COMPUTER SCIENCE

Paper 9618/11
Theory Fundamentals

Key messages

There is a need to use technical terminology correctly. Candidates should be aware that using a term inappropriately can completely change an answer. For example, it is not correct to say that because compression reduces the size of a file *the transmission speed* will be faster. The transmission speed depends on factors other than the size of the file. The correct response is that because compression reduces the size of a file *the time taken* to transmit the file will be shorter.

At this level of study some application of knowledge is expected, and candidates should ensure that they actually answer the question set. For example, if a question asks why lossy compression would **not** be used in a given situation, credit will not be given for just a description of lossy compression with no reference to why it is unsuitable for the scenario given in the question.

When a question requires some working, for example, number base conversions or low-level language program segments, candidates should make sure that their final answer is very clearly identified.

Candidates must ensure that they read the questions fully and think carefully about their answers before beginning to write. Sometimes examples are given in the stem of the question which are then explicitly excluded from accepted responses. For example, if a question states that a range check is an example of a validation check and asks for **two other** validation checks, no mark will be awarded if candidates give a range check as one of their answers.

General comments

Candidates should be very careful about writing their answer first in pencil and then over-writing in ink. Even after being erased the pencil markings are still picked up on the electronic scanning. This results in a double image which is very difficult to read. If an answer is illegible no marks can be awarded. Planning an answer or writing a first draft should be done either on any blank pages in the script or on an additional sheet of paper. In either case the rough copy should be clearly crossed through.

Similarly, if candidates write a response in the answer space on the question paper and later decide that the answer is incorrect and needs to be replaced, the new version should be written either on any blank pages in the script or on an additional sheet of paper, rather than trying to squash the new answer into the existing answer space. When this happens a note for the Examiner such as, ‘*Please see additional page 1*’, is very helpful. Candidates should also avoid writing in the margins of the page as these can sometimes be removed during the scanning process.

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 correctly chose the second option.

(b) This question was answered well, many candidates were able to draw a correct logic circuit for the given expression. The most common error was a misinterpretation of the position of the innermost brackets resulting in the circuit for NOT(NOT A AND NOT (B XOR C)) instead of the one given.

Question 2

(a) This was a question where candidates were expected to apply their knowledge of embedded systems to a specific example, in this case, a video doorbell. Answers tended to fall into two groups, either a repeat of the information given in the question which just described the doorbell without any reference to it being an embedded system, or a generic description of an embedded system with no reference to the doorbell. Candidates should read the question carefully and make sure that the responses match what is being asked.

(b) There were some good answers to this question, with appropriate justifications of both a monitoring and a control system. Some candidates should understand that vague answers such as, '*it is a monitoring system because it does not control anything'* are insufficient for credit at this level of study.

(c) (i) This question was answered well. The most popular correct answers were the start-up instructions and the current sensor readings.

(ii) This question was not answered at all well. Some candidates were able to identify the two types of logic gates used and the number of transistors contained in each cell. Many candidates need to improve their understanding of the principal operation of solid state memory.

(iii) This question was not answered well. Some candidates were able to state that a buffer was used as a temporary store for data. Many answers then included statements such as, '*the buffer sends the video data to the secondary storage*'. Candidates should be aware that statements like this demonstrate a lack of understanding, if a buffer is an area of storage, it cannot 'do' anything. It is a processor that ensures the data is sent from the buffer to the secondary storage. A better answer is, '*the microprocessor in the doorbell ensures that the video data is transmitted from the buffer to the secondary storage device*'.

(d) This was another question where candidates were expected to apply their knowledge, and it was not answered at all well. A few candidates realised that increasing the sampling rate would increase the size of the file and that it would therefore take longer to transmit the file to the smartphone and cause the secondary storage to fill up sooner, both of which affect the performance of the doorbell.

(e) (i) While there were a few good answers to this question, many candidates need to improve their understanding of what is meant by bit streaming.

(ii) This question was answered well. Many candidates were able to correctly give two differences. The most popular correct answers described live versus stored content and the whether the user was able to pause, rewind etc.

Question 3

(a) Many candidates were able to explain how an interpreter stops when it encounters an error, thus allowing the error to be fixed in real-time. Some otherwise correct answers were not able to be credited as candidates wrote that it was the interpreter fixing the errors rather than the developer.

(b) This question was generally answered well. Many candidates were able to explain that using an executable file meant that the users had no access to the source code and so could not edit or copy it. Further explanation was more challenging. Vague statements about .exe files executing faster without any expansion are insufficient for credit at this level.

Question 4

(a) There were a number of good, completely correct answers to this question. Candidates must ensure that their final answer is clearly indicated. Some candidates need to improve their understanding of the different operands used; specifically, when the operand is a value and when it is the contents of a memory location.

(b) This question was also answered well. Many candidates correctly applied the logic operations to the appropriate operands. The first answer was the one most likely to be incorrect where candidates had used denary 31 for the operand rather than the contents of memory location 31.

Question 5

(a) Candidates found this question challenging. Bland statements such as, '*the server is the bank server and the client is the customer'* are insufficient. Many responses described the actions of the smartphone **user**, when the question asked for the roles of the different **devices**. When describing the role of the server, for example, the minimum that would be expected is a statement about the storage of all the customer data and another statement about receiving and processing of requests from the client.

(b) There were some good, complete answers to this question. Some candidates need to improve their understanding of parity, particularly of the use of parity blocks.

(c) (i) There were some good, complete answers to this question. Some candidates should be aware that vague general answers such as just, '*a firewall blocks malicious software*' are not technical enough for credit at this level of study.

(ii) There were a few interesting and imaginative answers to this question. Many candidates seemed to be familiar with the use of facial recognition on a smartphone. Some candidates need to ensure that they are describing the use of Artificial Intelligence for facial recognition, not describing the use of facial recognition for authentication. There was considerable confusion between the two.

Question 6

(a) Almost all candidates were able to describe a table for the users including the fields username, email address and age. Some candidates overlooked the statement in the stem of the question that stated that the username was a unique attribute and incorrectly used the first-name and last-name combination as the primary key. Few candidates realised that the user rating would also be included in this table. The table describing each quiz often contained the correct attributes, with the filename a popular correct choice for the primary key. The linking table was a little more challenging, although some candidates recognised it as the breakdown of a standard many-to-many relationship and correctly implemented it as such.

(b) Descriptions of the Data dictionary were more often correct than the descriptions of the Logical schema. Some candidates need to improve their understanding of a logical schema. Many candidates were able to state that the data dictionary contained the metadata about the database and give appropriate examples of content. A common incorrect description for the logical schema was, '*a schema that stores the logic*'. Some candidates attempted to describe logical operations within a database and wrote about the evaluation of logic expressions. A correct description of a logical schema is, '*an implementation independent overview of the database, using a method such as an Entity-Relationship diagram*'.

(c) (i) There were a few correct SQL scripts. Many candidates need to improve their understanding of the syntax for the statement to add a foreign key to an existing table.

(ii) In this SQL script, the `FROM` clause was often correct. The `SELECT` clause was usually attempted, but there was confusion between the use of `SUM` and `COUNT`. Several otherwise correct `SELECT` clauses attempted to total the data in the `EventID` field instead of counting the number of events for each player. The `GROUP BY` clause was frequently missing.

Question 7

This question was answered very well. Some candidates should take care to ensure that their working is clearly visible and that any overflow is clearly indicated.

Question 8

(a) This question was also answered well. Some candidates need to improve their understanding of the difference between a star topology and a mesh topology.

(b) (i) This question too was answered well. Many candidates were able to correctly state what is meant by a public IP address.

(ii) Many candidates were able to correctly identify two differences between IPv4 and IPv6 addresses. The most popular correct answers were the different number of bits and the different number of groups of digits. Some candidates need to ensure they read the question carefully. No mark was awarded for the different separators as this was given in the question. A small but significant number of candidates incorrectly stated that IPv6 had six groups of digits rather than eight.

COMPUTER SCIENCE

Paper 9618/12
Theory Fundamentals

Key messages

There is a need to use technical terminology correctly. Candidates should be aware that using a term inappropriately can completely change an answer. For example, it is not correct to say that because compression reduces the size of a file *the transmission speed* will be faster. The transmission speed depends on factors other than the size of the file. The correct response is that because compression reduces the size of a file *the time taken* to transmit the file will be shorter.

At this level of study some application of knowledge is expected, and candidates should ensure that they actually answer the question set. For example, if a question asks why lossy compression would **not** be used in a given situation, credit will not be given for just a description of lossy compression with no reference to why it is unsuitable for the scenario given in the question.

When a question requires some working, for example, number base conversions or low-level language program segments, candidates should make sure that their final answer is very clearly identified.

Candidates must ensure that they read the questions fully and think carefully about their answers before beginning to write. Sometimes examples are given in the stem of the question which are then explicitly excluded from accepted responses. For example, if a question states that a range check is an example of a validation check and asks for **two other** validation checks, no mark will be awarded if candidates give a range check as one of their answers.

General comments

Candidates should be very careful about writing their answer first in pencil and then over-writing in ink. Even after being erased the pencil markings are still picked up on the electronic scanning. This results in a double image which is very difficult to read. If an answer is illegible no marks can be awarded. Planning an answer or writing a first draft should be done either on any blank pages in the script or on an additional sheet of paper. In either case the rough copy should be clearly crossed through.

Similarly, if candidates write a response in the answer space on the question paper and later decide that the answer is incorrect and needs to be replaced, the new version should be written either on any blank pages in the script or on an additional sheet of paper, rather than trying to squash the new answer into the existing answer space. When this happens a note for the Examiner such as, '*Please see additional page 1*', is very helpful. Candidates should also avoid writing in the margins of the page as these can sometimes be removed during the scanning process.

Comments on specific questions

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

Question 1

(a) This question asked for descriptions of four logic gates. Answers such as '*a NAND gate is the opposite of an AND gate*' or '*a NOR gate is an OR gate followed by a NOT gate*' are far too general for credit at this level of study. A complete description of a logic gate should include the output for every possible combination of inputs, so it is not enough to simply state when the output would be, for example, 1 (or high). An example of a good answer for a NOR gate is, '*the output is 1 when both inputs are 0, in all other cases the output is 0*'.

(b) This question was answered well, many candidates were able to draw a correct logic circuit for the given expression. The most common error was a misinterpretation of the position of the outermost brackets resulting in the circuit for NOT(A AND B) OR (B AND C) instead of the one given.

Question 2

(a) There were a few good answers to this question, many candidates were able to correctly complete the first statement. Some candidates need to improve their understanding of the principal operation of a VR headset.

(b) This question was not answered well. Some candidates were able to state that a buffer was used as a temporary store for data. Many answers then included statements such as, '*the buffer sends the data to the headset*'. Candidates should be aware that statements like this demonstrate a lack of understanding, if a buffer is an area of storage, it cannot 'do' anything. It is a microprocessor that arranges for the data to be sent from the buffer to the headset. A better answer is, '*the microprocessor in the headset ensures that data is retrieved from the buffer*'.

(c) The command word in this question was 'explain'. Many candidates were able to state the benefits of using EEPROM instead of other types of ROM but without further explanation or expansion. Candidates should be encouraged to look carefully at the command words in the question. A question that begins with the word 'explain' requires a different response to one that starts with the word 'state' or 'identify'.

(d) (i) Many of the responses to this question were very general and some described the encoding of characters rather than pixels. There were a number of references to each colour being given a code, but few responses made it clear that these codes were unique to each different colour or that the codes were then stored in the same order and sequence as the pixels in the image. Some candidates mis-interpreted the meaning of the word 'encoded' in the question and described different forms of compression.

(ii) There were some good, complete answers to this question. Many candidates were able to correctly describe the contents of a vector graphic drawing list. Some candidates need to take greater care with the wording of their answers. Several descriptions referenced all the shapes that can be drawn, which reads as the library of shapes rather than the subset of shapes required to make up a particular image.

(iii) This question was also answered well. Many candidates were able to give two correct reasons why the video did not need to be compressed. Easily the most popular answer was that there was so that there was no degradation in the quality of the video and hence the user experience was not adversely affected.

Question 3

(a) (i) This was a question where candidates needed to read the stem of the question carefully. Some candidates overlooked the statements in the question that explicitly ruled out any methods of authentication, and passwords or biometric scans were popular incorrect answers. The most frequent correct answer was a firewall, and many candidates who identified this measure were also able to give a complete description.

(ii) There were a number of excellent complete answers to this question. Encryption was easily the most popular correct choice. Candidates who choose to describe a particular method of encryption should be careful about stating that the encryption key is sent with the cipher text.

