 4 was rarely given. A common mistake was to assume that `MyString` was numeric and so `STR_TO_NUM()` was used rather than `LENGTH()` which would be valid in all cases. Some candidates mistakenly used `NUM_TO_STRING()`.

(b) This was a straightforward question on test methods which was better answered than the previous question. The full range of marks for this question was awarded.

(c) Apart from the common but mistaken answers of 'PrettyPrint', most candidates seemed to gain 1 or usually 2 marks for reference to the names of the features. Many explanations were considered too vague to gain the third mark.

Question 2

A mixed set of responses, most gaining at least 1 mark.

Some weaker attempts were seen with incorrect usage of shapes and multiple pathways through the flowchart, often resulting in no or very few marks being awarded.

Very weak responses often resorted to prose-style e.g. 'Is it empty' with no reference to what was being checked.

Some candidates combined MP2 and MP3 into a single conditional diamond, often successfully.

For viable solutions:

MP1: An easy mark, often omitted and also sometimes seen within the loop. A small number of incorrect values seen e.g. `-1` for `Index`.

MP2: The number of iterations was commonly one out. A mistake seen on a significant number of answers: to loop if the element was empty but no iteration test (MP2 penalised, but MP3 allowed). Sometimes the YES, NO labels were reversed.

MP3: Weaker responses expressed the test in prose. Occasional mistakes included mis-naming the array or testing the index rather than the element value.

MP4: Gained in many solutions. An occasional mistake was to attempt to sum the element values rather than count the non-empty ones.

MP5: An easy mark, often given in viable solutions. A small number tried to `RETURN` a value.

Question 3

(a) (i) A very straightforward task with most candidates gaining full marks.

Common mistakes were incorrect `TYPE` statements, omission of `ENDTYPE` (or equivalent) and missing `DECLARE` for the individual fields. `VARCHAR` was seen in place of `STRING` on a few occasions.

(ii) Well answered by most candidates.

Common mistakes included specifying the array type as `STRING` and getting the indices slightly wrong.

`RentalRecords` (plural) was seen on a few occasions and was not accepted.

(b) The MS required correct and precise descriptive language (note the bolding on the MS) that many answers did not use. Many candidates could not be awarded marks for generic 'textbook' answers that gave benefits of using records **or** arrays, but **not** the combination of both, which was needed for full marks to be awarded.

As a counterbalance to this approach, the third mark was generous and was often the only one given, however many could not gain this if they did not make clear that it was the **program** that was easier to understand.

No marks were awarded for the following:

- References to saving time, reduced memory requirement or being easier to read.
- References to arrays that `RETURN` something.
- Use of easy rather than easier.

Question 4

(a) There were sufficient easy marks so that most candidates were able to gain at least 2 or 3, but the concept of a loop where something different is done on the first iteration caused confusion. Often the loop contained two input statements so would only work for the first iteration.

Most solutions attempted to compare the input with the previous array element rather than use a second variable. This tended to cause indexing problems at the start of the loop where `Index -1` was accessed.

Some solutions attempted to use `STEP 2` in their loop which was not an appropriate approach.

Several solutions initialised the input variable to a very large negative number, although no range of values was given in the scenario.

Several solutions simply assigned the input value to the next element. Others simply swapped it with the previous element if that one was not smaller.

A significant number of non-viable solutions were seen.

Mark point breakdown:

MP1: An easy mark but often not given. Very many solutions declared the global array as a local. Declaration of loop counter/index was better than in previous series.

MP2: Probably most-often given as a benefit of doubt for the first assignment within the loop, this was not often seen as a single-assignment before the loop.

MP3: Loop was present in many solutions, count-controlled being the most popular. Usually these were terminated correctly. Conditional loops generally were less successful, often for an incorrect number of iterations. 'Hybrid' loops were seen often. Often the conditional test compared adjacent elements, i.e., *after* an assignment, had been made.

MP4: Given much less frequently than MP3, with the immediate `BREAK`/`RETURN` being the most successful approach.

MP5: Many attempts were seen but fully correct solutions were very rare.

Common mistakes:

- a comparison with a previous element or assignment to the next element would have given an index out of bounds
- having two inputs in the loop
- assigning the input to an element in all cases
- attempting to swap adjacent elements if element at `n+1` was not greater than element at `n`.

MP6: Occasionally given if the syntax was correct and some attempt had been made to maintain a count. Often the message was omitted or `and` was used to concatenate an `INTEGER`.

(b) (i) 'Permanent data storage' was usually the only mark given. Several answers approached this point rather vaguely, e.g., 'text files have less chance of losing data'.

As for **Question 3(b)**, the insistence on a reasonably precise description meant MP1 was frequently missed.

Candidates are unlikely to gain marks for answers that do little more than repeat the question (even with words re-arranged).

(ii) Many candidates answered this part correctly. Some made no attempt at this question.

Question 5

All parts of **Question 5** required candidates to analyse a block of code and answer questions regarding specific sections of the code. In general candidates found these questions challenging and the answers given were too vague. Candidates need to clearly address the question being asked and use language that directly and clearly demonstrates that they understand what the section of code being examined does.

The MS required correct and precise descriptive language (note the bolding on the MS) that many answers did not use, meaning candidates were too vague to be awarded marks.

(a) (i) Correctly answered by some candidates.

The answer 'because that's the value that will be returned if the string is not found' was seen often, as was 'that is the initial value'. Both these answers were not awarded marks.

(ii) Correctly answered by some candidates, though answers were mostly too vague.

(iii) The requirement to refer to `FoundAt` being updated (or not) depending on its value was not addressed by most candidates with only a quarter gaining this mark.

(b) (i) Correctly answered by some candidates, though again, answers were mostly too vague.

(ii) Most candidates could not relate the term 'loop construct' to the definition given in the syllabus and instead offered specific loop types so could not gain the mark for 'Construct'.

The 'Justification' tended to be textbook AO1 material (i.e., 'the condition is met') and so candidates could not gain a mark as they did not specifically relate to the block of code given.

Question 6

(a) Many candidates gained some marks for this question.

A significant number of candidates did not understand that the example number had to be a 5-digit number and/or the last 3-digits had to contain the same last three digits as the number given in the question.

Where the last three digits had been preserved, often the first two digits did not sum as required.

In many cases the 'Explanation' gained the mark but many candidates who stated only that 'the checksum is the same', or that the 'the sum of **all** the digits is the same' rather than 'the sum of first four digits is the same' could not gain the explanation mark.

(b) This question turned out to be more straightforward that **Question 4** with many viable solutions being seen.

Nearly all solutions were based around a loop, although weaker solutions often took the four-digit number from the example and hard-coded a four-digit solution.

The example solution given in the MS is one possible approach to answer the question and any attempted solution was credited with the appropriate marks from the MS.

More viable solutions were seen than for **Question 4**, although there were a noticeable number of no responses.

Most candidates' solution mapped to the MS 'conversion of `Number` to a string'

Mark point breakdown:

MP1: An easy mark, given very often. Some answers comprised little more than the function heading and ending.

MP2: Given in many cases.

MP3: Although missed by some candidates who attempted to apply `LENGTH()` to the integer parameter, this was done correctly in many solutions. A mistake seen in weaker solutions was to loop for a fixed number of iterations.

MP4: Given in many cases.

MP5: Again, many solutions correctly extracted a character and converted it to a number. Although many omitted the `STR_TO_NUM()` and attempted to add a `CHAR` to an `INTEGER`.

MP6: Seen in many solutions. Occasional error included using `MOD` as a function `MOD(Sum/10)` or attempting to use `DIV`.

MP7: The 'number to string to number solution' as per the MS was the most common and was often done correctly. The alternative based on multiplying and adding was rarely seen and, in a few cases, the calculated check digit was simply added to the original parameter value.

Question 7

(a) Very well answered, with the majority gaining full marks.

A very small number of very weak answers were seen, suggesting candidates had good experience of the topic.

Incorrect labelling was seen occasionally. A swap between S2 and S5 rarely caused issues, but when S3 or S5 were involved, the pre-printed loop often led to errors and typically resulted in the MP1 not being awarded. In some cases, candidates had crossed out the loop arrows entirely, suggesting they might have benefited from spending a little more time planning their response before beginning.

On rare occasions a mark could not be awarded if the candidate duplicated a state label or added an END line.

(b) A wide range of marks was given (0 to 5), with some candidates giving no response.

MP1 was the most commonly given. A small number of candidates expanded on the module name, often adding the module type or in some cases the whole module header, which meant the mark could not be awarded.

MP3, MP4 and MP5 followed MP1 in order of popularity.

MP2 was rarely awarded.

A small number of candidates used the parameter circle incorrectly (filled/unfilled).

Question 8

(a) A wide range of responses was seen, from no response to full-mark solutions.

There were a significant number of non-viable solutions.

Many 1 mark solutions were given for little more than the module heading and ending.

The design of many solutions suggested that candidates had understood the problem but often the specifics of field notation, loop termination and threshold testing meant marks could not be awarded.

Mark point breakdown:

MP1: An easy mark that was given to most viable solutions.

MP2: Most solutions contained a loop, count-controlled being the more popular. Loop clauses tended to be terminated correctly.

MP3: Many conditional loops included the correct use of a test based on `Count`, although it was not uncommon for this to test `Count > 5`. Solutions based on count-controlled loops tended to miss this mark. The loop counter variable was frequently uninitialised or (less often) was not incremented within the loop. Often a Boolean flag would be set (for use later in MP7) but this would not affect the number of iterations.

MP4: As stated, completely correct use of dot notation was not often seen.

MP5: Even given the possible allowance for incorrect dot notation this mark was not commonly given. The test was omitted in many solutions and often the array/record reference was missed, and test was simplified to something like `IF OnLoan = TRUE`.

