COMPUTER SCIENCE

Paper 9618/11
Theory Fundamentals

Key messages

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

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

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

General comments

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

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

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

Questions involving calculations or a clear methodology were more likely to be completed successfully, questions requiring application of knowledge and the writing of extended prose were more challenging.

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 very well. Many candidates were able to correctly write the two logic expressions. A few candidates confused the symbols for the NOR and XOR gates.

(b) This question was answered well, although some candidates applied the NOT operation incorrectly, either before the XOR or after the final OR.

Question 2

(a) Many candidates were able to explain the need for a buffer but then found it more challenging to expand their explanation sufficiently to achieve all the marks. Some candidates should improve their understanding of what is meant by real-time bit streaming.

(b) (i) Many of the answers to this question were completely generic and overlooked the fact that the conference was a live event. Candidates are expected to be able to apply their knowledge to different situations rather than repeat generalised information.

(ii) This question also required application of knowledge. Many candidates chose lossy compression but then went on to simply describe the compression method without any application to the given scenario.

(c) Candidates were able to provide correct answers for the use of modems; stating the use of dedicated lines proved to be more challenging. Vague statements that repeat the words used in the question could not be awarded marks. Sentences such as, '*Dedicated lines are dedicated connections.*' were frequently seen. This does not provide any information about their use.

Question 3

(a) The first and last answers were often correct. The middle answer was most likely to be incorrect. 128 and 255 were popular incorrect answers.

(b) (i) This question was answered well. Many candidates were able to give two advantages of the Unicode character set. Some candidates should read the question carefully as they stated features of the Unicode character set without suggesting why they might be advantageous.

(ii) There were many correct answers for this conversion. Some candidates who had the correct individual denary values need to take more care with the addition.

(iii) There were fewer correct answers for this conversion. Some candidates found the conversion from hexadecimal to denary quite challenging, and others left the answer in binary which was not what was asked for.

Question 4

(a) Many responses offered explanations of the automatic barriers or the billing system, which were not relevant to the question. Many of the responses re-worded parts of the question without referring to any AI. There were vague references to matching images to a database, but few descriptions of how the customers would be identified.

(b) The candidates' choice of sensor was generally appropriate. Stating how the sensor would be used proved to be more challenging. Some candidates should understand that repeating information given in the question, such as '*to sense that an item has been taken off the shelf*', could not be awarded marks.

Question 5

(a) There were some good, complete answers to this question. Some candidates need to ensure that they use a recognised convention for showing the relationships.

(b) This question was generally answered correctly. Some candidates should improve their understanding of what is meant by a candidate key.

(c) This question was answered very well. A few candidates gave the primary keys instead of the foreign keys.

(d) Candidates found this question quite challenging. Many of the responses simply described the two tables without explaining why they cannot be combined. Vague statements about not having two primary keys in one table were seen regularly, but there was very little explanation about the implementation of a one-to-many relationship and the avoidance of repeated attributes.

(e) There were some excellent attempts at this SQL script. Some candidates should make sure that they read the question carefully and remember that information given in the question stem applies to all the parts of the question. Candidates were told in the stem of the question that 'the attribute 'Complete in the table ORDER stores the Boolean value TRUE…', yet many otherwise correct conditional clauses used a string comparison rather than a Boolean one. The question also asked for an appropriate identifier to be given for the total cost, which was sometimes missing. Some candidates need to ensure that they understand the difference between the use of the SUM and COUNT functions.

Question 6

(a) (i) There were some good, complete answers to this question. Some candidates should improve their understanding of the principal operation of a hard disc. There was some confusion with an optical disc.

(ii) Many candidates found it challenging to explain how the different amounts of RAM would affect the system performance. Quite a few were able to state that more applications could be run at the same time but other factors affecting performance were rarely mentioned. This question required candidates to apply their knowledge, and so generic descriptions of RAM were not awarded marks.

(iii) Generally, candidates were able to state that a wider bus meant that more information could be transferred. Some candidates should understand that the increase is in the amount that is transferred *at the same time*. Also, in this type of question, two answers that are the opposite of each other will not both be given credit. For example, answers such as, '*a wider data bus means that more data can be transferred at the same time, whereas a smaller data bus means that less data can be transferred at the same time*', will only be credited once.

(b) Candidates were expected to use appropriate technical terminology and demonstrate more than just general knowledge in this question. Candidates needed to give a description alongside their list of management tasks performed by the Operating System to be awarded credit.

Question 7

(a) This question was answered well, with most candidates choosing the method of encryption. Some candidates should understand that encryption does not prevent an unauthorised person accessing the data, it prevents the data from being understood if it is accessed.

(b) This question was quite challenging for many candidates. There were a lot of vague responses about comparing a signature before transmission with one after transmission. Many candidates should improve their understanding of digital signatures.

(c) There were very few correct answers to this question. Many candidates should improve their understanding of what is meant by a checksum. There was a lot of confusion with check digits and parity. Vague answers described a checksum as checking the sum of something unspecified.

Question 8

(a) There were some excellent complete answers to this question. The most frequent mistake was the incorrect completion of the LDI instruction at address 203. Some candidates with otherwise correct trace tables stored the final value in address 80 instead of address 81.

(b) (i) This question was answered well. A few candidates confused the least significant (rightmost) bit with the most significant (leftmost) bit, and some candidates should take care to include the appropriate symbol to indicate the type of the operand.

(ii) This question was also answered well. Some candidates should improve their understanding of the XOR operation.

(iii) This question was also answered well. The most common error was the performance of a logical shift left instead of a logical shift right.

COMPUTER SCIENCE

Paper 9618/12
Theory Fundamentals

Key messages

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

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

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

General comments

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

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

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

Questions involving calculations or a clear methodology were more likely to be completed successfully, questions requiring application of knowledge and the writing of extended prose were more challenging.

Comments on specific questions

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

Question 1

(a) Many candidates identified that statement number 1 and statement number 3 were incorrect, although some of these candidates were then unable to correct them. A popular incorrect change for statement 1 was that the Program Counter counted the number of statements that had been executed. Corrections to statement 3 were better. Identifying that statement number 4 was incorrect was more challenging. Statement 5 was a popular wrong choice. Frequently when candidates had identified that statement 4 was incorrect, they were unable to correct it. Candidates should understand that registers hold or store data, they do not transfer it.

(b) There were many correct answers to this question; the Current Instruction Register (CIR) was a popular correct choice. Some candidates need to read the question carefully as one of the registers given in the question written as the answer.

(c) There were some excellent responses to this question. Some candidates should ensure that they look at the number of marks available for the question and that they make an equivalent number of distinct points in their answers. Statements about priority checking were often included, but there seemed to be some misunderstanding with only the relative priorities of the interrupts being checked without any reference to the priority of the process being interrupted.

Question 2

(a) This question was answered very well. The most common mistake was omitting the leading zeros. If the question asks for 12-bits, the leading zeros are required.

(b) (i) This question was also answered well. Some candidates who found the two's complement of the given binary integer and then converted that to denary, unfortunately forgot the minus sign.

(ii) Overall, this question was not answered well. Many of the responses gave denary values, despite the question asking for the two's complement binary integers.

(c) Many candidates were able to give an application for the use of Binary Coded Decimal (BCD). Justifying the use of the BCD proved to be more challenging for candidates.

Question 3

(a) This question required the application of knowledge and candidates generally found the question very challenging. Responses frequently just re-worded the information given in the question. Very few candidates realised that there would be at least three stages in this operation.