(b) While there was a small number of good answers, there was considerable confusion here between the characteristics of thin clients and the characteristics of thick clients. Some candidates need to improve their understanding of the differences between the two. The question asked for descriptions of the characteristics as used in the exam marking software. Frequently descriptions simply repeated the characteristic with no reference to the software. Candidates should also be aware that at this level of study marks will not be awarded twice when one characteristic is the converse of the other. For example, first answer, '*the server does most of the processing*', second answer, '*the client does very little processing*'.

(c) (i) This question was generally answered well. Some candidates need to be more careful with the use of the technical terminology. 'Package' is not acceptable instead of 'packet'.

(ii) Many candidates found this question challenging and need to improve their understanding of the role of the Public Switched Telephone Network (PSTN) in the transmission of data over the internet.

Question 4

(a) This question was answered very well. Most candidates were able to correctly identify the relationship between the given tables.

(b) There were several good, completely correct SQL scripts. When the table and attribute names are given in the question candidates should take care to copy them correctly in their answers.

(c) There were a small number of correct scripts, but many candidates found linking the foreign key much more challenging. A frequent error was the use of `UPDATE TABLE` instead of `ALTER TABLE`.

(d) Answers to this question would have benefitted from some initial planning. Very little in the way of planning was seen on any of the scripts. Many candidates realised that a table for data about the candidates would be needed, although frequently there was no formal identification of a primary key. Identification of the other tables needed proved to be much more challenging. Some candidates attempted to modify the given tables which was not what was required. Few candidates realised that in total three additional tables would be needed to avoid any many to many relationships. Some candidates did describe a second table to link the candidates to the exams they had taken and correctly included the primary keys of the `EXAM` table and the `STUDENT` table as foreign keys. More often than not this table also included the `ExamQuestionID` and the marks gained by the candidate which would have been better separated out to a third table.

Question 5

(a) There were a number of good, completely correct answers to this question. Candidates must ensure that their final answer is clearly indicated. Some candidates need to improve their understanding of the different operands used; specifically, when the operand is a value and when it is the contents of a memory location.

(b) This question was also answered well. Many candidates correctly applied the logic operations to the appropriate operands. The first answer was the one most likely to be incorrect where candidates had used denary 29 for the operand rather than the contents of memory location 29.

Question 6

Many candidates found this question challenging. There was some confusion between memory management and file management, with several responses describing utility programs such as defragmentation. Candidates should also be aware that at this level of study there is a need to demonstrate more than just general knowledge, and statements such as '*memory management manages memory*' or '*process management manages processes*' are far too vague for credit. The question asked for an explanation of **how** these two operating system management tasks supported multi-tasking, so reference to the use of memory and the allocation of other resources to several processes appearing to run simultaneously would be expected.

Question 7

(a) This question was generally answered well. Some candidates need to improve their understanding of the different units of file sizes. A popular incorrect answer was 3 mebibytes.

(b) Again, there were a good number of correct answers. A frequent incorrect answer was the calculation of (10 – 100) instead of (100 – 10). Some candidates should understand that when the question asks for a calculation to be completed using binary subtraction, no credit will be given for performing the calculation in denary and then converting the answer to binary.

(c) This question was also answered well. A small number of candidates correctly converted the Hexadecimal values to 4-bit binary but then omitted to include the zero in their final calculations.

Question 8

(a) Many candidates correctly stated that a compiler creates an executable file, but then proceeded to describe the operation of a compiler during the translation process rather than describing the benefits of using it during the testing phase of development which did not answer the question set.

(b) This question asked for features of an IDE for three different purposes during program development. It would thus be expected that the description of the feature chosen for each purpose would be described in the context of that purpose. This was usually the case with the feature chosen for debugging, but not with the other two. Frequently the descriptions of the features for coding and presentation simply expanded on the name of the feature. For example, if the feature chosen was *auto indentation*, the description given was, '*automatically indents the code as it is written*'. There was no reference to how the feature enhances presentation or helps with coding.

(c) There were a number of good, complete answers to this question. Some candidates should ensure that they read the question carefully. If the question asks for benefits to the programmer, it is not enough to just describe the features of a program library. There needs to be some reference to how the features benefit a programmer.

COMPUTER SCIENCE

Paper 9618/13
Theory Fundamentals

Key messages

There is a need to use technical terminology correctly. Candidates should be aware that using a term inappropriately can completely change an answer. For example, it is not correct to say that because compression reduces the size of a file *the transmission speed* will be faster. The transmission speed depends on factors other than the size of the file. The correct response is that because compression reduces the size of a file *the time taken* to transmit the file will be shorter.

At this level of study some application of knowledge is expected, and candidates should ensure that they actually answer the question set. For example, if a question asks why lossy compression would **not** be used in a given situation, credit will not be given for just a description of lossy compression with no reference to why it is unsuitable for the scenario given in the question.

When a question requires some working, for example, number base conversions or low-level language program segments, candidates should make sure that their final answer is very clearly identified.

Candidates must ensure that they read the questions fully and think carefully about their answers before beginning to write. Sometimes examples are given in the stem of the question which are then explicitly excluded from accepted responses. For example, if a question states that a range check is an example of a validation check and asks for **two other** validation checks, no mark will be awarded if candidates give a range check as one of their answers.

General comments

Candidates should be very careful about writing their answer first in pencil and then over-writing in ink. Even after being erased the pencil markings are still picked up on the electronic scanning. This results in a double image which is very difficult to read. If an answer is illegible no marks can be awarded. Planning an answer or writing a first draft should be done either on any blank pages in the script or on an additional sheet of paper. In either case the rough copy should be clearly crossed through.

Similarly, if candidates write a response in the answer space on the question paper and later decide that the answer is incorrect and needs to be replaced, the new version should be written either on any blank pages in the script or on an additional sheet of paper, rather than trying to squash the new answer into the existing answer space. When this happens a note for the Examiner such as, '*Please see additional page 1*', is very helpful. Candidates should also avoid writing in the margins of the page as these can sometimes be removed during the scanning process.

Comments on specific questions

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

Question 1

(a) The two calculations were usually performed correctly. Some candidates found completing the names for the two prefixes more challenging.

(b) This question was answered very well. Almost all candidates were able to correctly convert the given denary number to hexadecimal.

(c) This question was not answered well. Many candidates gave insufficient statements such as, '*the answer is greater than 255*'. This would be true if the two binary numbers being added were each 8 bits long, but there was no reference in the question to the length of the addends. The complete answer should cover all cases, no matter how long the original numbers.

(d) (i) This question was answered well. The answer most likely to be incorrect was the ASCII character set, where some candidates put 8 bits.

(ii) This question was also answered well. Some candidates need to ensure that they stress the uniqueness of the binary codes corresponding to each character, but there was a clear understanding that the binary values would be stored in the same sequence as the letters in the given word and that the uppercase and lowercase letters would have different codes.

Question 2

(a) An initial correct calculation for the number of bytes was usually seen. Some candidates should remember that if they then multiply by 8 to bring the value to bits, they also need to divide by 8 to bring it back to bytes before converting to megabytes.

(b) (i) This was a question where candidates needed to think carefully about the wording of their answers. The question asked for benefits of using lossy rather than lossless compression, so to simply state that the file size would be smaller is not enough. That is a statement of fact, not a benefit. A benefit would be that because the file size is smaller than using lossless compression the file takes up less storage space on the server, and hence more photographs can be stored. There is also a need here to differentiate between the transmission speed of uploading and downloading and the time taken to upload or download the file. The transmission speed depends on factors other than the size of the file.

(ii) There were some good, complete answers to this question. Some candidates need to ensure that they answer the question on the examination paper. Responses describing the use of run-length encoding to compress text files with sequences of repeating characters were seen regularly. When the question asks about compressing a photograph, that is, an image, no credit will be given at this level of study for answers about compressing text.

(c) This question was answered very well. Easily the most popular correct answers were bit depth and image resolution.

Question 3

(a) There were a number of good, completely correct answers to this question. Candidates must ensure that their final answer is clearly indicated. Some candidates need to improve their understanding of the different operands used; specifically, when the operand is a value and when it is the contents of a memory location.

(b) There were some good correct answers to this question. Some candidates confused their left and right and shifted the bits the wrong way. The second instruction using an XOR command was more likely to be incorrect than the AND statement.

Question 4

(a) This question was answered very well. Most candidates were able to correctly identify the relationship between the given tables.

(b) There were a small number of good, completely correct SQL scripts. The most frequent error was defining the `PerformanceID` as data type integer. Some candidates found the definition of the foreign key very challenging. When the table and attribute names are given in the question candidates should take care to copy them correctly in their answers, and care must be taken with the use of commas and brackets.

(c) Many candidates found writing this SQL script very challenging. Some candidates did not attempt to answer the question but instead tried to implement the example given on the examination paper with a series of WHERE clauses reflecting the data given in **part (b)**. The GROUP BY clause was rarely seen. The clauses most likely to be correct were the FROM clause and the WHERE or ON clause.

(d) Answers to this question would have benefitted from some initial planning. Very little in the way of planning was seen on any of the scripts. Many candidates realised that a table for data about the customers would be needed, although frequently there was no indication of a table name or formal identification of a primary key although the list of other attributes was often comprehensive. Identification of the other tables required proved to be much more challenging. Some candidates attempted to modify the given tables which was not what was required. Few candidates realised that in total three additional tables would be needed to avoid any many to many relationships. Some candidates did describe a second table to link the customers to the performances they wished to attend and correctly included the primary keys of the `PERFORMANCE` table and the `CUSTOMER` table as foreign keys. More often than not this table also included the `SeatID` which would have been better separated out to a third table linked to the booking.

Question 5

(a) (i) At this level of study, the answer to this question needed more than just general knowledge. Responses would be expected to include some more technical description of a cloud service and also what was meant by that service being private.

(ii) This question also asked for benefits to a company of using private rather than public cloud services, and so answers needed to be more than generic benefits of cloud storage or simple statements of fact. Answers such as, '*a private cloud is more secure*' are insufficient. What makes it more secure and how is the fact that it is more secure of benefit to the company?

(b) This question was generally answered well. Some candidates need to be more careful with the use of the technical terminology. 'Package' is not acceptable instead of 'packet'.

(c) (i) This question was not answered well. Many candidates were aware of the processes involved but answers included statements such as, '*the CSMA/CD sends a jamming signal*'. Candidates should be aware that statements like this demonstrate a complete lack of any real understanding. CSMA/CD is a protocol, a set of rules, and as such it cannot 'do' anything. It is the transmitting device that sends the jamming signal. In this subject, at this level of study, correct use of the technical terminology is essential for credit.

(ii) Many candidates were able to give one correct drawback of using CSMA/CD, usually connected with the time to transmit because of the need for devices to wait. Giving a second drawback proved more challenging and many responses were a repeat of the first answer using different words.

(d) As with the definition of a private cloud, responses to this question would be expected to include a description of a static IP address and also what was meant by that IP address being private. Many answers only stated what was meant by a static IP address and did not refer to the fact that it was private.

Question 6

This question was answered very well. Some candidates need to be careful if they change their minds about an answer and should make sure that the incorrect answer is clearly indicated as such.

Question 7

(a) This question was answered well. Many candidates were able to correctly identify two appropriate sensors and give suitable purposes for each.

(b) There were some excellent, imaginative answers to this question. Speech recognition and natural language recognition were often described in detail. Some candidates need to take care that they answer the question, which asked how Artificial Intelligence is used to communicate with the customer, rather than giving a detailed description of the processes after the robot has taken the order.

(c) Many candidates correctly stated that feedback ensures that the system operates within set criteria. Giving a second correct statement proved to be more challenging.

(d) There were many complete descriptions of the principal operation of a touchscreen. The most common types described were capacitive and resistive, although some others were also seen.

(e) (i) This question was answered well. Many candidates were able to correctly state what is meant by a program library.

(ii) There were some good answers to this question. Some candidates need to ensure that they are writing about the benefits of Dynamic Link Library files and not the benefits of using any generic program library. Answers such as, '*it saves the programmer time because he does not need to write as much code*' apply to the use of any library files and are not DLL specific.