MP6: Often given as a follow through so became something of a 'design' mark.

MP7: When given, was often as a follow through. A common mistake was to test for: `Count > 5`

(b) Similar to **8(a)**, a wide range of responses was seen, from no response to full-mark solutions.

The design of many viable solutions suggested that candidates had understood the problem but often the specifics of field notation, loop termination and threshold testing meant marks were not awarded.

Mark point breakdown:

MP1: Probably the easiest mark given. Conditional loops were more common than count-controlled (which seemed to map to weaker solutions).

MP2: The use of an immediate `RETURN` was seen in many solutions (often preceded by a redundant statement which set a flag in order to terminate the loop).

MP3: Often the only mark out of MP2, 3 and 4. It was not uncommon to see solutions that attempted to treat the index value and field as indices of a 2D array e.g. `Loan(index, 1)`.

MP4: Attempted by many, but few marks given. Weaker solutions often used incorrect constructs such as: `IF Loan(x) = CandidateID AND BookID`.

MP5: Many took the module definition to heart and incorrectly simply wrote: `OnLoan ← TRUE`.

MP6, 7 and 8: Given the MS guidance that these could be awarded as design marks, there was often a case for giving at least one. An example of this was when a flag had been set to `FALSE` before a loop and was set to `TRUE` within the loop when the loan was returned. In these cases, often the latter case (MP6) was not enough but the final return of the flag addressed both of the other 2 marks.

(c) This part was generally poorly answered.

Some candidates wrote about changes to the library system itself rather than the program.

The very common error was to suggest saving the loan length/category but not saving the start date.

MP3 was addressed by very few and often the line between fields added to a **new** record rather than the existing record was blurred. Solutions that suggested the use of arrays (rather than records) to store the category and length were seen on a couple of occasions and were awarded this mark.

The 'program design changes' were usually very vague, often doing little more than repeating the requirements from the question. Where this mark was given, it was often for reference to 'take the due date and compare with today's date'.

COMPUTER SCIENCE

Paper 9618/22

Fundamental Problem-solving and Programming Skills

Key messages

The standard of answers seen for questions which required the writing of pseudocode were very varied, suggesting that some candidates had not adequately prepared.

Content in the syllabus requires the use of technical language which is associated with the subject. See the later comments for **Question 5**.

Candidates should:

- read the question to be clear what is expected – see the later comments for **Question 8(c)**
- pay attention to the pointers which are sometimes present on the paper and are intended to give guidance. See **Question 8(d)** which showed the guidance labels 'Data' and 'Use'.
- be aware that if there are insufficient answer lines on the paper for their pseudocode solution, then their solution is probably longer than it needs to be. Either using one of the blank pages in the booklet or asking for addition paper are acceptable. However, it this a good idea to refer to this additional work in the intended answer space.

General comments

Over half of the available marks on this paper required a detailed knowledge of the pseudocode language. Some questions required an understanding of some given code, other questions required writing code from scratch for the stated problem. The writing of code is a skill and candidates should practise this.

The following list below are weaknesses with the pseudocode which were not uncommon on this paper.

Global versus local

If the scenario in the question states that a variable is defined as a global, then it should not be declared as a local within the procedure/function.

Data type awareness

Candidates need to be clear about the data type for all variables/constants used and the implication of this. See the later comment for **Question 6** where comparisons were wrongly made between two strings when the intention was to compare the lengths of the strings.

Parameter vs `INPUT`

If the scenario in the question states that a value is 'passed as a parameter', then it must not be assigned in the body of the procedure/function a value using `INPUT`. If this is seen in the answer, marks may not be awarded for a correct procedure/function header.

Record data type and field syntax

It was rare to see the dot notation used correctly. A significant number of candidates attempted to convert the array index and field name into two indices of a 2D array, often replacing the field name with a numeric value. See **Questions 8(a)** and **8(b)**.

Correct: `Loan(Index).CandidateID`

Incorrect: `Loan(Index, 1) and Loan.CandidateID(Index)`

Incomplete syntax

Examples frequently seen were:

- Procedure/Function header missing the matching `ENDPROCEDURE/ENDFUCTION`
- An `IF` statement with the matching `ENDIF` omitted

Count-controlled loops

Candidates frequently omitted the assignment symbol.

A 'hybrid' construct was seen quite frequently, where a count-controlled loop is placed inside a conditional loop. These will not correctly address the 'exit loop when 'some condition is true' mark point and they may iterate indefinitely.

Consider these two solutions for **Question 8(b)**:

Correct:

```
Index ← 1
IsStored ← FALSE

WHILE Index <= 7000 AND NOT IsStored
   IF Loan(Index).CandidateID = Blank THEN
      Loan(Index).CandidateID ← ThisCandidateID
      Loan(Index).BookID ← ThisBookID
      Loan(Index).OnLoan ← TRUE
      IsStored ← TRUE
   ENDIF
   Index ← Index + 1
ENDWHILE
```

Incorrect:

```
IsStored ← FALSE

WHILE NOT IsStored
   FOR Index ← 1 TO 7000
      IF Loan(Index).CandidateID = Blank THEN
         Loan(Index).CandidateID ← ThisCandidateID
         Loan(Index).BookID ← ThisBookID
         Loan(Index).OnLoan ← TRUE
         IsStored ← TRUE
      ENDIF
   NEXT Index
ENDWHILE
```

Comments on specific questions

Question 1

(a) All candidates were able to score marks. There was no clear pattern about which aspects of the statements were unclear. The key points the candidate needed to identify to secure all available marks were:

- Row 1 – The `IF` keyword indicates 'selection' – the `FOR` and `NEXT` keywords identifies iteration and the `CALL` statement identifies a subroutine.
- Row 2 – The keyword `OTHERWISE` as part of a `CASE` statement, indicating 'selection'.
- Row 3 – The `WHILE` keyword indicates 'iteration'. `AllDone()` must be a function returning a BOOLEAN value and a function is one type of 'subroutine'.
- Row 4 – The syntax indicates one of the clauses from a `CASE` statement, so 'selection'.

The most common incomplete answer was where the candidate scored the 2 marks for rows 2 and 4, correctly understanding two of the components making up a `CASE` structure.

(b) Three marks were secured for almost all candidates.

(c) This was generally well answered. The simple omission of the quotation marks from either answer '12' for row 2 or answer 's' for row 4 meant a mark could not be awarded. However, this was rarely seen.

Question 2

The standard of answers seen here was very varied. Most solutions drew two separate decision boxes for the two terminating conditions. However, a single decision box with labelling that included the '`OR`' was equally acceptable.

Common errors were:

- using a shape other than the decision box for the condition(s) test
- a decision box which did not have clear yes/no labelling
- an input box which was not inside a loop
- no clear identifier name used for the input value
- the initialisation of a variable, but which later provided the incorrect index for the array element assignment; typically, the first index value to be used was not 1
- the misspelling of the array name
- the ordering of the boxes such that the terminator value of 99.9 was written to an array element
- the loop performed the wrong number of iterations; usually out by 1
- the inclusion of boxes showing declaration of variable(s) and/or some form of output.

Candidates need to be aware that some of the constructs they are familiar with in their use of pseudocode do not translate directly into flowchart components. It was common to see an answer which stated the `FOR` loop syntax inside one of the flowchart shapes.

Question 3

(a) A good discriminator question where the whole range of marks were seen. Descriptive statements were required, not pseudocode. However, most candidates completed Step 6 with the pseudocode syntax for the concatenation of the string. A common omission for the final Step 8 was to realise that the steps would be repeated until an 'end of file' was reached. However, if the name of the file `OldFile.txt` was omitted, this did not gain credit.

(b) Varied responses were seen to this part. Answers were often vague or missed the point completely, such as, 'it is never used as an arithmetic operator'.

(c) The issue described in the rubric was appreciated only by a very small minority of candidates. The 'Example data' column in the question rubric illustrated the point that these two additional encrypted data items might well contain the '\' separator.

A common wrong answer stated that the file design changes would include changing to a different separator character.

The most able candidates did mention that the solution would somehow involve calculating the number of characters in each of the additional items but were then unable to express this concisely to a concise design change. The complete answer would have stated:

- a third '\' separator character is placed after the email address
- the length of each additional data item is calculated
- the string then added after the final separator would be:
- <two character length data 1><data 1 string>
  <two character length data 2><data 2 string>

Question 4

(a) The first question of the paper requiring some pseudocode and answers seen were varied.

For most candidates, there was a misunderstanding drawn from the rubric of the question; that the index value is only ever in the range 0 to 8. The index was representing the total score possible with five easy questions (each worth 3 points) and four hard questions (each worth 5 points). The math for this equates therefore to a total points score of 35. The candidate should therefore have deduced that the bounds of the `CheckTotal` array required were 0:35.

Most answers contained a `FOR` loop and gained credit. The other 2 marks were for the declaration of the index variable and the correct assignment of the `CheckTotal` element to `FALSE`.

(b) Another good discriminator with the full range of marks seen. Many candidates did not appreciate that the syntax of the pseudocode would dictate that the contents of the square brackets must equate to an integer. `HardQ, EasyQ` was a common incorrect answer.

(c) Very few candidates scored any marks here. The expected answer was that a distinct validation check could be carried out to verify that the input parameter was inside the possible range; then, if valid, the appropriate array element was to be used. If the candidate had incorrectly assumed a range from 0 to 8 in **part (a)**, they would not have been penalised here a second time.

The second 2 marks were awarded if the candidate stated clearly that the integer parameter would be used as the index into the `CheckTotal` array, and then returning that value at this index position.

Many incorrect answers described the need to loop through all the `CheckTotal` values and so missed the key point that the index is effectively used to directly identify the single array element.