(b) This question was answered better. Many candidates were able to state at least one social benefit of the program. Overcoming the language barrier and helping the visually impaired were popular choices. There were a small number of vague answers about saving time, but with no justification of how any time was saved.

(c) (i) This question needed to be read carefully. Describing the benefits of distributing the program under this type of licence is not answering the question. Many candidates were able to describe two features, usually that the program is sold for a fee and that the source code is not provided. There was some confusion with Shareware, and some candidates need to take care when discussing copyright issues. Copyrighting the program will not prevent it being edited or re-distributed, it just makes it illegal to do so.

(ii) This part question too needed to be read carefully. It is not enough to describe the features of an open source licence, candidates need to say why the feature makes it unsuitable for the program described earlier in the question.

Question 4

(a) This question was answered very well. A few candidates should improve their understanding of the difference between the symbols for a NOR gate and a NAND gate.

(b) This question was also answered very well. Some candidates should improve their understanding of the XOR operation.

Question 5

(a) Many candidates found this question challenging. Responses tended to describe the content of the CHARACTER_ITEM table without giving its purpose. There was very little mention of the many-to-many relationship that exists between the CHARACTER and ITEM tables. Candidates are told in the question that the database is normalised, so answers such as, '*its purpose is to normalise the database*' could not be awarded marks.

(b) There were some excellent, completely correct answers to this question. Some candidates should realise that the primary key is not always the first one listed in the table attributes.

(c) (i) This is a question that requires candidates to apply their knowledge, so describing the features of the chosen method is not enough to be awarded full marks. Answers need to explain how the chosen method protects the data. The question asks for methods to prevent unauthorised access and so responses such as '*it prevents unauthorised access*' is a repeat of the question. Passwords and encryption were two popular correct methods given. Some candidates should understand that giving two equivalent methods, such as a password and a PIN, will not both be given credit. Some candidates should understand the difference between access rights and views.

(ii) This question did not require any application and was answered better. Popular correct choices were validation (or verification) and enforcing referential integrity.

(d) (i) Many candidates were able to give a correct condition in their SQL scripts. Some candidates should understand the difference between the COUNT function and the SUM function, and some candidates should improve their understanding of how to use multiple tables in a single SQL script. The mark for joining the tables was the one least often awarded.

(ii) The condition clause was often correct in this question, although missing quotation marks were regularly seen. Many candidates found the statements needed to update the values challenging. Some candidates would benefit from more practice in setting up simple databases and using the various SQL statements in section 8.3 of the syllabus to query and maintain these databases.

Question 6

(a) This question was answered well. Area of coverage and ownership of the transmission media were the two most popular correct choices. There was a misconception among some candidates that a WAN was always wireless, and a LAN was always wired. These candidates need to improve their understanding of this topic.

(b) Many candidates found this question challenging. Few candidates mentioned anything about cells, towers or line of sight. Candidates should improve their understanding of the operation of the cell phone network.

(c) This question was answered well. The last two blank spaces were the ones most likely to be incorrect.

(d) The exact number of devices was given in the question. There was also an instruction to label all devices. Many candidates were not awarded marks, not because they did not know how to draw a star topology, but because they either omitted the labels on the devices or left out one or more devices altogether. Candidates were also expected to identify the device providing internet access, frequently this was not done.

(e) There were a few excellent, complete answers to this question. Some candidates should improve their understanding of the operation of a switch in a network. A common incorrect answer was, '*to* *turn the network on and off*'. Some candidates should ensure that when writing about forwarding packets they make it clear that the switch sends only to the intended recipient.

Question 7

(a) (i) This question was answered well. Most candidates were able to correctly trace the given program.

(ii) Some candidates found this question challenging and described the two commands by repeating the wording given in the table for the previous question. The question was asking for the effect of changing the instruction, which is what would happen when the changed program was executed.

(iii) Many candidates were able to correctly identify and describe another mode of addressing. Indirect and indexed were the most popular choices. Some candidates need to read the question carefully as modes of addressing given in the preceding table were explicitly excluded.

(b) (i) This question was answered very well.

(ii) This question was answered well. Some candidates should improve their understanding of the XOR operation.

COMPUTER SCIENCE

Paper 9618/13
Theory Fundamentals

Key messages

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

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

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

General comments

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

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

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

Questions involving calculations or a clear methodology were more likely to be completed successfully, questions requiring application of knowledge and the writing of extended prose were more challenging.

Comments on specific questions

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

Question 1

(a) (i) This question was generally answered well. Many candidates understood how to calculate the file size of the image. A common incorrect answer, however, was 32 MB, where candidates had omitted the division by 8 to convert bits to bytes.

(ii) Correct answers about the image not matching the original and a smaller file size were seen frequently. Some candidates need to look carefully at the question. The command word is ‘*explain*’ and so answers are expected to contain more than just a statement of fact. An indication of *how* or *why* the image or the file size is changed would be expected. The question states that the bit depth is decreased, so an answer which says, ‘*the file size is decreased*’ simply repeats the question wording and is insufficient.

(b) This question was answered well. There was some confusion between sampling resolution and image resolution. Some candidates should be aware that in questions like this, only the first answer in each space will be considered.

(c) (i) This question was also answered well. Some candidates need to take care when the question asks just for the characteristics of Unicode that they do not give comparisons with ASCII.

(ii) There were many completely correct answers to this question. The hexadecimal value was the one most likely to be incorrect.

Question 2

(a) (i) Many candidates found this question challenging. Many candidates did not understand the role of the system clock within the CPU and how the timing signals were used by the control unit for synchronisation. There was also confusion about which bus was used for the transmission of the signals. The data bus was a popular incorrect choice.

(ii) Several candidates did not read the question properly and wrote extended prose for the answer instead of the register transfer notation that was required. Some candidates who did write the transfer notation could improve their understanding of the meaning of the brackets in the statements.

(b) Many candidates understood that using cache memory improved the system performance but then were unable to expand their answer. Vague statements about reducing time to process applications were not enough to be awarded a mark at this level. Candidates should be aware of the need to use the correct technical terminology appropriately. Incorrect references to secondary memory instead of RAM and statements about cache being non-volatile were two examples of the incorrect use of terms.

(c) There were many answers stating that HDMI transfers both audio and video over a single cable, but then any further description was missing. Many candidates need to improve their understanding of HDMI connections.

Question 3

(a) This question was answered well. There were several possible acceptable alternatives in some of the spaces, although some candidates need to understand that at this level writing vague words, for example, ‘*quickly*’ for the final answer is not enough.

(b) The methods of data protection were often correct, although some candidates should understand that alternatives to password which mean the same, such as a PIN, are not accepted. Completing the second column of the table was more challenging as candidates often described the method rather than explaining *how* it protects data.

(c) There were some good answers to this question. Some candidates could improve their understanding of the role of the web browser. It was sometimes incorrectly identified as the server in this application rather than as an application on the client computer. Some responses gave comparison with peer-to-peer networks which was not needed.

(d) There were some excellent, complete answers to this question. A small number of candidates wrote that not belonging to a professional ethical body would automatically mean that the code produced by the programmer would be malicious or full of errors.

Question 4

(a) This question was answered very well. There was some small confusion between the XOR gate symbol and the NOR gate symbol.

(b) This question was answered well. The most likely cause of error was the incorrect application of the XOR operation.