(f) (i) This question needed to be read carefully. Quite a number of candidates saw the words 'data verification' in the stem of the question but then missed the words 'during data transfer' which followed. This resulted in incorrect answers such as double entry and visual checks. The most popular correct methods chosen were a checksum and parity. The descriptions of the checksum were usually correct. Care needed to be taken with the descriptions of parity. There seems to be a misconception that if odd parity is chosen a 1 is added to the byte regardless of how many other 1s there are, and similarly that if the parity is even a 0 is added. Descriptions often did not make it clear that each byte was checked for parity and there was little mention of horizontal and vertical parity checks on a block of data. Answers frequently implied that a single 0 or 1 would be added to the complete data to make it odd or even.

(ii) This question was answered very well. Almost all candidates were able to explain how encryption protects the security of data during transmission. Some candidates need to be careful about using the word 'unreadable' rather than 'not understood'. Encryption does not prevent a hacker accessing or reading the data, it prevents the hacker understanding what has been accessed.

COMPUTER SCIENCE

Paper 9618/21

Fundamental Problem-solving and Programming Skills

Key messages

This paper addresses the application of practical skills or 'Computational Thinking'. These skills involve analysing and understanding the requirement as well as designing and presenting a solution. Requirements are often presented through the use of a scenario and candidates need to understand this before formulating their answer.

Answers should be as precise and specific as possible. Candidates should familiarise themselves with the meanings of the command words used in this paper and form their answers accordingly.

Candidates should be encouraged to attempt all questions. Even the more advanced questions that ask for an algorithm to be written in pseudocode contain accessible marks.

General comments

This paper involves the application of practical skills. These skills involve analysing and understanding the requirement as well as designing and presenting a solution. Requirements are often presented using a scenario description. Candidates need to be able to identify the key elements of each requirement (for example, the need for an iterative structure) when designing their solution. The development of these skills requires practice.

This subject makes use of many technical words and phrases. These have specific, defined meanings and they need to be used correctly.

The functions and operators that are available for use in pseudocode answers are described in the Insert which accompanies the paper. Candidates should be advised that they should not use language-specific functions or methods that do not appear in the insert.

Candidates need to read each question carefully before attempting to answer it. Questions may address topics in many ways, and it is often necessary to apply knowledge in a specific way if marks are to be gained.

If answers are crossed out, the new answers must be written clearly so that the text may be read.

Many candidates make use of blank pages for rough work when preparing their final answer. In these cases, it is extremely important that this text is crossed out.

Comments on specific questions

Question 1

(a) This question part was well answered by most candidates.

Some candidates did not spot the quotation marks for variable `D` answering '`Boolean`' incorrectly.

(b) This question part was well answered by most candidates.

Some candidates did not use quotations for their string response on the third line.

(c) (i) Generally, this question part was well answered.

Some candidates did not recognise that single character variable names are not meaningful, or they hinted at this but were too vague.

(ii) This question part was less well answered. Many answers struggled to clearly identify the issue with non-meaningful variable names.

Some incorrect answers stated that the programmer might use the variable names as literal values.

(iii) A wide range of answers was seen, this question part was generally well answered.

Question 2

(a) Most solutions were able to pick up marks here. One common error was that some candidates did not write the 'Yes' 'No' from the diamond decision.

Another point of note was getting the correct logic operator such as '`>`' or '`=`' so that the loop was run the correct number of times.

(b) This question referred to a single variable storing 10 pairs of numeric values using a single input statement. Many responses here referred to storing values in an array using an array which was a very common misconception.

Question 3

(a) Candidates found this question more challenging. The question gave an example of a linked list Abstract Data Type (ADT) with items stored within it. However, when asking about the initial state of the linked list before items were added many candidates tended to describe the diagram in front of them.

(b) This part of the question required candidates to use their knowledge of linked lists to fill in the gaps. Candidates performed better than the previous question, many could identify that the two variables are pointers of type integer. Some candidates were not able to apply their knowledge of linked lists to this scenario where two arrays were required, one to store the data and the other to store the pointer to the next item.

Question 4

Many solutions started with a correct function header, but in some cases, there was no function end (MP1).

Many candidates spotted that a loop was required, and some responses used a single loop whilst others used two loops. A common mistake was to `MOD` the value within the array Data rather than the index itself.

Many correct responses were seen for the Return value, although some candidates used `OUTPUT` here.

Question 5

(a) This question drew a range of marks. Some common mistakes were missing initialisation values for `Data[1]`, `Data[2]`, `Data[3]` and also missing quotation marks for strings.

(b) (i) Candidates found this question more challenging Many answers were trying to treat the case statement like an '`IF`' '`ELSE`'.

(ii) Most candidates could identify the line that we were looking for.

Question 6

(a) This question part attracted a range of marks. Candidates were generally very good at calculating all three lengths correctly. There were some mistakes in syntax such as ² instead of `^2`

(b) Many candidates identified the problem here. Some responses were not close enough to the solution.

Question 7

(a) (i) This question part was looking for advantages of applying abstraction to this scenario and not a definition of abstraction itself. Many responses were too vague, such as 'makes the problem easier' missing words such as 'to solve' which would have been enough to complete the answer.

(ii) This question part saw a range of marks awarded. It was important for candidates to use the information given in the question and not bring in new information that has not been given.

(iii) Generally, candidates were able to consider the scenario and find an operation that would be required when a text message is received back. Some responses showed a misunderstanding of the word 'operation' and incorrectly referred to 'operators' such as Boolean Logic operators, or other programming constructs.

(b) (i) Candidates did find it difficult to access both mark points. Many recognised that the diamond shape was indication that selection was taking place but failed to give all module names.

(ii) Some good attempts were seen by candidates with more success for the module headers for Sub-B. Some candidates missed the `BYREF` for Sub-A, and some used the module header 'Module' rather than recognising whether the Sub was a `FUNCTION` or `PROCEDURE`.

Question 8

(a) Successes seen in this question part were candidates recognising that a loop is required and using string manipulation to find substring '//'.

Candidates needed to be careful not to loop to final character when extracting two characters so that their programs would not crash if no comment was found in the string.

Some success for simply appending a character from one string to the next until the '`//`' was found or end of string reached was nice to see.

Some confusion was seen where candidates used `OUTPUT` instead of `RETURN` or tried to open and close files unnecessarily.

Some empty responses were seen.

(b) Some pseudocode difficulties were seen opening and closing files, such as using quotation marks around the file name, or opening files in the incorrect mode.

Some success was seen of candidates following the stages correcting.

When counting, it is important that variables are initialised to `0` first.

Some empty responses were seen.

COMPUTER SCIENCE

Paper 9618/22

Fundamental Problem-solving and Programming Skills

Key messages

This paper addresses the application of practical skills including both computational and algorithmic thinking. This often involves analysing and understanding the requirements of a scenario as well as designing and presenting a solution. Candidates need to be able to identify the key elements of each requirement which, for example, could include the need for an iterative structure or methods to access data stored in a string or a file. The development of these skills requires practice.

Candidates need to follow the recommended pseudocode to communicate their solution to the Examiner. This will ensure that the Examiner will be able to follow the structure and logic of the pseudocode algorithm presented and credit solutions accordingly.

Candidates in preparation for this component may have been introduced to practical programming in one of the supported languages for Paper 4. The candidate needs to be aware of the differences in syntax and appreciate that if the question asks for pseudocode, then variations with their studied programming language will be unacceptable for this component.

This subject makes use of many technical words and phrases. These have specific, defined meanings and they need to be used correctly.

Answers should be as precise and specific as possible. Candidates should familiarise themselves with the meanings of the command words used in this paper and form their answers accordingly. Candidates need to read each question carefully to make sure they understand what is being asked. Candidates should also be aware that answering a question by simply repeating phrases from the question will not gain marks.

Candidates should be encouraged to attempt all questions. Even the more advanced questions that ask for an algorithm to be written in pseudocode contain the accessible marks.

General comments

Familiarity with fundamental programming concepts for example the confusion between a literal value and an identifier, or the misuse of `OUTPUT` in place of `RETURN`.

Several candidates find the use of parameters challenging, often replacing parameters to a subroutine with a series of prompt and input statements within the body of the subroutine itself.

Several candidates find the concept of reserved keywords challenging and candidates often use these as identifiers in their pseudocode.

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

The following invalid pseudocode constructs and statements have been seen in this series:

- Invalid variable names:
  ```
  DECLARE String : STRING
  ```
  Using reserved words, such as variable data types, as variable names.

- Incorrect use of arithmetic operator symbols:

  ```
  IF Side1 X Side1 = Side2 X Side2 + Side3 X Side3 THEN
  ```

  Instead of:

  ```
  IF Side1 * Side1 = Side2 * Side2 + Side3 * Side3 THEN
  ```

- Incorrectly formed arithmetical expressions where brackets are missing or not correctly used:

  ```
  Average ← Num1 + Num2 + Num3/3
  ```

  Instead of:

  ```
  Average ← (Num1 + Num2 + Num3)/3
  ```

<u>Comments on specific questions</u>

Question 1

(a) The majority of candidates gained at least one mark with many gaining all four marks. Rows 2 and 3 being the most commonly marks given.

(b) Many full-mark answers seen. Some candidates found this more challenging and wrote the compete function headers from the Insert rather than just giving the function name.

Common mistakes seen:

- Choosing a function that did not deliver a result of the correct type e.g. `IS_NUM()` for first function on the final row.
- Incorrectly naming functions from the Insert e.g. `NUM_TO_STRING()`.

(b) Few full mark answers; many candidates gained one of the two marks available.

Several candidates found this question challenging giving answers along the lines of 'store them in an array/record/file'. Some suggested the variable could be 'document in an array' another commonly seen incorrect answer was the use of 'comments'.

Question 2

(a) Few candidates gained full-marks for their solution to this question, however, only a small number of zero mark solution were seen.

Common mistakes:

- Test syntax:

  ```
  Num1 > Num2 AND Num3
  ```
  or
  ```
  Num1 > Num2, Num3
  ```

- Reversing the assignment: `Set Num1 to Ans`.
- Missing labels on arrows from second decision diamond.
- Missing brackets on arithmetic expressions : `Num1 + Num2 + Num3/3`.
- Lack of separators in `OUTPUT` statement and sometimes omission of keyword `OUTPUT` .

(b) A small number of very good answers but very many less focused approaches which struggled to capture any element of the given flowchart. Many solutions lacked any nested structure.

Common mistakes:

- Unterminated clauses.
- Interleaved rather than nested clauses.
- Use of jumps to exit a pseudocode block e.g. `BREAK` and `ENDPROCEDURE`.
- Incorrect use of the `GetStat()` function, including:
  - `CALL GetStat(Flag)`
  - `GetStat() → Flag`

Question 3

(a) (i) Well-answered by most candidates.

Common mistakes:
- Omitting the keyword `DECLARE.`
- Omitting `ENDTYPE` (or equivalent).
- Adding `DECLARE` before the `TYPE` header.
- Using dot notation for the individual data items.
- Defining a range of values for the two `Limit` data items.

(ii) Well-answered by many candidates.

Common mistakes:
- Missing or incorrect array type.
- Use of a comma suggesting a 2D array.
- Omitting keyword `DECLARE.`

(b) Candidates found this question challenging. The majority of candidates referred only to the use of an 'array' rather than to an 'array of records' as given in the question. Weaker responses answered along the lines of 'the data is easier to access'.

MP3 provided a 'program advantage' mark; a larger number of candidates used 'easy' rather than 'easier'.

Question 4

Most candidates managed to gain at least a two of the marks available for this question.

The majority of solutions included the basic 'building' blocks of the algorithm:

- Input the three values.
- attempt to determine the longest side
- perform at least one of the required tests
- output one of two messages.

Common mistakes:
- Passing parameters.
- Omitting the prompt before the input.
- Using `'x'` rather than `'*'` as the multiplication operator.
- Omitting keyword `ENDIF` from the clause which selects between the two possible `OUTPUT` statements MP5.

- Not common, but the attempt to output a Boolean value in a comma separated list was seen on a few occasions.

The incorrect use of the logical operator `AND` for continuation was seen regularly when determining the longest side, in statements such as:

```
Longest ← Num1 AND Side1 ← Num2 AND Side2 ← Num3
```

Some solutions used an array for the three input values. This was unnecessary for just three values and in many cases the increased complication introduced errors.

Some unnecessarily long solutions were seen. It should be noted that the answer lines provided indicate the length a solution is expected to take and when a candidate's solution is considerably longer than the answer lines provided then this usually indicates the approach being taken is not the most efficient one.

Question 5

(a) (i) A small number of full-mark answers were seen; most candidates managed to gain at least one of the two ‘syntax’ marks.

Although many candidates identified all three errors, often marks were lost by placing them in the wrong category. Candidates are expected to understand the difference between syntax, logical and run-time errors and be able to apply this to a given scenario.