Question 5

(a) Many answers gained the full credit. No variations of the correct descriptors gained credit. 'Programming' is considered a more all-embracing term and therefore was not a viable alternative for the final answer of 'coding'.

(b) Again, the exact technical term of 'acceptance testing' was required. It seemed to be known only to a minority of candidates.

Question 6

Most candidates were able to produce a syntactically correct function header but after that, marks proved hard to secure.

Solutions were often long and vague, with a countless number of IF statements and often incorrect syntax in pseudocode statements. This was evidenced by answers often spilling over onto additional pages or blank pages within the answer booklet.

The key points which gained credit were:

- The rubric suggested that a test would be needed to check that the length of the string `String2` was longer that the length of `String1`. A frequent error was to make a comparison of the strings, and not their lengths.
- The value for parameter `Position` then determined whether a comparison was needed at the start of `String2` or the end of `String2`.
- This check involved calculating the `String1` length of characters from `String2` ready for the comparison.
- For the `CaseMatters` parameter FALSE, then both `String1` and `String2` need to be changed to either all upper- or lower-case characters before the comparison.

A common misunderstanding of the problem was that the `CaseMatters` value was to be assigned at runtime.

Question 7

(a) Most candidates scored at least 1 of the 2 available marks for the correct term 'repetition'. The second mark required a complete statement that the module `Convert` would repeatedly call the modules `Analyse`, `Update` and `Sort` in that order. Often some part of this detail was omitted.

Incorrect statements such as 'the program is repeated called' did not gain credit.

Other incorrect answers suggested that the arrow indicated some data flow between the modules.

(b) A generally well answered part, with most candidates able to score 1 or more marks.

Frequent errors included:

- the inclusion of the variable `P1` in the definition of `Update`
- `Sort` incorrectly stated as function
- the inclusion of `ByRef` omitted from the parameter in `Sort` and/or omitting the data type.

Question 8

(a) A wide range of answers seen ranging from no attempt to the full 7 marks.

The major problem with many answers was that candidates did not write the correct pseudocode syntax (the dot notation) for elements in the `Loan` array.

See the earlier comments in the 'General comments' section.

Most solutions correctly contained a loop, the most appropriate for the given problem being a count-controlled loop.

Answers then attempted to compare the parameter `CandidateID` with an element of `Loan` but this mark was often not awarded due to the incorrect or absent dot notation.

However, there were marks to be secured with two count totals with the correct logic, their initialisation and their use outside the loop to produce a meaningful output. Any pseudocode syntax errors with the output statement(s) were not awarded marks.

(b) Again, a wide range of answers seen ranging from no attempt to the full 7 marks.

Most answers contained a loop and there were 2 marks available; for iterating through all 7000 loans and a condition to terminate the loop when an unused array element was found. A conditional loop requiring a compound terminating condition appears challenging for many candidates.

Often the terminating clause in a `WHILE` loop made incorrect use of `OR` rather than `AND`

For example: `WHILE Index <= 7000 OR NOT IsStored`

Sometimes the `BookID` was incorrectly included in the condition.

For example:

```
IF Loan(Index).CandidateID = Blank AND Loan(Index).BookID = ThisBookID
```

Here, the candidate has earlier defined a constant `Blank` which is considered good practice.

As for **part (a)**, candidates who could not demonstrate the use of the dot notation to describe the three array data items were limited in the marks they could achieve.

There was more than one possible variant for the final return value. The most common design had the use of an immediate `RETURN TRUE` within a count-controlled loop, followed by a `RETURN FALSE` after the loop.

A relatively common design error was to include an `ELSE` clause in the conditional test so that `FALSE` was returned as soon as the first non-empty record was found.

(c) This was a very poorly answered part, with some candidates only able to gain a single mark for referring to the use of 'a file'. Many incorrect answers stated the solution could be the use of an additional array. Even fewer candidates secured the second mark for some detail such as 'each archived loan record will be saved as a single line in the text file'.

Some candidates misunderstood the question and described the general benefits of archiving rather than explaining features of this new `Archive()` procedure.

(d) A varied level of answers seen. There were two distinct approaches which gained credit.

The first approach was stating the email address as the 'Data' which would be needed and then a simple statement for its 'Use', such as that it will be used to send the reminder email.

The second approach was to appreciate that the return date or some equivalence would be needed. Return date on its own would suffice, but an answer which stated the 'issue date' must then also mention the 'loan period'. Whichever data item(s) were described to secure the second mark required some explanation of a calculation or check taking place (involving the 3-day period).

COMPUTER SCIENCE

Paper 9618/23

Fundamental Problem-solving and Programming Skills

Key messages

This paper emphasises the practical implementation of computational and algorithmic thinking skills. Candidates are required to analyse and interpret scenario requirements, subsequently developing and presenting an appropriate solution. Demonstrating a strong command of these skills and their effective application within given scenarios is essential for achieving high marks.

The subject matter utilises numerous technical terms and expressions with strictly defined meanings, which must be employed with precision. Responses should be specific and accurate; candidates are advised to ensure they fully understand the command words presented in each question and tailor their answers accordingly. Careful reading and comprehension of each question are vital to understanding precisely what is being asked, and candidates should not assume that similarities to previous questions imply identical answers. Moreover, reproducing phrases from the question alone will not yield marks.

It is recommended that candidates attempt every exam question, including those relating to pseudocode, as these can provide accessible marks often overlooked by candidates. When programming, consideration should be given to whether to implement a procedure or function, select an appropriate loop type (count-controlled or conditional), choose a suitable selection structure (`IF` or `CASE`), plan variable usage, and identify relevant functions and operators (from the Insert) prior to writing code.

General comments

If any responses are crossed out, candidates should ensure that their revised answers are written legibly and can be accurately assessed. A significant number of candidates attempted to write within the margins of the answer lines, which compromised readability. Candidates are advised to use a continuation booklet when additional space is required.

Understanding basic programming principles is essential. For example, some candidates incorrectly use the `LENGTH()` function to determine the number of digits in an integer, when it is meant for finding the length of a string. Another frequent error is substituting `OUTPUT` for `RETURN`. Additionally, candidates often include unnecessary prompts and inputs in subroutines for parameters values that have already been passed into the subroutine.

The available functions and operators for use in pseudocode responses are detailed in the insert provided with the examination paper. Candidates should note that employing language-specific functions or methods not listed in the Insert will not be awarded marks.

Common misunderstandings of pseudocode seen

Field syntax

The dot notation syntax was often used incorrectly. A significant number of candidates attempted to convert the array index and field name into two indices of a 2D array, often replacing the field name with a numeric value.

Correct: `Loan[Index].StudentID`

Incorrect: `Loan[Index, 1]`

Global versus local

If a scenario says that a variable is defined as global then it should **not** be declared as a local variable within the module.

Parameter versus `INPUT`

If a scenario states that a value is passed as a parameter, then it should not be assigned a value using `INPUT` within the module.

Concatenation of numeric values

`Using & operator` to concatenate an `INTEGER`

Count-controlled loops

Omitted assignment statement e.g.

```
FOR Index 1 TO LENGTH(NumString)
```

instead of

```
FOR Index ← 1 TO LENGTH(NumString)
```

Comments on specific questions

Question 1

(a) This question was generally answered well with most candidates gaining 2 or more marks.

The first 2 mark points were awarded most often with the last scenario specific mark point being achieved by very few candidates.

(b) (i) Slightly more candidates gained full marks for this question compared to the previous one. A benefit of doubt for MP3 was given for 'easier to understand the code'; this was probably the most frequent mark given.

(b) (ii) Another question where most candidates gained at least one of the 2 marks available,

Some answers were too vague to be awarded marks, for example, 'parameter' by itself was not deemed mark worthy and a significant number of candidates used 'output' instead of 'return' as part of the answer for the second mark.

(c) Most candidates gained all 3 marks available for this question.

Question 2

(a) Candidates generally gained more marks for this type of question, i.e., 'a description of the steps in an algorithm' compared to previous series.

Many candidates were awarded 3 or more marks with MP1, MP4, MP5 and MP7 being the most common mark points awarded.

The second and sixth mark points were usually only seen in answers where 5 or more marks were awarded.

Some candidates just repeated phrases from the question which, by themselves, were not awarded marks.

It is worth noting that candidates rarely, if ever, used pseudocode in their answers.

(b) Many candidates did not attempt this question. Where attempted, most candidates gained the 'Construct' mark, but many did not get the 'Use' mark as they did not reference the scenario in their answer.

Most candidates correctly identified the construct 'selection'. Very few candidates identified 'sequence' which was the other possible construct that would gain a mark.

Question 3

Another well answered question with most candidates gaining MP1 and MP3.

A significant number of candidates gained all 5 marks with MP2 being awarded the least.

Question 4

The first question on the paper where candidates had to write a complete pseudocode solution.

Most candidates gained at least 2 marks with many candidates gaining 4 or more marks.

Most candidates choose to use a count-controlled solution with the array Index going from 1 to 20. Stronger responses broke out of the loop when the condition to end the loop was met.

A range of solutions was seen with candidates using different approaches to select the sequence of 3 digits from the string, where these were viable marks were gained.

Mark point breakdown for the count-controlled solution (this was the most common approach seen) :

MP1: An easy mark which was usually given. Some solutions declared the global array locally so were not awarded this mark.

MP2: Probably the mark point most often given (for using a `FOR` loop).

MP3: This 'attempt' mark was often given.

MP4: Given much less frequently than MP3.

MP5: Many attempts were seen but a common mistake was not converting to an `INTEGER`.

MP6: This was another mark point where candidates either did not attempt, or their attempt was not successful. A common mistake was incrementing by 3 instead of 4.

MP7: This was the mark point that was rarely given but as there were 7 marks available for a 6 mark question this did not prevent some candidates achieving full marks.