Question 5

(a) (i) There were many completely correct trace tables. The most frequent incorrect answer occurred when after the comparison at address 520 the jump was made to address 505 rather than back to address 501.

(ii) Many candidates were able to correctly identify two modes of addressing. Some candidates need to read the question carefully as they gave addressing modes used in the previous part question which were specifically excluded. Describing the chosen addressing modes was more challenging. Candidates who correctly used the term '*operand*' in their answers were often able to give a better description than candidates who attempted a description with alternative names.

(b) Many candidates needed to read this question carefully. The question asked for instructions (plural) and candidates were provided with 2 answer lines, but many responses only gave one instruction. A few candidates confused the most significant (leftmost) bit with the least significant (rightmost) bit.

Question 6

(a) The one-to-many relationships between CUSTOMER and ORDER and between SUPPLIER and PRODUCT were usually correctly identified. Some candidates then found the standard breakdown of the many-to-many relationship between ORDER and PRODUCT into two one-to-many relationships with a linking table more challenging. Candidates should be advised to use a recognised convention for showing the relationships.

(b) There were some good attempts at this question. Some otherwise correct missed the 'S' on the word VALUES, meaning the mark could not be awarded. Some candidates need to take more care with the data types of the attributes.

(c) There were some good attempts at this SQL script. Some candidates need to read the question carefully. Many otherwise correct clauses used a string comparison rather than a Boolean one and an otherwise correct COUNT clause omitted the name. Some candidates need to improve their understanding of the difference between a SUM function and a COUNT function.

(d) (i) Many candidates found this question challenging. There were four items given in the question that should have been excluded from answers, however, frequently alternative forms of these items were included, such as field names. This is an area where many candidates should improve their understanding.

(ii) This question was challenging for many candidates and is an area where understanding should be improved. Responses frequently described the general features of an Integrated Development Environment rather than describing *how* a developer could use a developer interface provided by a DBMS.

Question 7

(a) (i) This question was generally answered well. Some candidates confused the logical shift with a cyclic shift, and some candidates shifted the value right instead of left.

(ii) Candidates generally found this question more challenging. A common mistake was the correct preservation of the sign bit but then omitting to copy the sign bit as the shift was accomplished.

(b) The binary addition was correctly completed by most candidates. Some candidates should ensure that they clearly identify any overflow bits and that their working can be clearly seen.

(c) The binary subtraction proved to be more challenging. Several candidates incorrectly subtracted 0110 0100 from 0001 1110 instead of the other way around. Candidates who opted for finding the two's complement of 0001 1110 to add to 0110 0100 need to ensure that any overflow bit generated by the addition is removed from their final answer.

COMPUTER SCIENCE

Paper 9618/21
Fundamental Problem-Solving and Programming Skills

Key messages

A detailed knowledge of the CIE pseudocode is essential for success on this paper. Programming is a skill and so like most it demands practice. There are past examination papers for this component which should guide centres towards the general standard expected; similarly, the use of the CIE approved textbooks.

Many candidates struggled with basic building blocks of program design and construction. See the later comments about the choice of variable names for **Question 1(a)**.

Detail is important once we reach the coding stage of solving the problem. Often marks were lost due to errors in the pseudocode. Many of these are described in the General Comments section.

See the comments which follow for **Question 6(a)**. There are some clear statements in the rubric of the question which gave clear guidance as to what would/would not be suitable test data.

See the later comments for **Question 2(b)(i)**. The rubric was *'State a value ….'*. Answers frequently gave the answer as `BonusPay` or `ValueOfSales` which is not a 'value'.

General comments

There were a number of scripts seen, where there was little or no attempt at any of the pseudocode questions on the papers.

A number of candidates appear to struggle with the basic building blocks of problem design. The temptation of the novice programmer is often to immediately attempt to write the program code but, for all but the most trivial of problems, this approach should be discouraged.

Key questions such as:

- does the design merit a division into two or more modules?
- will they be implemented as a procedure or function?
- does the problem require a loop?
- if so, which loop structure will be the most appropriate - count-controlled or conditional?
- if selection is required, is the use of one or more IF structure(s) or a CASE structure best?

All of this is to be considered alongside the decisions about what variables are to be used – are some best implemented as a constant?

All of these decisions could be considered before candidates write their first line of program code.

<u>Comments on specific questions</u>

Question 1

(a) While most candidates successfully identified the appropriate data type for the four data items, the choice of each variable name, at times was not considered appropriate was often lacking.
Row 1 – `Category` although others were acceptable.
Row 2 – The identifier must convey that this is the date on which the item was sold; so – `DateSold`.
Row 3 – the cost of the item – `ItemCost`. A very popular answer was Price which was considered an acceptable answer.
Row 4 – `ItemInStock` or just `InStock` conveying the idea of only two possible values.

Weaker answers often did not appreciate the concept of an identifier name; either filling the column with more similar data items or repeated the same or a different data type.

It is recognised that there is a trade-off to be made between choosing names which are meaningful against the desire not to use names which are excessively long.

(b) All candidates were able to score here, and the question proved a good discriminator. Scoring the full available five marks however was rarely seen. There was no apparent pattern which suggested a specific lack of understanding.

(c) The question gave the stock control scenario as a starting point but only a few answers referred to it. Candidates often had a vague idea that the term 'decomposition' involved breaking 'something' down into smaller parts but rarely conveyed the understanding that it was the problem or the required system which needed to be considered. Answers which described the need for modules to monitor stock level, record new stock arrival, record stock sales (or similar) were rarely seen.

Question 2

(a) (i) A question which required a good explanation of the problem, and this was usually lacking. It was often suspected that the candidate knew what the problem was but were unable to express this. A simple statement such as *'the two decision boxes are in the wrong order'* would have been sufficient.

(ii) Often the answer given for part **(a)(i)** did not gain credit, but the candidate was able to score this mark for part **(ii)**. Again, a simple statement such as *'The two decisions boxed should be swapped'* was sufficient.

(b) (i) Not well understood. Candidates often stated that it was variables names in the algorithm (`ValueOfSales` or `BonusPay`) which should be replaced by a constant which is meaningless. Candidates who stated that each of the critical bonus pay thresholds (0, 10 and 100) and/or the value of sales figures (2000 and 4000) could each be replaced by a constant demonstrating their understanding of this design issue. Answers often gave just one of these critical values and gained credit.

(ii) Two marks available and the first mark gained was usually for stating that the value of a constant cannot be changed. The second mark was for demonstrating an understanding that there is a consequence of this. For example, it avoids repeatedly writing the assigned value throughout the program code. Alternatively, that a change to the value requires that it is changed in one statement only (that is where the constant value was defined). This second mark proved elusive to candidates.

(iii) Most candidates were able to secure the mark stating that the code would be 'tried and tested'.

Few answers suggested that the library routines would provide code which the programming might themselves find difficult to code.

(c) Answers expected were syntax and run-time errors. A syntax error was usually well explained – that the written syntax of a statement does not follow the grammar and languages rules. Some candidates gave a convincing practical example for this which secured the mark.

Run-time errors were less well understood. A popular answer on its own was to state *'divide by zero'* which was considered insufficient. This answer needed to be followed by the explanation that this would cause an illegal operation to be performed or would cause the program to crash. It was the consequence of the error which was often missing in the explanation given.