A small number of answers only provided a corrected line of pseudocode, contrary to the wording of the question.

(ii) Only a very few two-mark answers were seen.

A significant number of candidates made no attempt at this question.

(b) Correctly answered by a number of candidates. Incorrect answers seen seemed to be picked at random from those in syllabus, and supplemented with error types such as ‘human’, ‘machine’, ‘value’, ‘interrupt, ‘system’ etc.

Question 6

(a) This challenging pseudocode question attracted a wide range of different answers from virtually no attempt to fully working solutions.

All but the weakest solution included at least one loop. The mark scheme requirement for a conditional loop prevented many count-controlled solutions gaining this mark as they lacked any immediate `RETURN` or `BREAK`.

As there were eight-mark points candidates using a count-controlled solution could still be awarded the full seven marks available for this question.

Mark breakdown as follows:

- MP1 was given in many cases, although the absence of `ENDFUNCTION` lost this on several occasions.
- MP2 was attempted in many solutions but often consisted of just the initial length test and lacked the corresponding `RETURN` and in very many cases also the corresponding `ENDIF`. When these were present, a common mistake was to return an uninitialised variable (e.g. `NewLine`) or one that had been initialised to an empty string rather than the original parameter.
- MP3 requiring a conditional construct and was given to many solutions.
- MP4 was awarded less frequently that MP2, the test which was often a length test was ‘one out’ so for example the solution would stop at index position 14 and then attempt to concatenate three dots.
- MP5 given for the correct use of an inner loop was not common. Where given, this mark was usually for testing for a space character.
- MP6 was awarded for a reasonable attempt usually involving the use of the `LEFT()` function.
- MP7 was rarely given as it effectively required a completely working algorithm apart form MP8
- MP8 was awarded to many candidates.

Common mistakes:
- missing end-of-clause keyword
- use of keyword `Length` as an identifier
- use of `'+'` to concatenate
- treating a string as an array.

The innovative solution of working back from string index position 14 was seen on a few occasions and often gained full marks.

(b) (i) Many candidates found this question challenging with just over 30 per cent achieving this mark.

Many vague and hopeful answers, such as those referring to ‘unexpected results’.

Of those on the right line, there were many vague references to ‘making the string bigger’ or ‘uses more space’ which were not markworthy.

A few answers referred to the spaces being ‘not needed’ which mapped to the ‘redundant’ point on the mark scheme.

(ii) Many candidates did not attempt to answer this question. However, a similar number of correct answers to the previous question were seen.

A small number of candidates suggested run length encoding which was considered markworthy when this was appropriately explained.

Some answers were considered too vague, such as ‘replace the zeroes with spaces’. Candidates should try to use clear and precise wording in their answer.

Incorrect answers included such suggestions as ‘store as binary’, ‘use an array ‘, ‘use hexadecimal’ and ‘store each sample on a new line’. Although the question stem, in the case of the last of these incorrect answers, made it clear the samples were all stored as a single line.

(iii) Less than 10 per cent of candidates gained this mark, however many made no attempt to answer this question.

Question 7

(a) Despite a small number of perfect answers, the requirement seems to have been missed by the majority. Most answers failed to focus on the requirements of the new module, clearly stated in the question stem, and instead suggested various modules that might be of use in the complete membership system.

A common mistake was to suggest modules that setup initial club member details or which added members to the waiting list.

A number of solutions referred to the sending of an email rather than a text as given in the requirements of the question.

A small number of candidates missed the point and suggested computing terms such as module types or design methodologies.

(b) (i) Well-answered by the majority of candidates, with many gaining full marks.

(ii) Only 45 per cent gained the mark available for this question.

Question 8

(a) Generally, found this question challenging.

Many wrong answers focused on the teaching activity mentioned in the scenario e.g. ‘the teacher can easily see the errors made by her candidates’. Answers should relate to Computer Science.

Although the mark scheme offered one ‘easier to’ mark this was not often given as in many cases the answer given was too vague.

(b) This pseudocode question attracted a wide range of different answers from virtually no attempt to fully working solutions.

Many ineffective solutions were seen with a large number of candidates making no attempt at this question.

A significant number of solutions included file handling operations. Many of these included a statement which read a value for `Line` from a file, and so lost a subsequent mark which included the use of `Line`.

A small number of smart solutions, gaining full marks, seen were based around a simple `WHILE` loop rather than the 'count then trim' approach taken in the original mark scheme. An example of this type of solution is shown below:

```
WHILE Line <> '' AND LEFT(Line, 1) = Space
    Line ← RIGHT(Line, LENGTH(Line) - 1)
ENDWHILE

RETURN Line
```

Mark breakdown for most solutions as follows:

MP1: Count-controlled loop to the length of Line was seen frequently. A common mistake was to omit the assignment arrow i.e.: `FOR index 1 to LENGTH(Line)`
In addition, `LEN()` was seen several times.

MP2: Together with MP1, the two marks most often given. Both `MID()` and `LEFT()` were seen used correctly.

MP3: Many count-controlled solutions simply counted every space so lost this mark.

MP4: Generally, the use of `RIGHT()` or `MID()` tended to gain this mark, as did those that attempted to build a new string character by character.

MP5: Few candidates gained this for the generation of a correct new string.

MP6: A number of solutions incorrectly used `OUTPUT`.

(c) This question contained a more recognisable scenario than the previous question and thus most candidates gained better marks.

Many very good solutions were seen.

The filename presented problems to many. Often the parameter identifier incorrectly had `'.txt'` added and it was very common for the identifier to be enclosed in double quotation marks when subsequently used.

Mark breakdown as follows:

MP1: Often given and in some cases the only mark awarded. Two occasional errors: suggesting the module was a `FUNCTION` and using the keywords `INPUT` and `OUTPUT` as file identifiers.

MP2: Rarely given. The output file was usually opened in `WRITE` mode. `CLOSEFILE` statements were regularly omitted or lacked a filename.

MP3: One of the 'recognisable' marks which together with MP1 and MP4 gave a three-mark base.

Common mistakes:
- Omitting the parameter from `EOF()`.
- Missing `ENDWHILE`.
- Use of a count-controlled loop: `FOR Index ← 1 to EOF(InputFile)`.
- Use of a python-like syntax: `FOR Line IN InputFile`.

MP4: One of the more accessible marks, one mistake seen occasionally was including a line index as a third parameter.

MP5: More challenging. Many attempts incorrectly included the use of keyword `CALL` and failing to use the value returned by the first function as the parameter to the second.

MP6: Even given the problems caused by failing to correctly use the functions in MP5, very many solutions failed to test the return string before performing the `WRITEFILE`. The usual approach was to test for an empty string immediately following the `READFILE` operation.

MP7: Many gained mark for testing original line read from file.

MP8: A straightforward final mark. Frequently given as follow through from MP7 if this mark point was lost due to lack of initialisation.

In some cases, the concatenation of the integer value with a message string lost this mark.

COMPUTER SCIENCE

Paper 9618/23

Fundamental Problem-solving and Programming Skills

Key messages

This paper addresses the application of practical skills including both computational and algorithmic thinking. This often involves analysing and understanding the requirements of a scenario as well as designing and presenting a solution. Candidates need to be able to identify the key elements of each requirement which, for example, could include the need for an iterative structure or methods to access data stored in a string or a file. The development of these skills requires practice.

Candidates need to follow the recommended pseudocode to communicate their solution to the Examiner. This will ensure that the Examiner will be able to follow the structure and logic of the pseudocode algorithm presented and credit solutions accordingly.

Candidates in preparation for this component may have been introduced to practical programming in one of the supported languages for Paper 4. The candidate needs to be aware of the differences in syntax and appreciate that if the question asks for pseudocode then variations with their studied programming language will be unacceptable for this component.

This subject makes use of many technical words and phrases. These have specific, defined meanings and they need to be used correctly.

Answers should be as precise and specific as possible. Candidates should familiarise themselves with the meanings of the command words used in this paper and form their answers accordingly. Candidates need to read each question carefully to make sure they understand what is being. Candidates should also be aware that answering a question by simply repeating phrases from the question will not gain marks.

Candidates should be encouraged to attempt all questions. Even the more advanced questions that ask for an algorithm to be written in pseudocode contain the accessible marks.

General comments

Familiarity with fundamental programming concepts is vital. Lack of understanding is often illustrated by the confusion between a literal value and an identifier, or the misuse of `OUTPUT` in place of `RETURN`.

Several candidates find the use of parameters challenging, often replacing parameters to a subroutine with a series of prompt and input statements within the body of the subroutine itself.

Several candidates find the concept of reserved keywords is not well understood and candidates often use these as identifiers in their pseudocode.

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

The following invalid pseudocode constructs and statements have been seen in this series and resulted in marks being lost:

- Invalid variable names:
  ```
  DECLARE Integer : INTEGER
  DECLARE Input : STRING
  ```

Using reserved words such `INPUT` or data types as variable names.

- Invalid use of user-defined data types:
  `Batch.Weight(Index) > Max`

  Accessing record data items using incorrect syntax.

- Incorrectly formed arithmetical expressions where brackets were missing or not correctly used:
  `Average ← TotalA + TotalB/2`

  The above was often seen instead of:
  `Average ← (TotalA + TotalB)/2`

Candidates need to read each question carefully before attempting to answer it. The importance of clearly understanding the question before attempting to answer it cannot be over-emphasised. Questions may address topics in various ways, and it is often necessary to apply knowledge in a specific way if marks are to be gained. For example, if a scenario is given in a question, then to gain full marks candidates may have to frame their answer with reference to the scenario.

If answers are crossed out, the new answers must be written clearly so that the text may be read easily, and the correct mark awarded.

Many candidates make use of blank pages for rough work when preparing their final answer. In these cases, it is extremely helpful if this text is crossed out.

Comments on specific questions

Question 1

(a) (i) This question required candidates to give two benefits of using a modular approach when writing a program. Many candidates failed to recognise what was being asked so struggled to gain any marks, a significant number made no attempt at this question.

Where marks were awarded, they were spread fairly evenly over the four-mark points available.

Several references to 'library routines' suggested the question had not been fully understood.

(ii) Marks were most often given for reference to parameters and return value. Often these were the only marks awarded.

Several weaker answers were seen where the question was clearly not understood.

Answers frequently came close to gaining one of the MPs but fell short by:
- Omitting the reference to the using the code/program in MP1.
- Referring to 'calling when needed' but leaving out the part about replacing the original code where it appeared that was required to gain MP2.

(b) Most candidates gained full marks for this question.

Where a mistake was made, this was often in the second row often giving the answer as `LENGTH` instead of `DAY`.

Question 2

(a) A wide range of marks were given; a number of candidates did not attempt the question.

Many solutions borrowed heavily from the wording of the question and reference to lines being 'copied' was frequently seen. These were generally not considered markworthy.

MP1 was given in most cases where an attempt had been made.

MP2 was slightly less common, with reference to variables either omitted or limited to just the declaration.

Attempts at MP3 were at times vague, and it was unclear which steps were included in the loop.

MP4 it was unusual to read a description that clearly related to reading a single line from the file.

MP5 was addressed completely by some, but many suffered from the use of ‘copy’ or missed the reference to the array index relying on simply ‘next’ which was not markworthy.

MP6 was given frequently.

(b) Of the two answer options, ‘conditional loop’ was by far the more popular.

Marks were often lost for not using the required term and for not referring to the scenario when describing the ‘Use’.

Question 3

(a) (i) A significant number of answers gained full marks with most candidates gaining some marks for this question.

The common mistakes:
- incorrect field dot notation
- incorrect use of > and > =
- incorrect use of < and < =
- Use of literal vales, often the upper and lower bounds of the array.

(ii) Just under 20 per cent of candidates gained the mark for this question.

A common mistake was to refer to setting an element, rather than a field, to a particular value.

It seemed that ‘indicate’ had been interpreted as ‘count’ on many occasions, with answers along the lines of ‘iterate through the array and record the empty indices’

(b) Well-answered by many but a few no attempts.

Common mistakes:

Zone 1: Often only `Count` was initialised or `Index` was initialised to `0` rather than to `1`.

Zone 2: Occasionally omitting the decision symbol output labels.

Zone 3: Testing `Index` rather than `InRange(Index)`.

Zone 4: When attempted this mark was usually awarded.

Zone 5: Several incorrect tests, usually `Count >= 5` and occasional missing output labels.

Question 4

(a) Most candidates managed to gain at least a few mark for this question.

MP1, MP2 and MP3 were three easy separate marks and were given to most reasonable attempts.