Question 5

(a) Many candidates found this question very challenging with most candidates achieving 1 or no marks. Very few candidates gained more than 3 marks.

The most common mark awarded was for Zone 1.

(b) (i) Most candidates gained the mark for this question.

(ii) A much more challenging question than the previous one with some candidates not attempting it. When attempted most candidates did not gain any marks.

Question 6

(a) Very few candidates gained the first mark point with many gaining the second mark. Some candidates gained a mark for giving a relevant example.

(b) This was much better answered than **part (a)** with more candidates attempting this question.

A range of different solutions were seen with the most common converting the integer passed as the parameter to a string. Some candidates used `MOD` and `DIV` on the parameter directly to test the check digit.

Mark point breakdown for solutions where the parameter was converted to a string (this was the most common approach seen) :

MP1: This was the most common mark given. For weaker attempts, this was often the only mark. This mark point was made more accessible due to the reference to a function in the question.

MP2: Many candidates correctly converted the parameter to an integer, but weaker attempts missed this step out.

MP3: As for MP2, many candidates generally attempted this step mainly using the `MID()` substring function with fewer using the `LEFT()` function.

MP4: Many candidates attempted to iterate through the whole string, following conversion from MP2 so wrongly included the last digit.

MP5: Most candidates gained this attempt mark.

MP6: Fewer gained this compared to MP5 with many forgetting to initialise the sum.

MP7: Many candidates who attempted to sum the digits in the parameter gained this mark.

MP8: Most candidates who were awarded MP7 also gained this mark.

Question 7

(a) Very well answered, with the majority gaining full marks.

The state S3 was occasionally not labelled correctly, although sometimes the loop arrow had been crossed out, suggesting that these candidates could perhaps have spent a bit more time thinking before they started answering the question.

Incorrect labelling of events was infrequent.

(b) A surprisingly wide range of marks was given (0 to 4), with many making no attempt at the question.

MP1 was the most given mark. A small number of candidates expanded on the module name, often adding the module type or in some cases the whole module header. These additions could not be awarded credit.

MP4 was the next most awarded mark, followed by MP2. MP3 was rarely seen.

Question 8

(a) A wide range of responses was seen, from no response to full-mark solutions.

There were a significant number of non-viable solutions.

Many 1 mark solutions were given for little more than the module heading and ending.

The design of many solutions suggested that candidates had understood the problem but often the specifics of field notation and loop termination meant marks were not awarded.

Mark point breakdown:

MP1: An easy mark that was given to most viable solutions.

MP2: Most solutions contained a loop, count-controlled being the more popular.

MP3: Solutions based on count-controlled loops tended to miss this mark. Some solutions set a Boolean flag (for use later in MP8) but did not terminate the loop.

MP4: As already stated, completely correct use of dot notation was not often seen.

MP5: Even given the possible allowance for incorrect dot notation as a follow through mark, this mark was not commonly given.

MP6: This test was omitted in many solutions and often the array / record reference was not written, and the test was simplified to something like `IF OnLoan = TRUE`

MP7: This mark was only given where there was an attempt at MP6 and one of MP4 or MP5 and, where this was the case, this mark point was often awarded.

MP8: Often given, in most cases following the use of a flag. In some case this was not awarded if the flag had not been initialised before the loop.

(c) (i) Some candidates gave no response. Those who tried often gained 1 of the 2 marks available.

(ii) Again, some candidates gave no response. Of those who did answer, only about half gained the mark.

Some candidates correctly stated that '`OnLoan` was redundant as all books must have been returned' so gained this mark.

(iii) This part was poorly answered. Most answers did not precisely answer the question or were too vague.

COMPUTER SCIENCE

Paper 9618/31
Advanced Theory

Key messages

Candidates are advised to study, and also to practice and use, the relevant tools and techniques necessary to be able to provide appropriate responses to the questions on this type of examination paper.

Candidates are advised to answer each question appropriately to match the command word; for example, a question beginning with 'explain' requires more detail than a question beginning with 'state'. Also, if a question asks for working to be shown, candidates must ensure that they do this in order to gain full credit.

Candidates are further advised to make use of the published pseudocode guide when preparing for this examination, for example, in the areas of user-defined data types, algorithm construction or file handling, and answer questions requiring responses to be written in pseudocode using this syntax.

General comments

Candidates are advised to read questions carefully before beginning their answer in order to understand what is being asked of them, so as to maximise their mark.

Candidates must always make sure that they answer questions in the context of any scenario described in the question, rather than in generic terms, to receive maximum credit. In some cases, marks may be awarded for how an answer applies to the given scenario. Candidates are further reminded that credit is not usually awarded where trade names are used, for example, in **Question 9(a)(i)**.

Candidates who use the correct computer science language and technical terms when answering questions are more likely to hit marking points and therefore achieve higher marks.

Comments on specific questions

Question 1

(a) (i) Most candidates were able to set up a variable for one record of the composite data type `ClubMember`.

(ii) The vast majority of candidates achieved at least one mark for assigning some specified values to the variable they set up in **1(a)(i)**. Some candidates erroneously used the data type `ClubMember` as the variable name.

(b) (i) The vast majority of candidates achieved at least one mark for writing the type declaration of `Activity` to include the list of available activities. Some common errors seen included not beginning with `TYPE`, not using an equals sign between the key words and the list of activities, not enclosing the list of activities in brackets and using quotation marks around the individual activities.

(ii) Most candidates were able to write the new declaration statement for `Choice` that would be included in the record data type `ClubMember`. A common mistake seen was `Choice` being assigned another data type such as `STRING` or `ClubMember`.

Question 2

(a) The vast majority of candidates achieved at least one mark for this question. Most of the single mark candidates correctly wrote the mantissa part of the binary-floating point number. The correct value for the exponent was the binary equivalent of negative seven. Many incorrect responses had an exponent of positive seven.

(b) Many candidates demonstrated correct methods for calculating the normalised binary floating-point representation of a given negative denary number. Candidates seemed to find this question more difficult than previous similar ones. Three marks were available for the working and one for the final correct answer. The full range of marks was seen and candidates who arrived at the correct answer, with clearly discernible working, achieved all of the marks. Candidates who did not achieve the correct answer often achieved one or more of the working marks, where it was clear that they began correctly.

Question 3

(a) The majority of candidates achieved at least one mark for explaining that 'protocols set a standard for communication', or similar. Candidates who went on to expand their answer by adding 'protocols enable compatibility between devices on different platforms', or similar, achieved both marks.

(b) The protocols IMAP and SMTP were expected as the identification parts of this question. The vast majority of candidates correctly named one or both of these protocols. However, good quality statements were required for each one to achieve the description marks. For example, an answer like 'SMTP is used to send emails' was not considered to be enough for the mark. A more comprehensive answer such as 'SMTP is a protocol used to send emails from a computer to a mail server', or similar, was required.

(c) Candidates who described two methods in which packet switching ensures a correct message is received achieved all four marks. Most candidates achieved at least one or two marks. Methods expected included some form of checking upon receipt of the packets with requests sent to re-send missing packets, packets sent through different routes so that if a route became blocked a different route could be used, or references to the hop counter expiring and if that was the case, packets being re-sent. All of these variations were seen.

Question 4

(a) Most candidates were able to identify up to two scheduling routines, such as round robin, shortest job first, first come first served or shortest remaining time.

(b) Most candidates were able to describe at least one way in which the complexities of the computer hardware are hidden from the user, therefore achieving one or two marks. Some candidates went on to achieve all four marks. The most common answers seen included use of user interfaces with appropriate expansion and use of device drivers with appropriate expansion.

Question 5

(a) Most candidates achieved at least one mark for identifying up to two items commonly found with a digital certificate, including name of the owner, validity dates or serial number. Some common omissions that were seen included public key or digital signature without saying whose public key or digital signature, or simply stating Certificate Authority without any context, such as the certificate issuer.

(b) Candidates were required to explain why a digital certificate is required to validate a digital signature. Candidates did seem to find this question difficult, but many candidates noted that the digital certificate provides a public key. Some candidates went on to expand this point by adding 'this validates the private key used to create the digital signature'. A further expansion such as 'this makes the digital signature virtually impossible to spoof' would have achieved a third mark, but this was rarely seen.

Question 6

(a) This question was generally well answered with many full mark responses. However, some errors in some of the columns were seen, but most candidates achieved at least one mark.

(b) (i) Candidates who correctly completed the diagram with ones and zeros in all the correct places achieved the marks. Some errors that were seen included candidates who only added ones to the diagram and left the rest blank, or candidates who put ones in the wrong squares. A simple check that candidates could do is to make sure that there are the same number of ones in their Karnaugh map as there are terms in the original Boolean expression.

(ii) This question was generally well answered with most candidates achieving at least one mark. Candidates should be reminded, however, that loops must include only ones and the number of ones in the loop is expected to be a power of two, for example, two, four, eight.

(iii) Candidates mostly achieved one mark for this question for a partially correct simplified sum-of-products. The single one from the Karnaugh map, although not part of a group and therefore not requiring a loop, is still required to be part of the final Boolean expression.

Question 7

(a) The vast majority of candidates correctly identified either the A* algorithm or Dijkstra's algorithm as an Artificial Intelligence (AI) algorithm used to find the shortest distance between two points on a graph.

(b) Many good answers were seen providing descriptions of Deep Learning. Many candidates wrote that Deep Learning is a branch of Machine Learning making use of Artificial Neural Networks (ANN), which work in a similar manner to a biological brain, or similar, which gave them a good start in this question. Further marks were then available for describing the structure of the ANN and how it works to find hidden patterns in the data, that Deep Learning can involve different types of supervision and that it works well with large data sets. The full range of marks was seen for this question.

Question 8

(a) The majority of candidates achieved at least one mark for outlining the purpose of lexical analysis in program compilation. The most common answers seen included the removal of unnecessary white space and comments from the code or the conversion of the source code to a sequence of tokens.