Question 3

The first pseudocode question on the paper and a very varied level of response. A mark was awarded for the input of the user's guess, and this was often lost due to omission of some prompt text. There were three marks available for the correct expression to calculate each random number; use of the `RAND()` function followed by use of the `INT()` function and then the addition of 1 to generate the number in the correct range. Candidates often failed to secure the third mark.

There were frequent common errors seen usually which resulted in a loss of marks. These included:

- an incorrectly formed conditional loop; often the reason being it was not clear where the loop ended
- omitting to declare all the variables used
- an `IF` statement with the `ENDIF` omitted.

Question 4

(a) Only the specific terminology 'count-controlled loop' secured the mark. A clear statement that the this is the most appropriate when the nature of the task meant the number of iterations is known secured the second mark. This required a clear explanation and often statements such as *'the loop will be performed a number of times'* fell short.

(b) The vast majority of candidates secured only two of the available six marks for the number of iterations column and the first entries in the next three columns. It is suspected that it was the changing values within the `Chars` array that the candidates found too demanding.

Question 5

Many candidates answered the question with only an explanation of the general operation of a stack and the general operation of a queue, paying no attention to the particular task which was required by the question. This often involved some detail about the use of the pointers for each data structure but again failing to relate this to the particular problem the question required. Some candidates have some idea that the task involves removing the items from the stack on to the queue but nothing markworthy after that.

However, there were some answers seen which did gain full credit.

The key stages required were:

- removing items from the stack
- until the stack was empty
- removing each item from the queue on to the stack
- until the queue was empty.

Those four stages would have secured four marks.

The remaining marks were gained for an explanation of how the stack pointer, and the front and rear pointers of the queue were used to implement these data movements.

Question 6

(a) Most candidates gained some credit but rarely the full four marks.

If the rubric of the question states *'the sensors cannot read values less than 0'*, then showing a negative test data value is a fundamental error. Similarly, a text value was not appropriate or filling all of the rows with a value described as 'normal'. The critical values of 0 and/or 60 were usually identified but often the description given for the type of test data was inappropriate. The minority of candidates seemed unaware of the technical descriptors (as in the syllabus) of normal, boundary, abnormal and extreme. Candidates often used other descriptors such as low, high, limit – none of which gained credit.

(b) The second pseudocode question provided a varied level of responses.

Common errors/omissions included:

- the use of single loop – not two
- the use of a count-controlled loop instead of a conditional loop for the outer loop
- a boundary value of 2000 (not 1999) for the inner loop
- omission in the design of a decreasing boundary value
- omission of a structure which terminated the outer loop when the list was completely sorted
- the swapping of only one of the sensor value or sensor ID – not both.

Question 7

(a) The pseudocode required here proved challenging for many candidates and many scripts were seen with little or no attempt at the solution. The most successful answers seen had a solution which picked out the description in the question rubric stating what happens when a single cup of coffee is ordered.

The design then required pseudocode to deal which the purchase of multiple coffees. More able candidates gave a solution which included a `WHILE` loop with the content decreasing by 11 on each iteration. This often had flaws but did gain credit.

(b) (i) Many weaker answers were unable to score any marks. The solution required the file to be opened in `READ` mode, then closed before it was opened a second time in `APPEND` mode. The remainder of the design required no demanding techniques only those common to many file handling problems: These included:

- the reading of a line from the file
- extraction of the `CustomerID` using the `MID` function.
- its conversion to a number value
- a count-controlled loop to write the new data values to the file.

(ii) Many week answers seen. If any marks were secured, it was usually for stating that the file must now be opened in `WRITE` mode and/or that this customer must be given the customer ID `100001`.

Other aspects of the issue were rarely described, such as:

- a check must first be made to establish if the file exists or if it does, check if it is empty
- once the first customer has been addressed then the existing `AddNewCustomers` procedure can be used.

COMPUTER SCIENCE

Paper 9618/22

Fundamental Problem-solving and Programming Skills

Key messages

The emphasis for this paper is on the application of practical skills involving both computational and algorithmic thinking. Candidates need to have developed these and be able to apply them to the scenarios presented if they are to achieve high marks.

The art of problem-solving involves firstly understanding the requirement. In this paper the requirement is often presented in the form of a scenario description. Many candidates failed to demonstrate competence in terms of designing and implementing a solution to a problem. Candidates need to be able to identify the key elements of each question which, for example, could include the need for an iterative structure or methods to access data stored in a file. The development of these skills requires practice.

When writing algorithms many marks are lost by candidates who do not pay attention to key points of detail when writing pseudocode statements. Some of the more common errors seen, which would all result in a loss of marks, are listed below:

- A correct procedure/function header but the end statement is omitted (**Question 7(a)** and **7(b)**).
- Incorrect use of functions and operators given in the insert (provided in the examination), specifically but not exclusively the `RAND` and `INT` functions (**Question 3**). Other examples include `LEFT` or `MID` functions (**Question 5(b)**).
- Incorrect use of brackets for identifying the required element in a 2D array (**Question 7(a)**).
- An `IF` statement with the `ENDIF` omitted.
- A loop with the terminator statement omitted; for example, `ENDWHILE.`
- File handing statements incorrect (**Question 5(b)**):
  - The file opened using the wrong mode.
  - The file name omitted from the `CLOSEFILE` statement.
  - The file name omitted from the `EOF()` function.
- Using keywords for a variable identifier. For example, `String` used to store a line of text read from a file (**Question 5(b)**).
- `OUTPUT` statements where the '`&`' (ampersand) has been incorrectly used or the '`,`' comma omitted.

General comments

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. An example is **Question 2(b)(iii)**, where many candidates failed to perceive the signposting in the stem and incorrectly gave 'Syntax' as a type of error.

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

Familiarity with fundamental programming concepts is vital. Lack of understanding is often illustrated by the confusion between a literal value and an identifier (such as when using a string as a filename), or the misuse of `OUTPUT` in place of `RETURN`. Many candidates often replaced parameters to a procedure or function with a series of prompts and inputs within the body of the subroutine.

The following examples illustrate invalid pseudocode constructs of the kind that have been seen in this series:

- An invalid Boolean statement (**Question 3**):
  ```
  IF Num1 AND Num2 < 0 THEN
  ```

- Invalid loop constructs:
  `FOR Count ← 1 TO EOF("TimeTaken.txt")` (**Question 5(b)**)

  `FOR Index ← 1 TO LENGTH(Loyalty)` (**Question 7(a)**)

- The invalid use of the logical operator `AND` (**Question 7(b)**).
  ```
  OUTPUT Loyalty(Index, 1) AND "has loyalty points above 10"
  ```

Comments on specific questions

Question 1

(a) (i) Few candidate answers gained this mark; those that did were usually for a mention of budget/time or the need for rapid development/prototypes.

Most candidates tried to explain the benefits of using a programming life cycle rather than the need for *different* program development life cycles, The syllabus states in ***section 12.1***: 'Show understanding of the need for *different* development life cycles'.

Many candidates often gave answers such as 'to make the program easier to debug' or simply 'so that the program will run properly'. These candidates seemed to concentrate solely on the word 'program' in the stem rather than the whole question.

(ii) Due to a series-specific issue with **Question 1(a)(ii)**, full marks were awarded to all candidates for this question to make sure that no candidates were disadvantaged.

(b) (i) Most candidates gained at least one mark for this question. The most popular answer was a change of user requirement. Often this was given more than once. Legislative change was probably the next most common correct reason.