High-mark solutions usually used the straightforward two-loop solution (as per the mark scheme example). Solutions that attempted a single loop with a flag to indicate which total was being calculated often found that the additional complexity introduced errors.

Solutions that interpreted the question as meaning that two separate averages were required found that the additional complexity introduced an error, usually in the form of the count being one out.

Common mistakes:

- Declaring `TotalA` and `TotalB` as local variables and occasionally using different variables.
- Failing to initialise `TotalA` and `TotalB`.
- Failing to input a value either before the loop (in the case of a `WHILE` loop) or within the loop itself.
- Attempting to concatenate an integer with a string in the `OUTPUT` statement.

In a few cases the question had been misunderstood and the algorithm attempted to take the whole sequence as a parameter (or a single `INPUT`) and attempt to extract the individual values.

(b) (i) Many single marks given for the simple mention of 'an array', much fewer second marks were awarded for referring the size and type of array.

'2D array' was seen frequently, as was 'record and 'file' plus a selection of data types and ADTs, none of these were markworthy.

(ii) Many full mark answers. The MP2 'easier to' mark was particularly accessible.

Many marks were lost through claiming that something was 'easy' to do with an array and a small number who state that an array can 'return' a value.

Often attempts at MP3 failed to make it clear that it was the program/algorithm that was being described.

Question 5

(a) The two-part textbook answer was seen on many occasions.

Although the question was clearly focused on the loop construct, 'selection' was still suggested by a few candidates.

Many answers referred to the fact that `Index` would not need to be incremented using an additional statement which was not mark worthy.

(b) A small number of very good answers but generally most candidates did not understand that the function used the same arguments so only needed to be called once before the loop.

Many candidates incorrectly suggested 'the two functions should be combined the functions into one' and several answers mistakenly claimed that the `CASE` clause was inefficient because it's 'more complex' than an `IF` clause.

Question 6

(a) Some very good full-mark answers were seen based on both of example solutions given.

At the other end of the scale some answers were little more than the module header, however the number of no attempts was small.

Many solutions based on a 'selection' algorithm using discrete `IF` ... `ENDIF` clauses and while these were often functionally correct (and therefore given appropriate marks) they tended not to fit on the given answer lines. Some answers contained three dots to indicate missing clauses, but this then lost MP4.

Common errors:

- Use of `MOD()` rather than `DIV()`.
- Omitting `NUM_TO_STR()`.
- Use of `'+'` for concatenation.
- Incorrect `CASE` 'syntax', notably including the `CASE` variable in each condition, often in some sort of `IF` statement.

(b) (i) Many full-mark answers seen. A small number of candidates made no attempt to answer the question.

Common mistakes:

- Omitting one of the parameters.
- Defining the module as a procedure.
- Returning an identifier e.g. RETURNS Filename rather than a data type.

(ii) Around 40 per cent of answers mapped to the mark scheme, indicating that these candidates had a very good understanding of the scenario and could anticipate the effect of the suggested change.

Many candidates resorted to standard 'easier to...' answers, giving a range of incorrect 'benefits'.

Question 7

(a) (i) Just under 70 per cent of candidates gained this straightforward mark.

(ii) Vague answers were usually seen with few gaining one mark.

The answer required for one mark needed to be reasonable explanation of the difference between passing by value and passing by reference, but most candidates answers were too vague. For two marks they needed to refer to the scenario given.

Candidates struggled with the concept of the 'subsequent value' as used in the calling module and 'original value' was seen frequently but the description was rarely considered completely adequate.

(b) Just under 30 per cent of candidates gained all the six marks available for this question.

Around 25 per cent answers suggested candidates had no knowledge of the topic or made no attempt at the question.

Common mistakes:

- Missing second arrow on `BYREF` parameter.
- One or more missing annotations (MP4 and MP5).

Question 8

(a) Over half of the candidates gained at least half of the marks available for this question with around 12 per cent gaining the maximum seven marks available. At the other end of the scale there were around 24 per cent candidates who gained no marks for this question.

General comments

MP1: In weak solutions this was often the only mark given. `ENDFUNCTION` was omitted occasionally, and the use of `PROCEDURE` was seen.

MP2: Often the length test was included together with other sub-string operations, often correctly. Where it was included at the start of the algorithm in a separate `IF` ... `ENDIF` clause it was not uncommon to return either the original parameter string or to omit the return altogether.

MP3: A 'design' mark awarded in the majority of cases where a reasonable attempt has been made to test the first 'word' as opposed to just a single character.

MP4: Awarded in many cases. A significant number of solutions did not perform the full check as specified although several tested if first 9 characters was '`PROCEDURE`' and then tested if the 10th character was a space and the same for '`FUNCTION`'.

MP5: Included in many solutions. This MP was sometimes awarded in cases when the solution did not justify MP3 or MP4.

MP6: A challenging mark that required a correct length calculation and avoiding the use of a case-converted string.

MP7: The final `RETURN` was occasionally omitted but this mark was given to most reasonable attempts.

Common mistakes:

- Attempting to treat the string parameter as an array.
- Use of `LEN()` rather than `LENGTH()`.
- Extracting elements from the `ModInfo` array for use in the algorithm.
- The use of `OUTPUT` rather than `RETURN`.

(b) Over 40 per cent candidates gained no marks for this question many of these made no real attempt at an answer. However, just over 40 per cent of candidates gained at least half marks with around 20 per cent being awarded 7 or 8 marks

The filename presented problems to many candidates. Often the parameter incorrectly had '.txt' added and it was very common for the identifier passed as a parameter to be enclosed in double quotation marks when subsequently used.

Mark breakdown as follows:

MP1: Often given, although the `CLOSEFILE` statement was regularly omitted or lacked a filename, though this has improved over the previous few series.

MP2: One of the 'recognisable' marks.

Common mistakes:

- Omitting the parameter from `EOF()`.
- Missing `ENDWHILE`.
- Use of a count-controlled loop: `FOR Index ← 1 to EOF(InputFile)`.
- Use of a python-like syntax: `FOR Line IN InputFile`.

MP3: Another 'recognisable'/accessible mark. One mistake seen occasionally was the inclusion of a line index as a third parameter. As this was a two-part mark point it relied on the count variable having been initialised.

MP4: The function `Header` was used correctly in many solutions and the return value used. Some attempts incorrectly included the use of keyword `CALL`.

MP5: Several solutions tested the first character of the return value for `"P"` or `"F"` without realising that the return value might have been an empty string and therefore there would not have been a first character and consequently the substring operation might generate an error.

MP6: Many solutions gained this mark for three assignments to correctly referenced array elements

MP7: This mark was frequently lost due to the lack of conversion, using `NUM_TO_STR()` for the assignment to column 1

MP8: Initialisation of count/index variables this mark was often given.

COMPUTER SCIENCE

Paper 9618/31
Advanced Theory

Key messages

Candidates are required to demonstrate a detailed study of the topics covered by the syllabus using technical terminology as appropriate for this advanced theory paper. Candidates who have studied the relevant theory, and who have also practiced and used the relevant tools and techniques, are more likely to be able to solve the problems set on the examination paper.

Candidates are advised to answer each question in an appropriate manner for the command word of the question; for example, a question beginning with 'explain' requires more detail than a question beginning with 'identify'. If a question asks for working to be shown, candidates must also ensure that they do this, 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 or algorithm construction, and answer questions requiring pseudocode answers 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 and to make sure they answer the question that is asked.

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 advised to use the correct computer science terminology when answering questions, to maximise their marks. For example, with file storage and access questions, the distinction between records and files is important.

Comments on specific questions

Question 1

(a) The vast majority of candidates demonstrated their ability to convert a normalised floating-point binary number to its denary equivalent. Candidates who showed sufficient working so that it was clear how they used the exponent and mantissa to calculate the final value, along with a correct final value, achieved all three marks.

(b) The vast majority of candidates achieved at least one mark for converting a negative denary number to its normalised binary equivalent. Candidates who gave the correct normalised floating-point value along with evidence of how they arrived at their answer, achieved full marks.

Question 2

(a) Most candidates achieved one or both marks for arranging the given TCP/IP protocol suite layers in the correct order.

(b) The vast majority of candidates achieved at least one mark for describing the function of the Transport layer of the TCP/IP protocol suite.

(c) The vast majority of candidates gained marks for naming a protocol used in the Application layer, with many of these candidates going on to achieve a second mark for an expansion statement on their given protocol.

Question 3

(a) Most candidates were able to explain what is meant by non-composite data types and/or composite data types, with a high proportion of these candidates achieving marks for both types.

(b) Virtually all candidates achieved marks for declaring the given record data type, with many of these candidates achieving high marks. A minority of candidates incorrectly declared the telephone number field as an integer rather than a string, and the number of members field as a string, rather than an integer.

Question 4

(a) Candidates who applied the correct terminology so that it was clear that they were accessing a record within a file, by searching through them one after the other from the start of the file until the record was found, achieved the marks.

(b) Candidates who wrote separately about serial files, where records are stored chronologically, and sequential files, where records are stored in order of a key field, and explained file access in those contexts achieved the marks. It was also important to explain that it was a record within the file that was being accessed, rather than simply accessing the file.

Question 5

(a) This question was generally well answered with the vast majority of candidates achieving at least one mark, and a high proportion of these candidates achieving more than one.

(b) This question was also well answered with most candidates achieving marks.

(c) Another well answered question, with most candidates achieving marks. Those who accurately showed the process of a Reverse Polish Notation (RPN) expression being evaluated using a stack, and who included all the steps, achieved the highest marks.

Question 6

(a) Many candidates achieved high marks on this part. A truth table representing a logic circuit had to be completed, with all the correct working, to achieve all the marks.

(b) Candidates who simply wrote the correct Boolean terms as a sum-of-products for the given logic circuit, without attempting to simplify it, achieved the marks. Most candidates gained at least one mark here.

(c) (i) Most candidates correctly completed the Karnaugh map (K-map) for the given expression, with candidates who correctly filled every square with either a 0 or a 1, achieving both marks.

(ii) Candidates who correctly identified two loops of four 1s achieved full marks.

(iii) Candidates who correctly wrote the Boolean expression as a simplified sum-of-products directly from the loops in their K-map, without any further attempts at simplification, achieved this mark.

Question 7

(a) Most candidates were able to score at least one mark for describing a digital certificate, including some or all of: it is an electronic document used to authenticate the online identity of an organisation, and it is issued by a Certificate Authority.

(b) A large proportion of candidates were able to explain that a digital certificate provides the public key but very few went on to add that this public key is then used to validate an organisation's private key.

Question 8

(a) The vast majority of candidates demonstrated their ability to write clauses for a declarative programming language, with virtually all candidates achieving some marks for this question. The candidates who remembered that each clause needed a full stop at the end and could not have any capital letters in any of the text, achieved the highest marks.

(b) The vast majority of candidates were able to correctly show the output from the given goal without introducing any capital letters to the list and without placing a full stop at the end.

(c) The last part of the question was more difficult, requiring a rule to be constructed from the given information. Candidates generally performed well on this question with the full range of marks seen.

Question 9

Candidates who gave responses that were specifically related to Deep Learning, rather than generic answers that could be applied to many areas of artificial intelligence (AI), attained the highest marks. Most candidates recognised that this type of AI contains many layers, including input, hidden and output, to create an artificial neural network. Some went on to add that the larger the number of hidden layers, the more successful the output.

Question 10

(a) A well answered question with candidates generally being aware that the data in an array needs to be sorted for a binary search to be able to work.

(b) A well answered question, completing the missing parts of a search algorithm, with many candidates achieving the higher marks.

(c) Candidates were mostly able to state the Big O notation for a binary search, with the some candidates also being able to describe what this Big O notation actually means.

Question 11

(a) Candidates who stated features of Reduced Instruction Set Computers (RISC) processors, without trying to compare them to something else, achieved the marks. Most candidates achieved at least one mark.

(b) Candidates were required to simply outline the process of interrupt handling in general terms in this question. Many candidates understood this and achieved some of the marks. However, a number of candidates appeared to find the question part difficult.

(c) Many candidates understood that pipelining would add complexity to interrupt handling and some of these candidates went on to give a good explanation as to how this might work. However, high marks for this question were rare.

COMPUTER SCIENCE

Paper 9618/32
Advanced Theory

Key messages

Candidates are required to demonstrate a detailed study of the topics covered by the syllabus using technical terminology as appropriate for this advanced theory paper. Candidates who have studied the relevant theory, and who have also practiced and used the relevant tools and techniques, are more likely to be able to provide appropriate responses to the questions set on the examination paper.