(b) Many candidates achieved one mark for a partially correct conversion from infix to Reverse Polish Notation (RPN). Some candidates did get both marks for a completely correct RPN expression.

(c) This question was well answered with many high scoring responses seen. However, some candidates erroneously wrote mathematical symbols within the columns representing the stack. As these do not get pushed onto the stack, they are not expected to be included in the diagram.

Question 9

(a) (i) The vast majority of candidates achieved at least one mark for this question and the full range of marks was seen. Some common errors noted in responses were the use of the key word `RETURN` when it should be `RETURNS`, and vice versa, the use of `RETURNS` when it should be `RETURN`, or the incrementation of `TOP` by `+1` instead of `-1`.

(ii) Comparatively few candidates were able to construct a statement with a message to output the data item removed from the stack, such as:

```
OUTPUT 'The data removed from the stack is', Pop()
```

(b) The vast majority of candidates knew at least two of the three essential features of recursion, namely, that it should have a general case and a base case. Only the highest scoring candidates

also included the third feature, that it must change its state and move towards the base case with each recursive call.

Question 10

Candidates who were aware that exception handling is a process to respond to unwanted events when a program runs, to prevent it from stopping unexpectedly, and who then went on to name a possible cause of an exception, for example, division by zero, achieved the marks. A mixed set of marks was seen for this question that included the full range of possible marks. Candidates were mostly able to name an example of an exception.

Question 11

(a) A mixed set of marks was seen for this question where candidates had to write a brief sequence using assembly language and then show the initialisation of the constant and variable `Answer` that would be achieved when the code ran. The full range of marks was seen with many high mark answers. However, candidates were expected to write their code in the lined answer space and the constant and variable labels and values in the table provided. A relatively common error seen was candidates attempting to fit their code into the table. Some candidates also enclosed the operand part of their assembly language instructions within angle brackets, which is not correct.

(b) A high proportion of candidates incorrectly identified the final value of Answer that would be achieved after running the code as `55`, when the correct answer is `-55`.

COMPUTER SCIENCE

Paper 9618/32
Advanced Theory

Key messages

Candidates are advised to study, and also to practice and use, the relevant tools and techniques necessary to be able to provide appropriate responses to the questions on this type of examination paper.

Candidates are advised to answer each question appropriately to match the command word; for example, a question beginning with 'explain' requires more detail than a question beginning with 'state'. Also, if a question asks for working to be shown, candidates must ensure that they do this in order to gain full credit.

Candidates are further advised to make use of the published pseudocode guide when preparing for this examination, for example, in the areas of user-defined data types, algorithm construction or file handling, and answer questions requiring responses to be written in pseudocode using this syntax.

General comments

Candidates are advised to read questions carefully before beginning their answer in order to understand what is being asked of them, so as to maximise their mark.

Candidates must always make sure that they answer questions in the context of any scenario described in the question, rather than in generic terms, to receive maximum credit. In some cases, marks may be awarded for how an answer applies to the given scenario. Candidates are further reminded that credit is not usually awarded where trade names are used, for example, in **Question 9(a)(i)**.

Candidates who use the correct computer science language and technical terms when answering questions are more likely to hit marking points, and therefore achieve higher marks.

Comments on specific questions

Question 1

(a) (i) Most candidates were able to set up a variable for one record of the composite data type `Car`.

(ii) The vast majority of candidates achieved at least one mark for assigning some specified values to the variable they set up in **1(a)(i)**. Some candidates erroneously used the data type `Car` as the variable name.

(b) (i) The vast majority of candidates achieved at least one mark for writing the type declaration of `Body` to include the list of specified car body types. Some common errors seen included not beginning with `TYPE`, not using an equals sign between the command words and the list of body types, not enclosing the list of body types in brackets and using quotation marks around the individual body types.

(ii) Most candidates were able to write the new declaration statement for `BodyStyle` that would be included in the record data type `Car`. A common mistake seen was `BodyStyle` being assigned another data type such as `STRING`.

Question 2

(a) Many candidates achieved full marks for calculating the denary value of a given positive normalised binary floating-point number and showing appropriate working. Some common errors seen included using the exponent as a fractional addition to the number given in the mantissa or incorrect values derived from the mantissa after the application of the exponent. The full range if marks was seen for the question.

(b) Many candidates demonstrated correct methods for calculating the normalised binary floating-point representation of a given denary number. Candidates seemed to find this question less difficult than previous similar ones, therefore a large proportion achieved more than one mark. A common omission seen was insufficient evidence to award all the working marks; in particular, not showing how the exponent was used to achieve the final answer.

Question 3

(a) The majority of candidates gave a correct statement of the purpose of the HTTP protocol. However, many candidates gave a more generic purpose for the IMAP protocol meaning it could also have been applied to the POP3 protocol. Candidates who gave a more precise purpose for IMAP, such as 'it synchronises emails on any device', achieved the mark.

(b) Most candidates achieved at least one mark for this question, usually for identifying BitTorrent as a peer-to-peer networking system. Candidates who achieved the highest marks systematically described the steps involved with making a file available to share and then how the sharing is done using the BitTorrent protocol. A common error seen was candidates stating that a seed was a peer who has downloaded part of the file and is sharing it, whereas a seed is actually a peer who has downloaded the complete file and is sharing it. Many candidates were also very keen to discuss leeches. Although they have a part in BitTorrent, leeches are not relevant to how the files are shared.

Question 4

(a) Most candidates achieved at least one mark for this question. Marks were awarded for benefits of either circuit switching or packet switching but not given for simply stating features of each. For example, 'in circuit switching data is sent as a continuous stream' is a fact and would not achieve the mark. Candidates who also included 'so no time is wasted in reordering the data upon arrival', or similar, have correctly stated a benefit of circuit switching.

(b) Candidates who gave two differences between circuit switching and packet switching by stating how each dealt with the feature under discussion achieved the marks, with most candidates gaining at least one mark. Candidates who wrote 'circuit switching uses the whole bandwidth and packet switching does not' have not said enough. The alternative answer for packet switching would need to be something like 'bandwidth can be shared in packet switching'.

Question 5

(a) Candidates who answered describing how the low-level scheduler manages interrupts based on priority, with the higher priority processes being handled first achieved the marks. Some candidates were also credited for mention of an Interrupt Vector Table or Interrupt Despatch Table. Most candidates achieved at least one mark.

(b) Candidates were required to give one reason why a process may be in each of the three process states: running, ready and blocked. Candidates who noted the following: in the running state, processes were currently being executed; in the ready state, processes were able to be executed but were awaiting their turn; in the blocked state, processes were waiting for an event, such as an I/O operation, to be completed; achieved the marks. A common error seen was candidates who named the process state as the reason, such as the process is currently running, for the running state, or the process is ready to run, in the ready state. Answers such as these were not enough for the marks.

Question 6

(a) This question was generally well answered with many full mark responses. However, some errors in some of the columns were seen but most candidates achieved at least one mark.

(b) (i) Candidates who correctly completed the diagram with ones and zeros in all the correct places achieved the marks. Some errors that were seen included candidates who only added ones to the diagram and left the rest blank, or candidates who put ones in the wrong squares. A simple check that candidates could do is to make sure that there are the same number of ones in their Karnaugh map as there are terms in the Boolean expression.

(ii) This question was generally well answered with most candidates achieving at least one mark. Candidates should be reminded, however, that only ones should be included in the loops and the number of ones in the loop should be a power of two, for example, two, four, eight.

(iii) Candidates mostly achieved one mark for this question for a partially correct simplified sum-of-products. The single one from the Karnaugh map, although not part of a group and therefore not requiring a loop, is still required to be part of the final Boolean expression.

Question 7

(a) The vast majority of candidates correctly stated either symmetric cryptography or quantum cryptography. An occasional incorrect answer seen was asymmetric cryptography, which could not be awarded as it was stated in the question.

(b) Many full or near full mark answers were seen, indicating candidates were able to explain the use of asymmetric encryption for an organisation to receive a secure transmission. However, common errors seen included candidates not being clear that the public and private keys used were linked and belonged to the organisation, candidates who had the keys the wrong way round, so the private key was shared and candidates who mixed this question up with a question involving message digests and digital signatures.

Question 8

(a) Most candidates were aware that the purpose of optimisation during compilation was to improve the code by making it use fewer resources, or similar.

(b) Many two- or three-mark answers were seen for this question. A common error that was seen was not including brackets around the whole of the term that included the first `a`, `b` and `c`. A similar, but less common error was candidates missing out the brackets around the term involving the second `c` and `a`.

(c) This question, in general, was very well answered with many high scoring responses seen. However, a common error that should be avoided is the use of mathematical symbols within the columns. As these do not get pushed onto the stack, they should not be included in the diagram.

Question 9

(a) (i) Many good answers were seen where candidates stated uses of Deep Learning. Generic answers such as Artificial Neural Networks (ANN) or Artificial Intelligence, or trade names such as ChatGPT were not allowed. However, the majority of candidates achieved this mark.

(a) (ii) Most candidates were aware that Deep Learning could be made more effective by applying more hidden layers in its ANN.

(b) Many candidates gave good descriptions of the back propagation of errors method in Machine Learning, so many high mark responses were seen. The best answers gave a logical flow to the method, such as outputs are compared to expected outputs with weightings adjusted to minimise the difference between these. Calculus is used to determine the error gradient and the results are fed back into the neural network. Weightings of the nodes are re-adjusted as a result of the feedback, with the process repeating until results are more accurate.

Question 10

(a) Candidates who stated that exception handling is used to respond to unexpected events when a program is running and then went on to expand this with an example of how, achieved the marks. Most candidates acquired at least one mark. Some candidates, however, erroneously confused exceptions with interrupts.