Vague answers were common e.g. 'to allow the program to do more things'.

Many candidates answer gave reasons that related to other kind of maintenance such as perfective and corrective instead of adaptive maintenance. Examples of these types of answers included: 'fixing bugs' and 'making the program run better'.

(ii) Due to a series-specific issue with **Question 1(a)(ii)**, full marks were awarded to all candidates for this question to make sure that no candidates were disadvantaged.

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

Common mistakes were `CHAR` for row 1 and `INTEGER` for row 4.

(d) (i) Candidates were presented with a pseudocode statement that assigned a value to an element in an array and asked to state the dimensions of the array:

```
Product[x, y] ← 23
```

Only 1D and 2D arrays are mentioned in the syllabus and most of the candidates answered this question correctly.

Many incorrect answers included 23 (the value used in the example) and 891 (the product of the upper and lower bound values).

(ii) As for previous question, a variety of answer were seen. Just over a third of candidates gave the correct answer.

The most common incorrect answer seen was 891, but others commonly seen were 90, 100, and 2.

(iii) Most candidates gained at least one mark, with many gaining both marks.

Common mistakes included the use of double brackets for the indices, omitting the keyword `ARRAY` or jumbling the word order and minor errors such as a missing colon and use of `AS` in place of `OF`

Question 2

(a) A small number of answers gave a textbook description: 'To break down the problem to a level of detail/small steps from which it can be programmed'.

Candidates need to understand that 'stepwise refinement' is an activity from the design stage of program development. Many candidates incorrectly gave answers describing the breaking down of 'a *program*'. Although these were potentially able to go on to gain the second mark.

Where answers did refer to breaking down a problem the second mark point was often lost due to vague description such as 'make the problem easier to solve'.

Answers that were not mark worthy often referred to checking the program (and sometimes the data) and also 'reading/running the program line by line'.

(b) (i) An accessible question with most candidates gaining both marks.

A common mistake was to offer a range rather than discrete values. Often these answers gained a follow through mark for the second test.

A small number of answers gave values for bonus pay that were not in the table.

Only a very small number of repeat answers were seen for the second test (different values but same bonus pay)

A small number of answers that did not answer the question were seen, usually suggesting test data types (normal, boundary etc.) but without giving specific valid test data values as asked to do.

(ii) Around a quarter of the candidates gained the two marks available for this question.

Many answers read as if the stub module was somehow actively testing the rest of the program. The phrase 'to simulate the program' was often seen. In many cases it was unclear that the dummy module would be used in place of the unfinished one.

A large number of answers described integration testing rather than stub testing.

(iii) Most candidates gained one of the two marks available for this question. Fewer gained both marks, but most candidates performed better on this question than the previous one.

'Logic' and 'run-time' were seen frequently, and these two correct answers were seen the approximately same number of times. The 'unexpected result' description of a logic error was commonly seen, and the acceptable alternative description of 'an error in the logic of the algorithm' gained many answers a second mark.

In the case of run-time error, the second mark was more rarely given, with 'divide by zero' being not enough for many.

‘Syntax error’ was probably the most common incorrect answer, suggesting that candidates had not appreciated the scenario. General ‘validation errors’ were given such as range and type.

A number of no responses were seen.

Question 3

Some very good answers were seen, many being very similar to the example solution given in the mark scheme.

A number of no responses and non-markworthy answers were seen.

A significant number of solutions did not contain a loop, which was necessary. Some tried to generate two different numbers by using an `IF` statement to re-generate one of the values if they were the same.

Breakdown by mark points.

MP1: Most often given

MP2: Usually given, although a very common mistakes included attempting to provide two parameters and also including the return type e.g. `RAND(-10, 10)` or `RAND(10 : INTEGER)`

MP3: Commonly given often in conjunction with MP2 but occasionally as a follow through if MP2 was not given as described above.

MP4: Many solutions did not include this feature.

MP5: As this required the fully correct use of `RAND()` it was only given to a minority of solutions.

MP6: Often missed due to the lack of a loop, the output statement was commonly included with the solution to MP7 and MP8, outputting either two or three values as necessary.

MP7: Usually close behind MP1 in terms of awarding frequency. A common mistake was the misuse of the logical operator: `IF Num1 AND Num2 < 0 THEN`

MP8: Together with MP5, this was the most challenging mark due to the need for an appropriately generated random value. In addition, marks were often lost through the attempted concatenation of a numeric value in the output statement.

Several correct solutions generated a value using `INT(RAND(36))` inside a conditional loop that repeated until the value was `>= 30`

A significant number of candidates attempted solutions that were over complicated using arrays containing the valid range of values then randomly generating an index for a specific element in the array. Where this was done correctly marks were awarded as appropriate.

Question 4

A seemingly simple scenario, this attracted a very wide range of answers that tended to be at the extremes – ranging from solution that was the same as the example solution given in the mark scheme through to those suggesting little or no experience of producing program flowcharts. Often the latter were little more than a vertical series of boxes with the bullet points from the question written one per box.

Many solutions lacked continuity, with missing outputs from shapes or lines that mysteriously branched.

Many candidates did not appreciate that a single identifier could be used as both an array index and a loop counter.

Mark point breakdown:

MP1: Frequently given, in the case of weaker solutions often as the only mark. This mark was lost for placing the assignment inside the loop and occasionally using a different variable as the array index in MP4.

MP2: Many solutions attempted this, but often the number of iterations was 99 or 101. A mistake seen at times was testing the input value rather than a loop counter.

MP3: As for MP1, given to most viable solutions.

MP4: Again, given in most cases.

MP5: Often as a FT from MP2. Some solutions gained this mark by the use of two loop counters, initialising both at the start.

MP6: A straightforward mark and often given. Weaker solutions tended to miss the need for an output loop.

MP7: As for MP2, sometimes the number of iterations was out by one.

Question 5

(a) Generally well-answered, with most candidates gaining full marks. Events were usually correctly drawn (sometimes with the arrow at the mid-point) and occasionally structure chart arrows were used alongside event lines suggesting some confusion but generally this was condoned providing the meaning was clear.

A common mistake was to duplicate the states, perhaps not noticing that several rows of the table mapped to the same ‘Next state’.

There were a small number of no responses.

(b) Some very good solutions were seen, with many gaining six marks. MP6 (first mark scheme solution) proved the most challenging, and in this regard, solutions based on the alternative array-based approach had an easier task to perform and in general it was this latter type that tended to gain all 8 marks.

A significant number of candidates gained two marks or less.

Mark point breakdown (first solution):

MP1: Frequently given and in some cases as the only mark.

MP2: Often not given. Mark lost due to missing quotes around literal filename and missing parameter in `CLOSEFILE` (or statement absent completely). There seemed to be a tendency to assign the literal string to a constant and then to use that, which is good practice. The file mode was usually correct but `WRITE` and `APPEND` were both seen occasionally.

MP3: Missing brackets (on top of missing quotation marks) lost many marks. A minority of candidates suggested a hybrid loop structure i.e. `FOR index ← 1 to EOF("TimeTaken.txt")`

MP4: Generally given in most viable solutions. An occasional mistake was to overlook that this was a text file and to attempt to reference a specific line in the file (random-access) e.g. `ThisLine ← READFILE (Myfile, Index)`

MP5: Usually given as second half of the pair of marks with MP4. Occasional error was to use `RIGHT()` instead of `LEFT()`

MP6: A challenging mark point to achieve. Various design approaches: a fairly common one was to place the conditional loop (MP3) inside a count-controlled loop e.g. `FOR Hour ← 0 TO 23`. This almost always failed as by the time the outer loop entered the second iteration (for hour 1) the file had already been read to the end. A more basic error was to make the outer loop run from 1 to 24.