Candidates are advised to answer each question in an appropriate manner for the command word of the question; for example, a question beginning with 'explain' requires more detail than a question beginning with 'identify'. If a question asks for working to be shown, candidates must also ensure that they do this, 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 or algorithm construction, and answer questions requiring pseudocode answers 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 and to make sure they answer the question that is asked.

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 advised to use the correct computer science terminology when answering questions, to maximise their marks. For example, with file storage and access questions, the distinction between records and files is important.

Comments on specific questions

Question 1

(a) The vast majority of candidates were aware that re-distributing the bits between the mantissa and exponent affected the precision and range of the number stored. Candidates who stated that increasing the number of bits in the mantissa increased the number's accuracy while reducing the number of bits in the exponent reduced its range, or vice versa, achieved the marks.

(b) The vast majority of candidates achieved at least one mark for this question. Candidates who gave the correct normalised floating-point value along with evidence of how they arrived at their answer, achieved full marks.

Question 2

(a) Many candidates achieved one mark for this question with some achieving both. The most common correct answer seen was the importance of the use of protocols where the systems that were communicating were based on different platforms.

(b) The vast majority of candidates were able to state at least one protocol associated with sending and/or receiving emails. Many of these candidates correctly identified two protocols.

(c) The vast majority of candidates achieved at least one mark because they were able to give good descriptions of the operation of BitTorrent as a process for file sharing. However, candidates who gave good descriptions of how BitTorrent provides peer-to-peer file sharing, for example, by sharing files directly between peers, without the use of a central file server, achieved more marks.

Question 3

(a) Virtually all candidates achieved at least one mark for this question, with many candidates achieving more than this. Candidates were mostly able to achieve one of the description marks and give an appropriate example of a non-composite data type.

(b) Candidates who had made use of the most recent version of the A Level Pseudocode Guide that accompanies this course were more likely to achieve the highest marks for this question by following the guidance in that document for the syntax of a set user-defined data type.

Question 4

Candidates who demonstrated a good understanding of asymmetric encryption by describing the acquisition of the receiver's public key in order to encrypt the document, then sending this cypher text to the recipient for them to decrypt it with their private key, achieved the highest marks. The question was generally well answered with the full range of marks, one to four, seen, including many with higher marks. However, relatively few candidates achieved the fourth mark.

Question 5

(a) This question was generally well answered with the vast majority of candidates achieving at least one mark.

(b) Another well answered question. Candidates who took the most care to accurately show the process of a Reverse Polish Notation (RPN) expression being evaluated using a stack, without missing any steps, achieved the highest marks.

(c) The vast majority of candidates achieved one or two marks for this question, with only a few candidates achieving all three marks.

Question 6

(a) A generally well answered question, with many candidates achieving high marks for correctly completing a truth table for a logic circuit, including the working.

(b) The vast majority of candidates who simply wrote the correct Boolean terms as a sum-of-products for the given logic circuit, without attempting to simplify it, achieved the best marks.

(c) (i) Most candidates correctly completed the Karnaugh map (K-map) for the given expression, with those who correctly filled every square with either a 0 or a 1, achieving both marks.

(ii) Candidates who correctly identified two loops of four 1s achieved both marks.

(iii) Candidates who correctly wrote the Boolean expression as a simplified sum-of-products directly from the loops in their K-map, without any further attempts at simplification, achieved this mark.

Question 7

(a) Candidates who applied the correct terminology so that it was clear that they were accessing a record within a file, without other records being read, along with an expansion of how this may be achieved, gained the marks.

(b) (i) Many candidates answered this question as though sequential access was being applied, whereas the question asked for an explanation of how direct access may be applied to a sequential file. Candidates who described the role of an index of key fields achieved the marks.

(ii) Candidates generally fared better with this question so long as they wrote about a hashing algorithm being applied to the key field of a record and moved on from there, they achieved the marks. Some candidates incorrectly stated that the hashing algorithm was applied to the file.

Question 8

(a) The vast majority of candidates achieved at least one mark for this question, with many candidates achieving higher marks, however, full marks were rarely achieved.

(b) This question was generally well answered. Candidates who were able to give good descriptions of how the linear and binary search routines compared, along with their respective Big O notations, achieved the marks.

Question 9

(a) Most candidates achieved some marks for stating benefits and limitations of a virtual machine. Those candidates who gave two distinct benefits and two distinct limitations achieved the most marks. A few candidates incorrectly mistook virtual machines for virtual memory, giving benefits and limitations of the latter.

(b) Most candidates achieved at least one mark for explaining the roles of the host and/or guest operating systems as used in a computer running a virtual machine. A number of these candidates were able to give very clear explanations, allowing to achieve all or most of the marks.

Question 10

(a) The vast majority of candidates demonstrated their ability to write clauses for a declarative programming language, with virtually all candidates achieving some marks for this question. The candidates who remembered that each clause needed a full stop at the end and could not have any capital letters in any of the text, achieved the highest marks.

(b) Due to an issue with question 10b, full marks have been awarded to all candidates for this question to make sure that no candidates were disadvantaged. There was an error in question 10b using `today` rather than the `choice`. This has been corrected in the published version of the paper.

(c) The last part of the question was more difficult, requiring a rule to be constructed from the given information. Candidates generally performed well on this question with the full range of marks seen.

Question 11

Candidates who gave responses that were specifically related to Reinforcement Learning, rather than generic answers that could be applied to many areas of artificial intelligence (AI), attained the highest marks. The majority of candidates recognised that this type of AI depends on feedback in the form of rewards and punishment. Those who could expand on that aspect achieved more than one mark.

COMPUTER SCIENCE

Paper 9618/33
Advanced Theory

Key messages

Candidates are required to demonstrate a detailed study of the topics covered by the syllabus using technical terminology as appropriate for this advanced theory paper. Candidates who have studied the relevant theory, and who have also practiced and used the relevant tools and techniques, are more likely to be able to solve the problems set on the examination paper.

Candidates are advised to answer each question in an appropriate manner for the command word of the question; for example, a question beginning with ‘explain’ requires more detail than a question beginning with ‘identify’. If a question asks for working to be shown, candidates must also ensure that they do this, 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 or algorithm construction, and answer questions requiring pseudocode answers 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 and to make sure they answer the question that is asked.

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 advised to use the correct computer science terminology when answering questions, to maximise their marks. For example, with file storage and access questions, the distinction between records and files is important.

Comments on specific questions

Question 1

(a) The vast majority of candidates demonstrated their ability to convert a normalised floating-point binary number to its denary equivalent. Candidates who showed sufficient working so that it was clear how they used the exponent and mantissa to calculate the final value, along with a correct final value, achieved all three marks.

(b) The vast majority of candidates achieved at least one mark for converting a negative denary number to its normalised binary equivalent. Candidates who gave the correct normalised floating-point value along with evidence of how they arrived at their answer, achieved full marks.

Question 2

(a) Most candidates achieved one or both marks for arranging the given TCP/IP protocol suite layers in the correct order.

(b) The vast majority of candidates achieved at least one mark for describing the function of the Transport layer of the TCP/IP protocol suite.

(c) The vast majority of candidates gained marks for naming a protocol used in the Application layer, with many of these candidates going on to achieve a second mark for an expansion statement on their given protocol.

Question 3

(a) Most candidates were able to explain what is meant by non-composite data types and/or composite data types, with a high proportion of these candidates achieving marks for both types.

(b) Virtually all candidates achieved marks for declaring the given record data type, with many of these candidates achieving high marks. A minority of candidates incorrectly declared the telephone number field as an integer rather than a string, and the number of members field as a string, rather than an integer.

Question 4

(a) Candidates who applied the correct terminology so that it was clear that they were accessing a record within a file, by searching through them one after the other from the start of the file until the record was found, achieved the marks.

(b) Candidates who wrote separately about serial files, where records are stored chronologically, and sequential files, where records are stored in order of a key field, and explained file access in those contexts achieved the marks. It was also important to explain that it was a record within the file that was being accessed, rather than simply accessing the file.

Question 5

(a) This question was generally well answered with the vast majority of candidates achieving at least one mark, and a high proportion of these candidates achieving more than one.

(b) This question was also well answered with most candidates achieving marks.

(c) Another well answered question, with most candidates achieving marks. Those who accurately showed the process of a Reverse Polish Notation (RPN) expression being evaluated using a stack, and who included all the steps, achieved the highest marks.

Question 6

(a) Many candidates achieved high marks on this part. A truth table representing a logic circuit had to be completed, with all the correct working, to achieve all the marks.

(b) Candidates who simply wrote the correct Boolean terms as a sum-of-products for the given logic circuit, without attempting to simplify it, achieved the marks. Most candidates gained at least one mark here.

(c) (i) Most candidates correctly completed the Karnaugh map (K-map) for the given expression, with candidates who correctly filled every square with either a 0 or a 1, achieving both marks.

(ii) Candidates who correctly identified two loops of four 1s achieved full marks.

(iii) Candidates who correctly wrote the Boolean expression as a simplified sum-of-products directly from the loops in their K-map, without any further attempts at simplification, achieved this mark.

Question 7

(a) Most candidates were able to score at least one mark for describing a digital certificate, including some or all of: it is an electronic document used to authenticate the online identity of an organisation, and it is issued by a Certificate Authority.

(b) A large proportion of candidates were able to explain that a digital certificate provides the public key but very few went on to add that this public key is then used to validate an organisation's private key.

Question 8

(a) The vast majority of candidates demonstrated their ability to write clauses for a declarative programming language, with virtually all candidates achieving some marks for this question. The candidates who remembered that each clause needed a full stop at the end and could not have any capital letters in any of the text, achieved the highest marks.

(b) The vast majority of candidates were able to correctly show the output from the given goal without introducing any capital letters to the list and without placing a full stop at the end.

(c) The last part of the question was more difficult, requiring a rule to be constructed from the given information. Candidates generally performed well on this question with the full range of marks seen.

Question 9

Candidates who gave responses that were specifically related to Deep Learning, rather than generic answers that could be applied to many areas of artificial intelligence (AI), attained the highest marks. Most candidates recognised that this type of AI contains many layers, including input, hidden and output, to create an artificial neural network. Some went on to add that the larger the number of hidden layers, the more successful the output.

Question 10

(a) A well answered question with candidates generally being aware that the data in an array needs to be sorted for a binary search to be able to work.

(b) A well answered question, completing the missing parts of a search algorithm, with many candidates achieving the higher marks.

(c) Candidates were mostly able to state the Big O notation for a binary search, with the some candidates also being able to describe what this Big O notation actually means.

Question 11

(a) Candidates who stated features of Reduced Instruction Set Computers (RISC) processors, without trying to compare them to something else, achieved the marks. Most candidates achieved at least one mark.

(b) Candidates were required to simply outline the process of interrupt handling in general terms in this question. Many candidates understood this and achieved some of the marks. However, a number of candidates appeared to find the question part difficult.

(c) Many candidates understood that pipelining would add complexity to interrupt handling and some of these candidates went on to give a good explanation as to how this might work. However, high marks for this question were rare.

COMPUTER SCIENCE

Paper 9618/41
Practical

Key messages

The only document that should be submitted for each candidate is the evidence document. No other files, including source files or program code, should be submitted. This document must not be in a zip folder. Candidates need to include their name and candidate number in the header of the evidence document.

Candidates need to copy their program code into each required box in the evidence document, this should be copied and **not** a screenshot. This is because the colours used by IDEs are not always clearly visible on the documents, especially if there is a black background. Screenshots that are too small are also illegible. If the program code cannot be read due to the screenshot being too small, or inappropriate colours used, then marks will not be awarded. Screenshots of the outputs of testing should be either white text on a black background or black text on a white background.

When copying code candidates need to make sure it is all visible on screen and that the boxes are not expanded beyond the width of the page; any code that is not visible on the document cannot be awarded marks.

Candidates also need to make sure all of their code is included and not cut off, especially the start of subroutines and that indentation is appropriate when using Python.

General comments

Candidates showed a good understanding of Object-Oriented Programming (OOP), often accurately creating a class, constructor and the get and set methods.

Candidates do need to make sure they are meeting each requirement of the question, for example using correctly parameters if instructed, or storing an outputting a return value if that is given as a task in the question.

More candidates were attempting later question parts even if an earlier question part was not done accurately.

When producing a screenshot for testing candidates need to make sure all data is visible, that includes the data that they have input. Without this input data it is not possible to check whether the candidate has used these inputs to create the given output.