(b) Candidates were mostly able to identify causes of exceptions, usually types of runtime error, such as division by zero or I/O error.

Question 11

A mixed set of marks was seen for this question where candidates had to write a brief sequence using assembly language and then show the initialisation and values of the given variables that would be achieved when the code ran. The full range of marks was seen with many high mark answers. Candidates were expected to write their code in the lined answer space and the variable labels and values in the table provided. A relatively common error seen was candidates attempting to fit their code into the table. Also, candidates sometimes enclosed the operand part of their assembly language with angle brackets, which is not correct.

Question 12

(a)(i) This question was mostly well answered with the full range of marks seen. A common error or omission was a missing or incorrect data type with the parameter name in the procedure header statement.

(ii) Candidates mostly gave partially correct answers for this question. A common error was candidates missing the `CALL` command from in front of the procedure name.

(b) Candidates were generally able to state that a stack operated as Last In First Out or First In Last Out and achieved that mark. Some candidates expanded this further and related this operation to data being pushed onto the stack and then popped from the stack in reverse order, during the operation of a recursive algorithm. The full range of marks was seen.

COMPUTER SCIENCE

Paper 9618/33
Advanced Theory

Key messages

Candidates are advised to study, and also to practice and use, the relevant tools and techniques necessary to be able to provide appropriate responses to the questions on this type of examination paper.

Candidates are advised to answer each question appropriately to match the command word; for example, a question beginning with 'explain' requires more detail than a question beginning with 'state'. Also, if a question asks for working to be shown, candidates must ensure that they do this in order to gain full credit.

Candidates are further advised to make use of the published pseudocode guide when preparing for this examination, for example, in the areas of user-defined data types, algorithm construction or file handling, and answer questions requiring responses to be written in pseudocode using this syntax.

General comments

Candidates are advised to read questions carefully before beginning their answer in order to understand what is being asked of them, so as to maximise their mark.

Candidates must always make sure that they answer questions in the context of any scenario described in the question, rather than in generic terms, to receive maximum credit. In some cases, marks may be awarded for how an answer applies to the given scenario. Candidates are further reminded that credit is not usually awarded where trade names are used, for example, in **Question 9(a)(i)**.

Candidates who use the correct computer science language and technical terms when answering questions are more likely to hit marking points and therefore achieve higher marks.

Comments on specific questions

Question 1

(a) (i) The vast majority of candidates achieved at least one mark for writing the type declaration of `Spectrum` to include the list of spectrum colours. Some common errors seen included not beginning with `TYPE`, not using an equals sign between the command words and the list of colours, not enclosing the list of colours in brackets and using quotation marks around the individual colour names.

(ii) The vast majority of candidates achieved at least one mark for identifying at least one characteristic of the list of values in an enumerated data type. Many candidates acquired both marks.

(b) This question was generally well answered with many three or four-mark responses seen. Some common errors included not using `DECLARE` for each field in the definition or declaring Colour as `STRING` instead of `Spectrum`.

Question 2

(a) Candidates generally understood that reducing the bits in the mantissa from 12 to 10 would mean the exponent would increase from 4 to 6 bits and it would reduce the precision of the number stored in the mantissa. Candidates who also went on to say that the possible range of numbers that could be stored would increase, achieved all three marks.

(b) Many candidates were generally able to describe a situation that could cause an overflow to occur, usually some form of mathematical operation. Candidates who went on to describe the consequences of this as giving a result that may be outside the range of possible numbers that could be stored, leading to the most significant bits of the number in the exponent being lost, achieved all or most of the marks.

Question 3

This question was well answered with the vast majority of candidates achieving some marks for describing how packet switching is used to pass messages across a network. Many candidates achieved high marks.

Question 4

A mixed set of marks was seen for this question where candidates had to identify and describe three protocols used in the Application Layer of the TCP/IP protocol suite. Most candidates were able to name their protocols, but the descriptions were often not specific enough to achieve the mark. For example, for FTP a candidate may write a protocol used to transfer files, which is little more than saying the meaning of the letters F T P. To achieve the description mark in this case, the words '… from one location to another on a network', or similar, would need to be added.

Question 5

(a) The vast majority of candidates knew that the purpose of multi-tasking is to allow many processes to be executed at the same time.

(b) Candidates seemed to find this question difficult. Those who achieved a mark usually did so for writing about the operating system's use of scheduling or scheduling routines. Candidates achieving more than one mark were rare, however, marks were available for expanding the point about scheduling to include a statement about the operating system making sure processes don't clash, or similar. A mark was also available for stating that the operating system monitors the state of each process.

Question 6

(a) (i) This question was generally well answered with the vast majority of candidates achieving two or three marks.

(ii) The vast majority of candidates achieved at least one mark for writing a simplified sum-of-products to represent their answer to **Question 6(a)(i)**, with many achieving both marks.

(b) Many candidates were able to simplify the given Boolean algebra expression using a range of laws including De Morgan's. A high proportion of these candidates achieved two or three marks.

Question 7

(a) Most candidates were aware that the process of optimisation during compilation included the identification and removal of redundant code so that fewer resources, such as memory use or CPU time were used.

(b) This question, requiring an infix expression to be converted to Reverse Polish Notation (RPN), was well answered, with many three-mark responses seen.

(c) Candidates were required to show the changing contents of the stack as a given RPN expression is evaluated. This question was very well answered with many full mark responses seen.

Question 8

The vast majority of candidates achieved high marks, many of whom scored full marks, for finding the shortest path between a start location and a number of other nodes, and also for showing their working. Very few errors were seen in the actual shortest distances. However, the working was not always clear enough to map to mark points. For example, working marks were awarded for initialisation, which means setting the start location to 0 and all the rest of the nodes to ∞. Other working marks were awarded for showing visited

nodes, a clear calculation of at least one of the routes taken, and evidence to show calculations of different routes to a single node, in order to find the shortest.

Question 9

Candidates were asked to outline the process followed to acquire a digital certificate. Most candidates achieved at least one mark, but full mark answers were rare. Candidates were mostly aware, however, that requests for digital certificates went to a Certificate Authority (CA), or that the company would send authentication information to the CA for them to verify, or that the CA will authorise the digital certificate once they have verified this information. Candidates who wrote all of these things in the correct order and with appropriate expansions achieved all the marks.

Question 10

(a) Candidates were required to write declaration statements for an array, two variables and a constant, all related to a stack that was created using an array. The vast majority of candidates achieved at least one mark. A common error seen was the constant being declared as another variable, rather than as a constant.

(b) Candidates were asked to write a procedure to initialise the top and bottom pointers of a stack. A mixed set of results was seen. Although some candidates did achieve full marks, many candidates made errors in defining the procedure, often including incorrect parameters, as these were not required in this case. Most of the marks awarded were for appropriate assignment statements to initialise the pointer variables.

Question 11

Candidates were rarely able to describe when the use of recursion would be beneficial. Answers expected included ‘recursion is beneficial for problems that can be broken down into smaller repetitive problems’ which can be expanded with a second point such as ‘especially for problems that are too complex for an iterative approach’. Most candidates were, however, able to give an example of where recursion could be used.

Question 12

(a) Candidates who stated that an exception is an unexplained event that happens when a program is running, due to an error in programming that was not detected during program construction, resulting in a program halting unexpectedly, achieved the marks. Most candidates achieved at least one mark, but full marks were rare. Common errors seen were candidates describing exception handling, rather than the exception itself, or confusing exceptions with interrupts.

(b) The vast majority of candidates were able to identify an example of an exception with a high proportion of these candidates also able to expand their answer to give a reason why their chosen exception may cause a problem.

Question 13

A generally well answered question where candidates had to write a brief sequence using assembly language and then show the initialisation and values of the given variables that would be achieved when the code ran. Many high scoring responses were seen. One common error that was noted involved some candidates enclosing the operand part of their assembly language statement with angle brackets, which is not correct.

COMPUTER SCIENCE

Paper 9618/41
Practical

Key messages

Candidates were often able to define classes and constructors, but fewer candidates were able to create instances of these objects, store the objects and then manipulate them using the methods.

Candidates need to consider the purpose of the points in data structures as given in the question. These can vary, for example a top of stack pointer can point to the last element in a stack of the next free space. Candidates need to analyse the question and its requirements before starting to write the solution to determine how this affects their algorithms, for example how to identify when a stack is full or empty.

Some questions require candidates to write a procedure that outputs data and some require a function that returns data. Candidates need to identify the result of the functions and make sure they are meeting these criteria, returning values when required and outputting when required. This can impact future question parts where a return value is expected and needs to be then dealt with by the algorithm that called it.

Candidates often found the creation and manipulation of 2D arrays challenging, these were often defined inaccurately and then used as a 1D array instead.

General comments

Candidates often attempted more questions, understanding that questions are independent and they do not need to have correctly answered each part to gain marks later on in the tasks.

Candidates need to make sure that their solution is completely visible in the answer space and that the correct code is in each answer space, for example the code for **Question 1a** is stored in the evidence document in the space for 1a.

Candidates should copy the code into the answer space instead of taking a screenshot, this is because the screenshot can often include colours that are not always clearly visible, the resolution on some screenshots can make the text illegible. When screenshotting the outputs candidates need to make sure the entire output is visible and that the results is legible on screen. Some screenshots were cut-off meaning that not all of the output could be seen.

Comments on specific questions

Question 1

(a) Candidates were able to assign the correct value to the pointer. Some candidates created a 1D array but did not initialise the array with 20 null values. Candidates need to consider the scenario and select a null value that is appropriate, or make use of an inbuilt null or none value depending on the programming language being used.