An alternative approach was to initialise hour to some rogue value before the MP3 loop, but this usually led to an unwanted output statement.

MP7: Often awarded as a follow through when an attempt had been made at MP6

MP8: A challenging multi-part mark, this was rarely given. Common mistakes included: outputting a message for each hour regardless of the number of pictures taken, use of the concatenate operator with a numeric value in the output statement and resetting `Count` to zero rather than 1.

MP9: Rarely addressed.

Note this was an eight-mark question.

Mark point breakdown (second solution – array solution) where different from above:

MP1: Often omitted in weaker solutions.

MP6: Usually followed MP5 extraction. An occasional error was to replace `STR_TO_NUM()` by `INT()`

MP7: A straightforward mark. Usually attempted but often missed if the array had been declared as `(1:24)` rather than `(0:23)`

MP7: Many solutions simply attempted to output all 24 count values in a loop without checking for zero. Many examples of using the concatenate operator with a numeric value were seen.

Question 6

(a) (i) Well answered by most of the candidates, with many full marks given. Most candidates gained at least the first mark.

Where marks were lost, it was usually down to different values appearing in any of the first four columns or below the region 4 area.

(ii) Well answered by most of the candidates, most candidates gained at least one mark with many full marks given.

Many missed MP3 by omitting the last two zero values in the `Data` array.

(b) Many candidates identified the linked list mark, although both stack and queue were suggested fairly frequently.

The second mark was often missed due to vague and imprecise use of language. A common mistake was to suggest that the procedure was implementing a (bubble) sort.

A small number gained MP2 for a reference to how one of the arrays was manipulated.

Question 7

(a) A number of different approaches were seen, with conditional loops being the most popular, closely followed by count-controlled loops. Several ‘hybrid’ solutions were seen that were based on nested loops.

The ‘direct access’ and binary search solutions were rarely seen. When they were presented, the direct access solution tended to gain high marks and the binary search around half-marks.

Some very good solutions were seen but also many that gained three marks or less.

Very few solutions attempted to validate the `CustomerID` parameter.

Many no responses seen.

Mark point breakdown (conditional loop solution):

MP1: Frequently given, in some cases often as the only mark.

MP2: As mentioned, rarely seen. Where attempted, the conditional clause sometimes omitted the final `ENDIF`

MP3: Often implemented as a `REPEAT` ... `UNTIL` loop, a common mistake was to omit checking the index in the termination condition, relying instead on testing a flag which was set when the `CustomerID` was found (MP6) or `9999` was found (MP7).

A fairly common error was to use the assignment symbol in the test e.g. `UNTIL Flag ← TRUE`

MP4: One of the more accessible marks and often given. Invalid indices 'syntax' was not uncommon e.g. Loyalty(Index)(1) and treating Loyalty as a 1D array was seen occasionally. Also seen occasionally was the incorrect inclusion of an `ELSE` clause with an immediate `RETURN` if the `CustomerID` was not found at the current location.

MP5: Often given, albeit sometimes as a follow through following an incorrect MP4.

MP6: The 'matching' mark for MP5 and one of the more accessible marks, frequently given.

MP7: Probably the least popular 'loop termination' test, this was given mainly to higher-scoring solutions.

MP8: A simple conditional test gained this for most of the better solutions. Many solutions had implemented an immediate return if the `CustomerID` had been found so merely had to return –1 following the loop.

Weaker candidates made the mistake of outputting the value or even returning the string '–1'.

Mark point breakdown (count-controlled loop solution) where different from above:

MP3 and MP4: Often given as a pair in most solutions.

MP6: Regularly given. Solutions based on both immediate `RETURN` and storing the points value were seen.

MP7: Rarely seen.

MP8: Usually implemented by a conditional clause following the loop.

(b) Many good solutions gaining five marks were seen, but also many that gained only one or two marks.

Many no responses seen.

Mark point breakdown:

MP1: Frequently given, in some cases often as the only mark.

MP2: Attempted in many solutions, often correctly.

MP3: Most viable solutions gained this mark for a straightforward count-controlled loop. Some solutions incorrectly used the `CustomerID` as the index value.

MP4: Rarely seen.

MP5: Usually implemented in most viable solutions and one of the more accessible marks.

MP6: Normally seen in conjunction with MP5.

MP7: Only the better solutions gained this mark – mostly solutions ignored the column 1 value, so the value of unused elements was added to the sum.

MP8: Although available as a follow through following a reasonable attempt at MP7, there were many reasons why this mark was not given: declaration of Average as an `INTEGER`, dividing by 1000 rather than `Count`, and using '&' incorrectly or '+' in the output statement.

(c) (i) Many candidates failed to clearly express the point that an array can only store values of a single data type.

(ii) A small number of candidates suggested the use of a record type but only a small percentage of these went on to clearly state that an array of this new data type would be declared. Marks were lost through imprecise statements.

The available alternative mark for mentioning conversion and concatenation was rarely given. Again, where this approach was suggested, the actual language used was often vague and imprecise.

Many answers incorrectly suggested the use of multiple arrays.

COMPUTER SCIENCE

Paper 9618/23
Fundamental Problem-Solving and Programming Skills

Key messages

Many candidates would benefit from further development of basic design skills which are needed to map to a given task or problem. All aspiring programmers will at some stage be too keen to move immediately to the coding stage. This, however, can be at the expense of a poorly designed algorithm, and this haste should be discouraged.

Decisions such as – is the problem best coded with:

- a count-controlled loop or a conditional loop?
- Shall I use multiple `IF` statements or a `CASE` structure?
- What are the parameters needed for the function/procedure header?
- What data type should be declared for the various variables?

Once the basic design is in place, there is no substitute for a detailed knowledge and its application of the CIE pseudocode language. Whilst the list of functions is provided with the examination paper insert this is no substitute or help with the understanding of the key structures and their application to a given problem.

Some programming basics appear not to be well understood, and these are manifest when the coding contains:

- Output of a variable when the rubric of the question says it should be returned by the function
- The use of reserved words for identifier names. For example: the use of `LENGTH` in **Question 5** and `MONTH` and `TODAY` in **Question 7**
- A loop structure where the end of the loop is omitted (no `NEXT`/`ENDFOR`, `ENDWHILE` or `UNTIL`)
- A file open statement with the mode type or file close omitted.

Lack of detail will inevitably loose marks. An example was seen in **Question 3**; the variables must be declared – the total to be calculated must be initialised and the variable used are declared and of the appropriate data type.

It was encouraging to see the marked improvement in the standard of answers seen for **Question 4(a)** where a general description had to be written as a sequence of steps ready for coding.

General comments

There were some impressive and high scoring scripts seen.

Computer Science is a technical subject, and such subjects will use their own technical language. Each section of the paper 2 syllabus has technical terms which are required in answers on this paper. Often, they will allow simplicity in a candidate's answer. An example from this paper **Question 1(a)(ii)**; a detailed explanation of where/where not local variables would be understood and its implications, could be simplified with a simple statement using the technical term 'scope' such as *'the scope of a local variable is only inside the module in which it is defined'*.