Comments on specific questions

Question 1

This question required candidates to input data into an array and then manipulate this data using standard sorting and searching methods.

(a) Many candidates were able to accurate declare the 1D array with the appropriate space and data type identified. Some candidates provided a comment that showed their intention to create an array but did not actually create an array.

(b) Candidates were often able to accurately output a message to prompt for the input and then read in this input. Some candidates were then able to use this input value within a loop to iterate the number input quantity of times, reading in an integer each time and storing it in the array.

The stronger responses used suitable validation for the first number input, often by looping until this input was valid. Some candidates inaccurately validated the numbers that were input within the loop, attempting to restrict these integers to be between 1 and 20 (inclusive).

(c) (i) Many candidates were able to call `Initialise()` accurately. There were a range of methods of outputting the contents of the array, in Python candidates often output the array direct, whilst other languages required a loop through each element and output of each element.

(ii) This question required candidates to enter the data in the order given. Part **(b)** required validation, so the first input of 30 needed to be rejected and then the number 5 being entered to indicate that 5 numbers were to be input. Some candidates incorrectly included an additional input of 7 and then input all of the numbers as data to store in the array.

(d) (i) Candidates were often able to produce a bubble sort algorithm. There were a range of approaches including candidates who used two count-controlled loops and some who create more efficient algorithms that stopped as soon as the array was ordered.

(ii) Many candidates were able to call the bubble sort procedure and then output the contents of the array again.

(iii) Candidates were often able to produce the correct result showing the values input and the output of the sorted array.

(e) (i) This question required candidates to write an iterative binary search algorithm. Some candidates did not meet these requirements due to them writing a recursive binary search algorithm.

Candidates who did attempt the iterative algorithm were often able to meet several of the requirements, for example the header and calculating the middle value. These candidates also often correctly identified when the value was found and returned this. Fewer candidates correctly updated the first and last pointers accurately or overwrote a found index to return –1 even when the data was found.

Some candidates inaccurately compared the middle index to the data to find, instead of comparing the element in the middle index of the array.

(ii) Candidates were often able to take the number to search for as an input and use this in the function call. Fewer candidates correctly output the return value from the function call. Some candidates chose to output a different message depending on the return value which was not required but still output the index correctly when found.

(iii) This question required evidence of two tests from the user, each tested needed to show the inputs and the output. Many candidates provided the correct outputs, but fewer candidates also included the inputs in their results.

Question 2

This question required candidates to write a program using Object-Oriented Programming (OOP) to store and access data.

(a) (i) Many candidates were able to accurately create the class `Tree` and declared the attributes with the appropriate data types. Candidates were also often able to create the constructor with the appropriate parameters that they then assigned to the parameters.

Some candidates attempted to initialise the values in the constructor to default values such as 0 instead of using the parameters.

(ii) Candidates were often able to accurately declare the get methods. Some common errors including passing a parameter to the functions or attempting to assign or overwrite the attribute in the get method.

(b) This question required candidates to read data in from an external text file. Candidates were often able to open the file accurately and read in the data. Fewer candidates made appropriate use of exception handling for if the file was not found; for example, by including the file opening within the try and then including closing the file external of the exception handling which would not be possible if the file had never been opened.

Many candidates accurately split the data that was read in the file and instantiated a new object to be stored in the array.

Few candidates closed the file that they opened.

(c) This procedure needed to take an object as a parameter to then be output. Some candidates took a parameter but did not use it as an object, for example using it as an array and then attempting to access elements instead of making use of the get methods to access the data.

The stronger responses accurately used the get methods to check the value of evergreen and used this to determine which message to output. Some candidates output a standard message and then checked this for the final statements, whilst others repeated the statement.

(d) (i) Many candidates were able to accurately call `ReadData()` but fewer candidates stored the value returned from this function and use it to call `PrintTrees()`. Some candidates did not include a parameter when calling `PrintTrees()`.

(ii) Many candidates were able to get the correct output for this question.

(e) (i) This question required candidates to take input from the user and compare it to find trees that had data meeting the requirements.

Many candidates were able to take the required inputs from the user. Candidates were often able to iterate through each element in the array, but fewer were able to access the attributes for each value. The stronger responses made appropriate use of the get methods to perform the comparisons on each tree object in turn.

Candidate were often able to create an array and assign the `Tree` objects that met the requirements to this array. Some candidate then sent each object to `PrintTrees()` when identified, whilst some candidates iterated through the new array after all objects were found.

Some candidates took the input of requirements within the loop and therefore required new requirements to be input each time an object was compared.

(ii) This question required the editing of the procedure previously created. Candidates were often able to take the required data as input. Fewer candidates were able to accurately calculate the number of years it would take for that tree to grow to its maximum height. Some of the weaker responses use a `Tree` object but without showing how this object was accessed i.e. how the correct tree that the user input was found in the array of acceptable trees. Validation was not a requirement for this question, but some candidates still implemented this to ensure the algorithm worked correctly.

Some candidates chose to round the number of years up, some candidates rounded it down, some candidates calculated the number of years and months whilst some output the result of the calculation, all of which were acceptable.

(iii) Candidates were often able to generate the correct outputs for the system. Some candidates did not include the input of the tree requirements and the tree selection in their screenshot and therefore it could not be proven that the correct data was input.

Question 3

This question required candidates to create a queue data structure with its appropriate enqueue and dequeue functions.

(a) Many candidates were able to create an array and initialise the head pointer and tail pointer. Fewer candidates were able to initialise all 20 elements of the array to a suitable null value.

(b) Candidates were often able to create the `Enqueue()` function and store the data in the correct position. Some candidates did not increment the tail pointer in the appropriate place and overwrote the last item because `QueueTail` points to the index of the last item and not the next empty space.

Some candidates attempted to check if each value in the array had something other than null stored to check if the array was empty, whilst some candidates attempted to compare the array to an empty array which would not work due to the required initialisation of each of the 20 elements in part **(a)**.

Some candidates recognised the need to change the head pointer if it was currently pointing to –1 to become 0 and including this check and change in their solution.

(c) Few candidates were able to identify how to check if the queue was empty, with many only checking the initial empty queue. This did not consider if elements had been enqueued and dequeued previously which would mean that the head pointer was no longer storing –1.

(d) (i) Candidates were provided with a check digit calculation and an example that they had to implement for data input from the user. Candidates were often able to do this calculation accurately with a number of different methods used to access each alternate value in the input. The stronger responses also checked whether the result was 10 and made the comparison to x instead of a number.

Some candidates did not take 10 numbers from the user, instead taking 10 characters as input and attempting to use these in the calculation. Some candidates only took one input from the user.

Only the inputs that were valid based on the check digit were to be stored in the array and they were to be stored without the check digit, some candidates stored the full number input by the user and did not remove the check digit.

Fewer candidates correctly counted and output the number of invalid data items that were input by the user.

(ii) Candidates were often able to accurately call `StoreItems()` and `Dequeue()`. Fewer candidates stored the return value from `Dequeue()` and then output an appropriate message depending on the value, often by not returning the returned value instead stating that a value was removed.

(iii) Few candidates gained the correct output, often due to not outputting the number of invalid items or by outputting a different value as the result of dequeue. Some candidates did not output the value returned from the function call.

COMPUTER SCIENCE

Paper 9618/42
Practical

Key messages

The only document that should be submitted for each candidate is the evidence document. No other files, including source files or program code, should be submitted. This document must not be in a zip folder. Candidates need to include their name and candidate number in the header of the evidence document.

Candidates need to copy their program code into each required box in the evidence document, this should be copied and **not** a screenshot. This is because the colours used by IDEs are not always clearly visible on the documents, especially if there is a black background. Screenshots that are too small are also illegible. If the program code cannot be read due to the screenshot being too small, or inappropriate colours used, then marks will not be awarded. Screenshots of the outputs of testing should be either white text on a black background or black text on a white background.

When copying code candidates need to make sure it is all visible on screen and that the boxes are not expanded beyond the width of the page; any code that is not visible on the document cannot be awarded marks.

Candidates also need to make sure all of their code is included and not cut off, especially the start of subroutines and that indentation is appropriate when using Python.

General comments

Candidates showed a good understanding of Object-Oriented Programming (OOP), often accurately creating a class, constructor and the get and set methods.

Candidates do need to make sure they are meeting each requirement of the question, for example using correctly parameters if instructed, or storing an outputting a return value if that is given as a task in the question.

More candidates were attempting later question parts even if an earlier question part was not done accurately, for example in **Question 1** they were calling `Play()` even if they had not written the subroutine, this allowed them to gain marks for these tasks.

When producing a screenshot for testing candidates need to make sure all data is visible, that includes the data that they have input. Without this input data it is not possible to check whether the candidate has used these inputs to create the given output.

Comments on specific questions

Question 1

This question required candidates to read string data from a text file and manipulate the data using user inputs.

(a) Many candidates were able to create the procedure header and take a parameter. Candidates were often able to open the correct file using the parameter and then read in the data. The stronger responses used an appropriate loop that repeated until the end of the file, stripping the carriage return from the line when required by their programming language. Some candidates made appropriate use of exception handling to catch if the file did not exist.

Fewer candidates correctly totalled the number of answers, this should exclude the main word so was the number of lines read in minus 1. Candidates who did this correctly used a variety of methods such as starting the counter at –1.

Some of the weaker responses did not use the parameter appropriately, instead reading the filename as input or attempting to open a file based on different possible values of the parameter.

(b) Candidates were often able to accurately output a suitable message and read an input from the user. Some output messages that were not appropriate for the scenario included 'Enter data', or 'Filename:' without asking them to enter the three options that were given in the question.

Fewer candidates were able to convert the input into the appropriate filename. Candidates who did this successfully often concatenated '.txt' to the input or used a selection statement. Some candidates output the final filename and did not call `ReadWords()` with it as a parameter.

(c) (i) This question set-up the game so that users could guess the words until they wanted to stop. The stronger responses used an appropriate conditional loop that iterated until 'no' was entered, some of the weaker responses used a count-controlled loop that allowed one guess for each word that was stored.

Some candidates recognised that the first word in the array was not one of the answers and correctly excluded this from the search, for example by looping from the second element. Candidates were often able to remove the word when correctly guessed, most commonly by replacing it with a null value. Some candidates did not use an appropriate null value, for example 'null' which could have been one of the answers and could produce an inaccurate result.

Some candidates correctly output if the answer was correct but did not correctly output when the guess was incorrect. This was due to including the output within a for loop, so every time a comparison was made in the array an output would state that it was incorrect. This would result in many incorrect outputs for each guess.

(ii) Some candidates were able to correctly calculate the percentage of answers correct, at times this would produce an incorrect result due to their addition of the total number of answers being incorrect from a previous part.

When outputting which answers were not entered many candidates were able to perform the correct comparison and output for a null value based on what they did in part **(c)(i)**. Fewer candidates recognised that the first element in the array should not be included and correctly missed this in the outputs.

(d) (i) Many candidates were able to correctly call `Play()`.

(ii)(iii) Candidates were often able to correctly enter the words as input, but fewer had the correct outputs commonly due to inaccurate percentage calculations and incorrectly outputting the main word as one that the user missed.

Question 2

This question required candidates to use Object-Oriented Programming (OOP) to create a binary tree.

(a) (i) Many candidates were able to accurately create a class and a constructor for their class. Fewer candidates included the appropriate single parameter as required in the class design and then assign this to the attribute in the constructor. Some candidates took additional parameters and assigned these to `LeftPointer` and `RightPointer` instead of assigning them –1 as required in the class design.

(ii) Candidates were often able to create appropriate get methods for the three attributes. Some candidates attempted to include a parameter in their methods or read a value as input and then return this value instead of the attribute.

(iii) Many candidates were able to accurately create the set methods as required. Some candidates did not include a parameter in their methods, instead attempting to read a value in from the user and assign this within each method. Some candidates returned or output values in the method instead of assigning to the attributes.

(b) (i) `TreeClass` used containment to store elements of the type class `Node`. Some candidates attempted to use inheritance for `TreeClass` instead, defining it to inherit from the class `Node`, this was then repeated in the constructor where the super or parent class constructor was incorrectly called for `Node`.

Some candidates were able to correctly recognised that there were no parameters to the constructor. All attributes were initialised to default values by the constructor. The array needed to be initialised to 20 objects of type `Node` with a data value of –1, this required an object to be instantiated and stored in each element. Some candidates initialised the array as an empty array, or as an array of integers with each element storing –1 instead of a `Node` with the data value of –1.