(b) Candidates were often able to define the function, but some responses did not return the appropriate value for example returning a string instead of a Boolean. The stronger candidates identified that the stack is full when the top of stack was equal to 29, a common error was checking if the top of stack was greater than 29. Candidates need to consider the purpose of the top of stack pointer as given in the question, in this scenario it pointed to the index of the last element in the stack, it did not point to the next empty space. Candidates were often able to increment the top of

stack appropriately. Some candidates incorrectly inserted the data in the top of stack index before it was incremented.

(c) Candidates were often able to define the function and return appropriate values. Many candidates were also able to check if the stack was empty and returned the appropriate value, some candidates inaccurately returned the string '`-999`' instead of the integer. Candidates often decrement top of stack, but some candidates incorrectly returned the element at the decremented top of stack instead of the value before it was decremented.

(d) Many candidates stored 40 values within the array. Some candidates generated a random number to store in each element although candidates often took data as input to store or hard-coded values into the array. Some candidates called `Push()` correctly with each random number and output an appropriate message. Few candidates identified the need to stop generating numbers within the loop when the stack was full, this required the return value from `Push()` to be checked and to break out of the loop at the appropriate time. Some candidates attempted to loop fewer times than required by the question to take into account the length of the array.

(e) This question required candidates to use the `Pop()` function that they had written to access each element in the stack. A common error was accessing the data in the stack's array directly by iterating through it. Some candidates attempted to use `Pop()` but called it inappropriately, often by calling it within the loop condition and then calling it a second time within the loop, this meant that the return value from the first call was not used within the loop. There were a range of methods used to find the highest and lowest values. Some candidates stored these values in variables during the loop, some candidates stored all returned values in an array and then sorted the array or used inbuilt functions to find the largest and smallest from that array. Some candidates found the highest or lowest, but through the structure of their elseif within the loop it would not find one of these if it was the first value returned by the array. Candidates need to test their programs thoroughly during creation, for example testing this algorithm with the first value returned being the largest, then the smallest, to see if the algorithm produced the correct output.

(f) (i) Many candidates were able to correctly call `FindValues()`. Candidates did not need to have attempted **part 1(e)** to answer this question.

(ii) Candidates often had an output but many did not meet all the criteria. The most common error was missing the output of '`Stack full`' from **part 1(d)**.

Question 2

(a) (i) Many candidates were able to declare the class and its constructor. Candidates often used the appropriate parameters and assigned these correctly to the attributes. Some candidates did not use their programming language constructor, for example using `create()` or a common misspelling in Python was `__int__` instead of `__init__`. Candidates need to carefully check the data types they are using or indicating match those given in the question, in this case a string and an integer. The question required private attributes, some candidates declared these correctly but some of the weaker responses created public attributes.

(ii) Many candidates were able to create a correct get method and return a value. Some candidates attempted to include a parameter in the get method and then attempted to return this value.

(b) The stronger responses were able to accurately create the four required instances of the class `Train`. Some candidates created instances but did not store them in a variable or overwrote the same variable with each new instance. Some of the weaker responses did not attempt to create an instance of the object, instead storing the data in variables or an array, or calling a function such as one of the get methods with the data. Candidates need to consider the data types of the attributes, the train ID number is a string, and the route is stored as an integer in the class.

(c) (i) Candidates were often able to define the class and its constructor. A common error was not including attribute declarations, most notably the array of type `Train` that some candidates created as a variable and not an array. Some candidates created public attributes instead of Private ones. Some candidates initialised all attributes to parameters and did not assign Trains to an empty array and `NumberTrains` to `0` as requires in the class description.

(ii) Candidates were often able to create the method header, a common error in Python responses was creating a function without the required self (or equivalent) parameter. When checking if there is space available, some candidates attempted to check `NumberPlatforms` in isolation to identify if there were any available, instead of comparing `NumberTrains` to `NumberPlatforms`. A common error was also checking if `NumberTrains` was greater than `NumberPlatforms`, without checking the condition for if they were equal, i.e., all the platforms were full. Candidates were often able to store the parameter in the appropriate position in the array and increment `NumberTrains` when this was done.

(iii) Some candidates were able to declare the method appropriately, some candidates in Python created a function instead of a method, some candidates did not always return a string in all circumstances. Candidates were often able to check if there were no trains and return the appropriate message, some candidates output instead of returning. Candidates that use Python need to make sure they are concatenating strings instead of using a comma to join separate values as tuples. Candidates often joined the values into a tuple or an array and then returned this data structure instead of returning a string. Some candidates looped through each train in the array Trains accurately and made appropriate use of the get methods to access the data. Some candidates did not concatenate each statement to the string to return, instead overwriting it each time or only returning the train data without the first statement that included the station ID.

(d) (i) Some candidates were able to create the two instances and store them appropriately. A common error was storing the number of platforms as a string instead of an integer or creating an instance and not storing it.

(ii) This question required candidates to use the methods they have created with the instances of the objects. The stronger responses methodically called `AddTrain` for each station and train in turn. Some candidates overwrote the return value each time and only checked this at the end to identify if the last attempt indicated the station was full, instead of checking each return value. The method `GetTrains` returned a string and candidates needed to call this method and output the return value, a common error was calling the method but not outputting the return value.

(iii) Some candidates were able to produce the appropriate output; a common error was missing the output of Station is full or having this output multiple times.

Question 3

(a) Candidates were often able to create the class and its constructor. Some candidates declared the class but did not use assign the parameters to the attributes.

(b) Some candidates created a 2D array and initialised each of these elements to an appropriate null record. Some candidates created a 1D array instead of a 2D array or had an incorrect number of elements. Some candidates created the array but did not create the procedure and therefore did not initialise the elements in the array.

(c) Candidates were often able to create the function and return a calculated value. Candidates often were unable to use the appropriate operator for their chosen language, most commonly those using Python, who attempted to use MOD as in the question instead of a modulus operator.

(d) Some candidates were able to call the `Hash()` function to calculate the value with the appropriate value, by accessing the key from the parameter record. A common error was sending the parameter to `Hash()` and not extracting the key, or by accessing the key incorrectly for example treating the record as an array and not an object. Some candidates were able to access the appropriate element in the 2D array, but fewer checked if there was already data in the element accurately. Candidates who did not create a 2D array often continued to access a 1D array at this stage. Few candidates were able to detect and deal with collisions by looping through the second dimension in the array until an empty space was found.

(e) Candidates were often able to open and close the file in appropriate places and read through each line in the file. Some candidates were able to split the data, most commonly using an inbuilt function. Fewer candidates then used the extracted key and data to create an instance of the class Record.

(f) Some candidates were able to use the function `Hash()` with the parameter to access the correct index in the 2D array. Candidates that were using a 1D array were unable to access the correct element to then compare to the parameter. Few candidates iterated through each element in the 2nd dimension of the array to check if the data being searched for was found in one of the possible 10 positions. Some candidates output the values Not found or the data instead of returning them from the function.

(g) (i) Candidates were often able to call `InitialiseHashTable()` and `ReadData()` even when they had not answered the earlier questions which is acceptable as each question is independent.

Some candidates did not take 5 key fields as input from the user, often when using a loop it would only take 4 values. Candidates often called `GetRecord()` but did not output the value that was returned from the function call.

(g) (ii) Some candidates were able to gain the correct output for both the key fields that were found and the one that was not found.

COMPUTER SCIENCE

Paper 9618/42
Practical

Key messages

Candidates were often able to define classes and constructors, but fewer candidates were able to create instances of these objects, store the objects and then manipulate them using the methods.

Some questions require candidates to write a procedure that outputs data and some require a function that returns data. Candidates need to identify the result of the functions and make sure they are meeting these criteria, returning values when required and outputting when required. This can impact future question parts where a return value is expected and needs to be then dealt with by the algorithm that called it.

Candidates often found the creation and manipulation of 2D arrays challenging, these were often defined inaccurately and then used as a 1D array instead.

General comments

Candidates often attempted more questions, understanding that questions are independent and they do not need to have correctly answered each part to gain marks later on in the tasks.

Candidates need to make sure that their solution is completely visible in the answer space and that the correct code is in each answer space, for example the code for **Question 1a** is stored in the evidence document in the space for 1a.

Candidates should copy the code into the answer space instead of taking a screenshot, this is because the screenshot can often include colours that are not always clearly visible, the resolution on some screenshots can make the text illegible. When screenshotting the outputs candidates need to make sure the entire output is visible and that the results is legible on screen. Some screenshots were cut-off meaning that not all of the output could be seen.

Comments on specific questions

Question 1

(a) (i) Candidates were often able to create the class and constructor. Some candidates were able to assign the parameters to the attributes, but fewer candidates also initialised the two positions to the required values with some candidates assigning another parameter instead. Some candidates declared their attributes as public instead of private or did not make this private declaration explicit when programming in Python.

(ii) Many candidates were able to create the get method accurately. Some candidates attempted to send a parameter to the function and then use this within the function.

(iii) Candidates were often able to create the method header accurately, although some candidates who wrote in Python did not create a method by including the required self (or other appropriate) parameter. Some candidates returned the attributes in two separate return statements and did not attempt to create a single string using the attributes. Candidates should understand that after a return statement executes no further code in the function will run. Some candidates who were writing in Python did not concatenate the data to return a string, instead they used a comma to separate to values creating a tuple, or created a list, then returned this data structure instead of the required string.

(iv) The stronger candidates were often able to create the method accurately, making appropriate use of the class attributes to calculate the distance and adding or subtracting the correct values from the attribute positions.

Some candidates attempted to take the values as input within the function instead of using parameters, or instead of accessing the object attributes. Some candidates added 1 to the position instead of adding the results of the distance travelled.

(b) Some candidates were able to create two instances of the objects with the required data and store these appropriately. Some candidates sent the real numbers as string data and some candidates send additional parameters that were not required. Some responses did not save the created instances in data structures or overwrote the first object with the second.