More able candidates have the confident to amalgamate pseudocode structures. An example would be in question 3 where a random integer in the range one to six had to be generated. The statement:

`RollValue ←INT(RAND(6)) + 1` secured three marks.
A less confident candidate may code this with two – possibly three – statement is in sequence:

```
RollValue ← RAND(6)
RollValue ← INT(RandNum) + 1
```

Both solutions would secure the maximum three marks. Centres need to assess their candidates' strengths and decide if this this combination approach should be encouraged. It is noted that the final row in **Question 1(b)** required this understanding, and this was understood by almost all candidates.

A further example, is given in the general comments for **Question 7(b)(i)**.

Comments on specific questions

Question 1

(a) (i) A varied level of responses seen. A number of candidates did not realise the starting point for the thinking was that we have a major project for which we are considering the design.

The answers looked for an appreciation that specific parts/tasks which make up the solution could each be implemented as a module. Candidates were often too keen to immediately give answers which related specifically to program code that was to be written. Full marks however could be scored here stating that the individual modules should reduce the complexity of the program code and make the code therefore easier to test and maintain.

Examples of weak answers which would not have gained credit were:

'Make the programming easier' or *'Saves the programmer time'* as a non-worthy attempt at describing that the module can be repeatedly called.

(ii) Generally well understood. Answers simply stating that a local variable is recognised only within the particular module, within which it has been defined gained credit; conversely global variables are recognised throughout the main program. Very few answers used the technical term 'scope'.

(iii) Most candidates were able to give one benefit, and the most popular answer was that this allowed the same identifier name to be used both within other modules and/or the main program. A more subtle answers which was not uncommon was that the use of local variables makes individual modules self-contained.

'Local variables need to be declared not global variables do not' was not uncommon and showed a lack of understanding about programming basics.

(b) Well answered by the majority of candidates often scoring all of the available five marks.

(c) It was rare for the candidate not to score the available four marks. Errors seen were the use of `INT` as a shortened form of `INTEGER`. This is not allowed as `INT` is one of the functions documented in the Insert document. Some candidates did not appreciate there is available a `DATE` data type and so typically wrongly suggested `STRING` for the `MemberDOB` variable.

Question 2

(a) (i) The majority of candidates were able to score the two marks appreciating the difference between the address of the memory location and the data value it contained. Candidates who did not have this clarity wrongly stated the value 'B' as the memory location.

(ii) Well answered. The majority of candidates scored at least two of the available three marks. The final mark was usually lost as the candidate did not label location 408 as `TopOfStack`.

(b) (i) This is a good example where the candidate needs to understand the computing terminology being used; essentially:

- use of a record data type consisting of a Data value and a Pointer
- an array of this data type
- implementing a linked list with the data in the array
- using the index to address individual items of the array.

There are many computer science concepts all contained here in a single paragraph and diagram.

The standard of answers seen varied. Weaker answers did not show an understanding of the mechanics of the linked list but intuitively guessed that Uranus would be the last item in the list.

(ii) Again this required the clarity of terminology required for the earlier part **(a)(i)**.

(c) Generally, well answered with the majority of candidates able to score one of the two available marks. Most common answers stated that the mechanics of the queue are ‘First data item in will be the first to leave’ (or variants) and that the queue requires a front of queue and rear of queue pointer to operate. Less common but worthy answers stated that the queue can operate as a linear list or can be circular, or that the queue would be managed using Dequeue and Enqueue modules.

Question 3

The first pseudocode question on the paper and generally well answered. Many candidates were able to secure the full seven marks. Some answers declared a variable for the final average value – other solutions outputted the calculation of the total divided by the function parameter. Detail was required that all variables had been declared and the average value if used was of data type `REAL`. Also, the total had to be initialised.

Some candidates confused the number 6 on the dice with the number of iterations of the loop. There were three marks available for the correct expression to calculate each random number; use of the `RAND()` function followed by use of the `INT()` function and then the addition of 1 to generate the number in the correct range. Candidates often failed to secure the third mark.

Other not uncommon errors included:

- The omission of the `OUTPUT` statement for each number generated
- One or more errors related to the final value returned. This included the average variable declared as `INTEGER`, the output (not return) of the final value or use of the `DIV` operator.

Question 4

(a) A good discriminator question where all candidates were able to secure some of the marks. Most candidates described a loop either looping until the end of the file was reached, or by describing a count-controlled loop. However, it was often unclear where in the description the loop began and ended. Many answers correctly stated the key point that on each iteration the algorithm will read a line from the file. However, vague statements such as *‘read all the lines from the file’* gained no credit. The answer had to make it clear that it was single line being read. Candidates appreciated that the value read must be converted to an integer value before a comparison with 500 could be made.

Some answers described each matching value being stored temporarily in an array and then this list of values being output after the initial loop had terminated. This was an acceptable alternative solution.

Lack of detail, such as the omission of closing the file or omission of the `READ` file mode.

(b) Generally, well answered with the range of candidates abilities all able to score marks. It is appreciated that the term ‘walkthrough’ can be used to describe both testing of the solution by colleagues and peers or the detailed examination of the program code. The mark scheme catered for both interpretations. Most marks were gained for the description of a trace table being used to carry out a dry run of the program, examining the code line by line. The use of test data was described either with statements which described a set of test data/inputs not producing the expected outputs. A more general description of the use of test data which included normal, erroneous and extreme data was acceptable.

Question 5

A more challenging pseudocode question than **Question 3** and with more varied responses.

Most solutions demonstrated an understanding of the basic design required. However, marks were frequently lost for a variety of reasons:

- The identifier name `BitString` given in the rubric of the question was not used.
- `Length` was frequency used as an identifier for the length of `BitString`.
- The count variable had not been initialised before the loop.
- Using a mixture of data types for a statement e.g., concatenating a number value with `BitString`.
- Use of the `+` operator to concatenate two strings.
- Omission of the final `ENDIF` in the code block to append the 0 or 1 character.

Most solutions did correctly use the `MOD` operator to test for an even/odd number.

Some solutions attempted to treat the string as an array of characters and hence index individual characters. This is present in some programming languages but not in the CIE pseudocode language.

Question 6

(a) Generally, well answered with candidates stating that a function will return a single value but procedures do not. A general statement such *as 'functions return values …'* was considered creditworthy. Technical language was important here and other descriptors such as *'a variable is returned'* was not sufficient.

(b) Generally, well answered with most candidates able to score 4 or 5 of the available marks. The requirement for the third module to pass the `U` variable by reference was well understood. The only (and frequent) error was often to incorrectly define the third module as a function.

(c) Two of the three available marks here requiring only the technical terms selection and repetition. Alternatives were considered acceptable for the second term. For the third mark the detail of what each module does was required. Often the calling module name – Main and Sub_A – was not mentioned and therefore penalised.

Question 7

(a) (i) A pseudocode question with no loop required for the design and generally well answered. The frequent error was that the solution did not use the `TODAY()` function for today's date. However, solutions did attempt to compare two values using the `MONTH()` function. Using `Month` or `Today` as an identifier name was a not uncommon error.

Some candidates set the condition testing for both the month and the current year. The inclusion of the year clause was not expected.

The `IF` statement was usually formed with the correct syntax and logic.

(ii) A varied level of response. The statement required followed through from the variable names which had been used in part **(a)(i)**. Usually, the comparison of two expressions both using the `DAYINDEX()` function secured the mark.