(ii) This question provided candidates with a structured series of steps on how to create the method `InsertNode()` for the binary tree. Many candidates were able to correctly follow the first few steps, but many did not complete the algorithm.

A common error was in identifying whether a tree was empty. Some candidates attempted to compare the array `Tree` with an empty array, but the class design initialised the array with 20 elements therefore it would not be null.

The design stated that the method took a `Node` object as a parameter, some candidates incorrectly used treated the parameter as a data value and then attempted to create a new `Node` object using that data.

(iii) `OutputTree()` is a method that is within the class `TreeClass`. Some candidates wrote a procedure for `OutputTree()` and therefore were unable to access the data as required.

Some candidates that created a method were able to iterate through each element in the array `Tree` and output each of the pointers and data of each `Node` object appropriately.

Some candidates attempted to traverse the tree, for example writing an in-order traversal, which meant that only the data values were being output and not the pointers.

(c) (i) Many candidates were able to accurately instantiate an object of type `TreeClass`.

(ii) This question required candidates to assign a series of nodes to the tree. Candidates needed to identify that a `Node` object was needed for each value and then the method `InsertNode()` needed to be called for the tree.

A common error was sending the data as a parameter and not creating a `Node` object. Some candidates called the method `InsertNode()` without reference to the object that it was for, for example calling it as a procedure. Similarly, some candidates called `OutputTree()` as a procedure instead of a method.

(iii) Few candidates were able to produce an accurate output for the tree. Some responses had the integer values output in order without the pointers, some candidates who did output pointers did not have the correct ones.

Question 3

This question required candidates to manipulate an array of integer data using a recursive and iterative sorting method and then write a searching algorithm.

(a) Many candidates were able to correctly create an array with the integers given. Some candidates created an array but did not assign the values provided.

(b) (i) This question provided a pseudocode algorithm for a recursive insertion sort method. Candidates were required to read the pseudocode and convert it into their language. Many candidates were able to accurately convert some aspects of the algorithm. Some responses did not include all of the code, for example missing a selection statement, or not using the appropriate parameters.

(ii) Candidates were often able to call the function correctly with the two correct parameters. Some candidates hard-coded the number of elements whilst others used a suitable function to find the length of the array.

Fewer candidates recognised that the function returned an array and that this array needed to be stored and then used in the output as required by the third point in the question ‘output the content of the returned array’.

(iii) Candidates were often able to produce the correct output. Some candidates who had not sent the correct number of elements as a parameter in part **(ii)** did not have a fully sorted array.

(c) (i) This question required candidates to rewrite the function from part **(b)(i)** so that it no longer used recursion, instead using iteration.

Some candidates wrote an insertion sort as an iterative algorithm and did not use the algorithm provided, which still gave them the correct result. Some candidates did not remove the recursive call or rewrote the algorithm and removed the line with the recursive call leaving the remainder of the algorithm as it was originally.

(ii) Some candidates were able to call their iterative function appropriately and output the content of the returned array. Some candidates did not output the content of the returned array because they did not store this from the function call.

(iii) Candidates often gained the correct output for the sorted array.

(d) (i) This question required candidates to write a recursive binary search function. Candidates were told which parameters were provided and that the function needed to return –1 if not found, or the index if the data was found.

Many candidates were able to produce a suitable recursive binary search algorithm. Some candidates include iteration within their recursive function which meant that the calls would repeat indefinitely, whilst others had suitable return statements preceding each recursive call.

Fewer candidates were able to correctly return –1 only when the data was not found. Some candidates had inappropriate checks for not found, for example not considering when two pointers switch positions, or they were missing return statements earlier on so that when the recursive calls unwind –1 was always returned.

(ii) Some candidates were able to call their recursive function with the correct parameters, some candidates did not use the correct value for the last index instead using the length of the array. Some candidates clearly showed how they used a sorted array, for example using the returned array from a previous function call or rewriting the array with the data in order.

The question required the return value to be stored and then a message output depending on the content. Some candidates only output the return value or output a message that did not include the index if this was the return value.

(iii) Candidates were often able to produce the correct output of 6.

COMPUTER SCIENCE

Paper 9618/43
Practical

Key messages

The only document that should be submitted for each candidate is the evidence document. No other files, including source files or program code, should be submitted. This document must not be in a zip folder. Candidates need to include their name and candidate number in the header of the evidence document.

Candidates need to copy their program code into each required box in the evidence document, this should be copied and **not** a screenshot. This is because the colours used by IDEs are not always clearly visible on the documents, especially if there is a black background. Screenshots that are too small are also illegible. If the program code cannot be read due to the screenshot being too small, or inappropriate colours used, then marks will not be awarded. Screenshots of the outputs of testing should be either white text on a black background or black text on a white background.

When copying code candidates need to make sure it is all visible on screen and that the boxes are not expanded beyond the width of the page; any code that is not visible on the document cannot be awarded marks.

Candidates also need to make sure all of their code is included and not cut off, especially the start of subroutines and that indentation is appropriate when using Python.

General comments

Candidates showed a good understanding of Object-Oriented Programming (OOP), often accurately creating a class, constructor and the get and set methods.

Candidates do need to make sure they are meeting each requirement of the question, for example using correctly parameters if instructed, or storing an outputting a return value if that is given as a task in the question.

More candidates were attempting later question parts even if an earlier question part was not done accurately.

When producing a screenshot for testing candidates need to make sure all data is visible, that includes the data that they have input. Without this input data it is not possible to check whether the candidate has used these inputs to create the given output.

Comments on specific questions

Question 1

This question required candidates to input data into an array and then manipulate this data using standard sorting and searching methods.

(a) Many candidates were able to accurate declare the 1D array with the appropriate space and data type identified. Some candidates provided a comment that showed their intention to create an array but did not actually create an array.

(b) Candidates were often able to accurately output a message to prompt for the input and then read in this input. Some candidates were then able to use this input value within a loop to iterate the number input quantity of times, reading in an integer each time and storing it in the array.

The stronger responses used suitable validation for the first number input, often by looping until this input was valid. Some candidates inaccurately validated the numbers that were input within the loop, attempting to restrict these integers to be between 1 and 20 (inclusive).

(c) (i) Many candidates were able to call `Initialise()` accurately. There were a range of methods of outputting the contents of the array, in Python candidates often output the array direct, whilst other languages required a loop through each element and output of each element.

(ii) This question required candidates to enter the data in the order given. Part **(b)** required validation, so the first input of 30 needed to be rejected and then the number 5 being entered to indicate that 5 numbers were to be input. Some candidates incorrectly included an additional input of 7 and then input all of the numbers as data to store in the array.

(d) (i) Candidates were often able to produce a bubble sort algorithm. There were a range of approaches including candidates who used two count-controlled loops and some who create more efficient algorithms that stopped as soon as the array was ordered.

(ii) Many candidates were able to call the bubble sort procedure and then output the contents of the array again.

(iii) Candidates were often able to produce the correct result showing the values input and the output of the sorted array.

(e) (i) This question required candidates to write an iterative binary search algorithm. Some candidates did not meet these requirements due to them writing a recursive binary search algorithm.

Candidates who did attempt the iterative algorithm were often able to meet several of the requirements, for example the header and calculating the middle value. These candidates also often correctly identified when the value was found and returned this. Fewer candidates correctly updated the first and last pointers accurately or overwrote a found index to return –1 even when the data was found.

Some candidates inaccurately compared the middle index to the data to find, instead of comparing the element in the middle index of the array.

(ii) Candidates were often able to take the number to search for as an input and use this in the function call. Fewer candidates correctly output the return value from the function call. Some candidates chose to output a different message depending on the return value which was not required but still output the index correctly when found.

(iii) This question required evidence of two tests from the user, each tested needed to show the inputs and the output. Many candidates provided the correct outputs, but fewer candidates also included the inputs in their results.

Question 2

This question required candidates to write a program using Object-Oriented Programming (OOP) to store and access data.

(a) (i) Many candidates were able to accurately create the class `Tree` and declared the attributes with the appropriate data types. Candidates were also often able to create the constructor with the appropriate parameters that they then assigned to the parameters.

Some candidates attempted to initialise the values in the constructor to default values such as 0 instead of using the parameters.

(ii) Candidates were often able to accurately declare the get methods. Some common errors including passing a parameter to the functions or attempting to assign or overwrite the attribute in the get method.

(b) This question required candidates to read data in from an external text file. Candidates were often able to open the file accurately and read in the data. Fewer candidates made appropriate use of exception handling for if the file was not found; for example, by including the file opening within the try and then including closing the file external of the exception handling which would not be possible if the file had never been opened.

Many candidates accurately split the data that was read in the file and instantiated a new object to be stored in the array.

Few candidates closed the file that they opened.

(c) This procedure needed to take an object as a parameter to then be output. Some candidates took a parameter but did not use it as an object, for example using it as an array and then attempting to access elements instead of making use of the get methods to access the data.

The stronger responses accurately used the get methods to check the value of evergreen and used this to determine which message to output. Some candidates output a standard message and then checked this for the final statements, whilst others repeated the statement.

(d) (i) Many candidates were able to accurately call `ReadData()` but fewer candidates stored the value returned from this function and use it to call `PrintTrees()`. Some candidates did not include a parameter when calling `PrintTrees()`.

(ii) Many candidates were able to get the correct output for this question.

(e) (i) This question required candidates to take input from the user and compare it to find trees that had data meeting the requirements.

Many candidates were able to take the required inputs from the user. Candidates were often able to iterate through each element in the array, but fewer were able to access the attributes for each value. The stronger responses made appropriate use of the get methods to perform the comparisons on each tree object in turn.

Candidate were often able to create an array and assign the `Tree` objects that met the requirements to this array. Some candidate then sent each object to `PrintTrees()` when identified, whilst some candidates iterated through the new array after all objects were found.

Some candidates took the input of requirements within the loop and therefore required new requirements to be input each time an object was compared.

(ii) This question required the editing of the procedure previously created. Candidates were often able to take the required data as input. Fewer candidates were able to accurately calculate the number of years it would take for that tree to grow to its maximum height. Some of the weaker responses use a `Tree` object but without showing how this object was accessed i.e. how the correct tree that the user input was found in the array of acceptable trees. Validation was not a requirement for this question, but some candidates still implemented this to ensure the algorithm worked correctly.

Some candidates chose to round the number of years up, some candidates rounded it down, some candidates calculated the number of years and months whilst some output the result of the calculation, all of which were acceptable.

(iii) Candidates were often able to generate the correct outputs for the system. Some candidates did not include the input of the tree requirements and the tree selection in their screenshot and therefore it could not be proven that the correct data was input.

Question 3

This question required candidates to create a queue data structure with its appropriate enqueue and dequeue functions.

(a) Many candidates were able to create an array and initialise the head pointer and tail pointer. Fewer candidates were able to initialise all 20 elements of the array to a suitable null value.

(b) Candidates were often able to create the `Enqueue()` function and store the data in the correct position. Some candidates did not increment the tail pointer in the appropriate place and overwrote the last item because `QueueTail` points to the index of the last item and not the next empty space.

Some candidates attempted to check if each value in the array had something other than null stored to check if the array was empty, whilst some candidates attempted to compare the array to an empty array which would not work due to the required initialisation of each of the 20 elements in part **(a)**.

Some candidates recognised the need to change the head pointer if it was currently pointing to –1 to become 0 and including this check and change in their solution.

(c) Few candidates were able to identify how to check if the queue was empty, with many only checking the initial empty queue. This did not consider if elements had been enqueued and dequeued previously which would mean that the head pointer was no longer storing –1.

(d)(i) Candidates were provided with a check digit calculation and an example that they had to implement for data input from the user. Candidates were often able to do this calculation accurately with a number of different methods used to access each alternate value in the input. The stronger responses also checked whether the result was 10 and made the comparison to x instead of a number.

Some candidates did not take 10 numbers from the user, instead taking 10 characters as input and attempting to use these in the calculation. Some candidates only took one input from the user.

Only the inputs that were valid based on the check digit were to be stored in the array and they were to be stored without the check digit, some candidates stored the full number input by the user and did not remove the check digit.

Fewer candidates correctly counted and output the number of invalid data items that were input by the user.

(ii) Candidates were often able to accurately call `StoreItems()` and `Dequeue()`. Fewer candidates stored the return value from `Dequeue()` and then output an appropriate message depending on the value, often by not returning the returned value instead stating that a value was removed.

(iii) Few candidates gained the correct output, often due to not outputting the number of invalid items or by outputting a different value as the result of dequeue. Some candidates did not output the value returned from the function call.