(c) (i) Many candidates were able to take the three values as input. Some candidates attempted to validate this with some accurately checking all three against all possible conditions, a common error was not full validating the time for example checking it was >= 0 but not also <= 500.

Candidates were often able to output the appropriate data for each bird, some of these candidates also made appropriate use of the get methods to access the data in each occasion. Some candidates were able to use the appropriate methods to update the chosen bird's position with the correct parameters. A common error was attempting to manually adjust the object data or not being able to manipulate the object.

(ii) This question required evidence for four tests, including the data that was input for each test and the outputs. Some candidates did not provide evidence of the inputs or cut these off the screenshots so they could not be clearly checked. Some candidates hard-coded the data to be input instead of taking them from the user.

Question 2

(a) Candidates were often able to create a 1D array. Fewer candidates were able to generate random numbers to store within the array, some candidates took these values as input or hard-coded numbers into the array. A common error was missing the requirements for each of these random numbers to be unique. Some candidates attempted to check if each number was unique, but when it was found to be a repeat, they would generate another number and store this without rechecking if the new number was unique. The stronger responses made use of inbuilt functions to check for uniqueness and/or used a conditional loop that repeatedly generated a new number until it was unique at which point a counter was incremented.

(b) Candidates were often able to produce a procedure that output the required data from the array. Some candidates did not take an array as a parameter. Some candidates output the data from the array but without the required formatting of a space between each value on a single line.

(c) Many candidates were familiar with a bubble sort and were able to produce partial solutions. Some candidates did not take the array as a parameter and return the sorted array as required by the question, candidates need to make sure they are meeting all requirements of the question. The sort needed to work on an array of any length which required candidates to calculate the length of the parameter as part of the solution, many candidates were able to identify this need and used an inbuilt function to calculate the length.

(d) (i) Candidates were often able to call the functions, but fewer used the appropriate parameters. The description of `BubbleSort()` stated that the array needed to be returned, this meant that when the function is called the return value needs to be stored and then used as appropriate, a common error was not sending the array as a parameter and not storing or using the return value. Some candidates called all three functions but did not output '`Sorted`' as required by the question.

(ii) The stronger responses produced an output that met all requirements, a common error was missing the output of `Sorted`.

(e) Some candidates were able to produce a recursive solution that sorted the array accurately. A common error was not attempting a recursive solution, instead using iteration or writing a different

searching algorithm. Some candidates produced a recursive solution but had inaccurate conditions, for example a common error was returning that the data was not found when lower = upper, which would mean that there was still one value left to check.

(f) (i) Candidates were often able to prompt for and take the value as input. The recursive call commonly had inaccurate parameters, for example sending the number of elements in the array (20) instead of the upper bound in the array (19). Few candidates checked the return value accurately and output an appropriate message, a common error was outputting the array at the position returned instead of outputting the actual index (the return value).

(ii) Some candidates produced tests for all three conditions. Some candidates were able to produce accurate screenshots showing a working solution even when their algorithms did not work, or when a different searching and sorting algorithm were used than those requested, that is acceptable.

Question 3

(a) Some candidates were able to create a 2D array. A common error was creating a 1D array or inaccurately creating a 2D array in their chosen language. Some candidates created a 2D array with an incorrect number of elements, for example 50 by 2 elements instead of 60 by 3. Some candidates initialised each element to a value other than `-1`.

(b) Candidates were often able to produce partial solutions for `AddNode()`, most commonly checking if the tree was full and outputting the appropriate message or incrementing `FreeNode`. Some candidates did not make use of the 2D array, for example creating objects for a node and then attempting to access pointer attributes instead of the 2D array structure storing the node positions. Some candidates accurately checked whether the left or right node should be accessed, then accessed this node, but repeated this directional check i.e. continued checking left each time, instead of checking if it should be right.

(c) Candidates were often able to open and close the appropriate text file; many could also access each line in the file individually. Some candidates were able to call `AddNode()` correctly with each value.

(d) Candidates were often able to define the procedure and include exception handling, some candidates used exception handling inappropriately, for example opening the text file within the try but then including file access outside of the exception entirely meaning it could potentially still crash the program. Some candidates then attempted to read data from the text file again and did not identify that this question required the writing of data to the text file. The stronger candidates were able to loop through the 2D array and join the data together into a single string to write to each line in the text file. A common error was writing all of the data from the array without creating it in the correct format.

(e) (i) Candidates were often able to correctly call `WriteAllToFile()`.

(ii) Many candidates did not produce a response to this question. Some candidates produced a screenshot that clearly showed the name of the text file and the data, some candidates had to include multiple screenshots to show all the data in the file which was acceptable provided the filename was clearly visible with some of the data on one of the screenshots. Some candidates did have the correct data stored in the text file in the correct format.

COMPUTER SCIENCE

Paper 9618/43
Practical

Key messages

Candidates need to check the data structures being created and their requirements, for example a linear queue is different in functionality than a circular queue and candidates need to consider how this affects the algorithms when producing their answers.

General comments

A small cohort took this variant of the paper, and many candidates were able to produce working solutions to the tasks.

Candidates often attempted more questions, understanding that questions are independent and they do not need to have correctly answered each part to gain marks in later tasks.

Candidates need to make sure that their solution is completely visible in the answer space and that the correct code is in each answer space, for example the code for **Question 1a** is stored in the evidence document in the space for 1a.

Candidates should copy the code into the answer space instead of taking a screenshot, this is because the screenshot can often include colours that are not always clearly visible, the resolution on some screenshots can make the text illegible. When screenshotting the outputs candidates need to make sure the entire output is visible and that the results is legible on screen. Some screenshots were cut-off meaning that not all the output could be seen.

Comments on specific questions

Question 1

(a) (i) Candidates were often able to accurately create the class and its constructor. Many candidates took the correct parameters and assigned these to the attributes within the class. Candidates often used comments appropriately to give the data types of the attributes.

(ii) Most candidates were able to create the get methods correctly and return the appropriate attributes.

(iii) Most candidates were able to create the five instances of the class and store these objects in appropriate variables.

(b) (i) Most candidates were able to create the class and its constructor. Some candidates attempted to included inheritance in the class definition which was not appropriate for this scenario.

Fewer candidates accurately created a 2D array and stored an instance of `BoardObject` in each element with the correct initial values.

(ii) Candidates were often able to create the get method and take the correct parameters, returning the appropriate value. Some candidates did not make appropriate use of the 2D array to access the data in the correct location.

(iii) Candidates were often able to create the set method appropriately and store the object parameter in the correct location. Some candidates did not use the 2D accurately.

(iv) Many candidates were able to create the method and use `GetCode()` to output each element. Fewer candidates output the data in the correct format, with a space between each element and a new line for each row in the board.

Some candidates created a procedure instead of a method, especially in Python, where self (or the appropriate parameter) was not used.

(c) (i) Candidates were often able to create a new instance of `Board()` and store the previously created objects in the appropriate positions. Candidates were also often able to call `DisplayBoard()` although some candidates called this as a procedure instead of a method.

(ii) Candidates were often able to produce the correct output.

(d) (i) Some candidates were able to take the inputs from the user; fewer did this repeatedly until they were valid.

Some candidates attempted to access the board array directly and check if there was an object as opposed to using the get method. Some candidates did not check if the return value was empty, for example did not check the Code from the return value or compare the return value with an object with appropriate values.

(ii) Candidates needed to test the program with the invalid and valid data, some candidates did not input data into the tests and hard-coded the data. Some candidates missed the invalid data entry and only used those that were valid.

Question 2

(a) Most candidates were able to initialise the pointers with the appropriate values. Candidates often created a 1D array but fewer initialised 100 elements to an empty string.

(b) Candidates often created the correct function header, but fewer accurately checked if the queue was full. Some candidates attempted to create a circular queue instead of the linear queue and only checked the number of items, which would not consider a queue that had items added and already removed.

Candidates often incremented `QueueTail` accurately, although some candidates incremented the head instead and stored the data in the tail position which would create a stack instead of a queue.

(c) Candidates were often able to create the function header and check if it was empty, most commonly by checking the number of items. Some candidates returned the data in the index of `QueueTail` instead of `QueueHead`.

(d) Candidates were often able to open and close the correct file appropriately. Many candidates read in each value from the file and called `Enqueue()` with the values. Some candidates made appropriate use of exception handling, some candidates did not include all relevant code within the try, for example opening the file within the try but then accessing or using the data outside of the exception construct.

(e) Candidates were often able to call `Dequeue()` appropriately and make use of the return value. Fewer candidates were able to loop based on the return value i.e. looping until `Dequeue()` returned no more values. Some candidates attempted to access the array directly instead of making use of `Dequeue()`.

There were a range of approaches for compressing the string, most commonly using multiple loops that kept count of each digit and then concatenated each result to a string each time.

(f) (i) Most candidates were able to call the functions accurately and output the resultant string in the global variable.

(ii) Candidates were often able to gain the correct output.

Question 3

(a) (i) Many candidates created a recursive function, this often took the appropriate parameters and checked if the array was empty. Candidates often proceeded to check the next element in the array, but fewer extracted the remaining elements and made the recursive call with this value. Some candidates called the recursive functions but called each in turn without then returning the result from these calls.

(ii) Most candidates stored the correct data in a 1D array. Many candidates made the correct call of their function with the appropriate parameters and output the return value.

(iii) Candidates often produced the correct output.

(b) (i) Most candidates were able to store the string accurately.

(ii) Candidates were often able to create the header and loop through each character. Some candidates were able to loop through each character and compare it with a semi-colon. Fewer candidates stored the elements in the array. Some candidates did not return the final array; this was either output or there was no return statement.

(iii) Most candidates called the function correctly and many of these stored the return value and output it appropriately.

(iv) Many candidates were able to produce a screenshot with the correct output.