(b) (i) The most demanding pseudocode question. Many candidates correctly used the `FindCustomer()` function with the parameter given in the `MondayCheck` header.

The task then was to isolate the characters which represented the number of loyalty points. Solutions adopted one of two alternative strategies.

The simplest approach was used by candidates who realised that all the other characters in the string – the six character customer ID number, the eight character date and the two comma characters – always totally to sixteen characters.

Hence the length of the loyalty points string was then a simple subtraction.

Some candidates incorrectly followed the example given in the question rubric and assumed that the loyalty points string was always three characters.

The more complex approach was to use a conditional loop and find the position in the string of the second comma character.

There were some exemplar solutions seen which used complex statements which could then attract several of the available marks with a single statement.

An example would be:

```
LPoints  ←  STR_TO_NUM(MID(FindCustomer(CustomerID), 8, LENGTH(Line)-
16))
```

That statement would have secured five marks. It is suggested this approach is only encouraged by centres with the most able candidates.

(ii) Well answered. Candidates correctly first isolated the various parts:

- The day number using the `LEFT()` or `MID()` function
- The month number using the `MID()` function
- The year number using the `RIGHT()` or `MID()` function.

Each was then converted from a string to a number and used as the three parameters for the `SETDATE()` function.

Some solutions ignored that declaration of the integer values which were to be used and instead used the reserved function names `Day`, `Month` and `Year`.

COMPUTER SCIENCE

Paper 9618/31
Advanced Theory

Key messages

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, using relevant technical terminology, 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 in order to gain full credit.

Candidates are further advised to make use of the published pseudocode guide when preparing for this examination, especially in the areas of user-defined data types, algorithm construction or file handling. When answering questions that require responses to be written in pseudocode, candidates should use this syntax.

General comments

To maximise their mark, candidates are advised to read questions carefully before beginning their answer in order to understand what is being asked of them. Candidates should be careful to answer the question that has been 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 also advised to use the correct computer science language and technical terms when answering questions.

Comments on specific questions

Question 1

(a) This question was generally answered well. Most candidates achieved at least one mark for their pseudocode statement to declare the stated enumerated data type.

(b) This question was mostly well answered. Most candidates achieved at least one mark and many achieved high marks. Common errors seen included using incorrect data types for some of the fields or either missing `DECLARE` or not putting this in the correct place with each field.

Question 2

(a) Most candidates achieved at least one mark for writing the given binary number in its normalised floating-point form for the given system, with many of these achieving both marks. The most common error seen was an incorrect exponent.

(b) This question was mostly answered well. Credit was given to candidates where correct elements of their working were seen, even if they did not achieve the correct final answer.

Question 3

(a) Candidates who gave answers describing the purposes of the Application and Transport layers of the TCP/IP protocol suite in terms of their role in preparing data for transmission or receiving data and preparing it for a user, rather than concentrating on which other layers they passed data to or received data from, achieved the best marks.

(b) A range of responses were seen for this question. Many candidates achieved high marks for describing the process of packet switching as a method of transmitting messages across the internet. Some candidates confused packet switching with circuit switching. Some candidates incorrectly gave benefits and/or drawbacks of packet switching, which was not the focus of the question.

Question 4

(a) Candidates who added four new nodes in the correct positions with the correct connections, and who also labelled all the null pointers, as required, achieved full marks. Most candidates achieved at least one mark for this question, but common errors included nodes in the wrong place, missing arrow heads on the pointers or pointers not originating from the correct pointer box.

(b) Most candidates demonstrated that they knew what recursion is and achieved at least one mark. Candidates who fully described the general case and the base case achieved both marks.

(c) Most candidates were able to identify an Abstract Data Type that can implement recursive algorithms, other than a binary tree.

Question 5

(a) Most candidates correctly wrote the Boolean logic expression corresponding with the given truth table as a sum-of-products.

(b) (i) Candidates who correctly filled the Karnaugh map with both 0s and 1s achieved both marks.

(ii) Candidates who correctly identified two appropriate groups in the Karnaugh map and drew loops around them achieved both marks.

(iii) Candidates who wrote the expected two term simplified Boolean sum-of-products, taken directly from their Karnaugh map, achieved both marks.

Question 6

Candidates were generally able to describe parts of the process of executing a program using an interpreter and achieved some marks. Some candidates incorrectly described compiling or processes within compiling rather than interpretation.

Question 7

(a) Candidates who gave clear reasons why `#Jd7` and `C%6A` are not valid passcodes, achieved both marks.

(b) Candidates were aware that to answer the first part of this question, they needed to list all the acceptable uppercase letters included in the syntax diagram `uppercase`. Candidates who achieved this without including an ‘or’ symbol, ‘`|`’, at either the start or end of their list, achieved the mark. Candidates found the second part of the question more difficult. To achieve full marks, candidates had to include recursion in their final answer. Many good responses were seen, with those candidates achieving high marks.

Question 8

(a) Most candidates were aware that multi-tasking allows multiple processes to run concurrently and achieved the mark. Those who also stated a benefit to process management of this achieved both marks.

(b) Many candidates were able to achieve a mark for stating that the shortest remaining time scheduling routine processed jobs with the shortest burst time first. Some of these candidates also achieved the mark for giving a benefit, such as the waiting time is minimised. The higher scoring candidates gave more detail as to how the jobs were ordered for processing or described the fact that this scheduling routine is pre-emptive.

Question 9

(a) The question was generally well answered, with most candidates able to name two functions of Secure Socket Layer (SSL)/Transport Layer Security (TLS).

(b) Most candidates were able to give two good examples of situations where the use of SSL/TLS would be appropriate.

Question 10

(a) Most candidates achieved at least one mark for this question, with many candidates able to describe the purpose of a graph when used in an Artificial Intelligence (AI) system, such as recording relationships between entities using nodes and edges.

(b) Most candidates achieved at least one mark. Candidates who explained the use of artificial neural networks in Deep Learning in terms of their being designed to function similarly to the brain, their architecture, how they are structured and how the different elements work together, including how Deep Learning models learn from data, achieved the marks.

Question 11

(a) Many candidates were able to complete the class diagram for `Appointment` and achieve some marks. Common errors included inconsistency of naming across the various parts of the question. For example, the attribute for identification of the patient had to be `PatientID`, because the setter is given as `SetPatientID`. Similarly, its getter would need to be `GetPatientID()`. Also, any parameters given in the setters are expected to have different names to their respective attributes, so if the attribute is `Doctor`, the parameter in the setter would have to be something different, such as `DoctorID`. Candidates whose answers followed these expectations correctly achieved the highest marks.

(b) (i) Most candidates recognised that the object-oriented programming (OOP) feature whose function includes restricting external access to the data is encapsulation.

(ii) Most candidates were able to describe the OOP feature, inheritance, and achieve at least one mark. Those who gave a full description achieved both marks.

Question 12

The full range of marks was awarded for the completion of this algorithm. Candidates with a good knowledge of file handling methods from the pseudocode guide achieved the highest marks.

COMPUTER SCIENCE

Paper 9618/32
Advanced Theory

Key messages

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, using relevant technical terminology, 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 in order to gain full credit.

Candidates are further advised to make use of the published pseudocode guide when preparing for this examination, especially in the areas of user-defined data types, algorithm construction or file handling. When answering questions that require responses to be written in pseudocode, candidates should use this syntax.

General comments

To maximise their mark, candidates are advised to read questions carefully before beginning their answer in order to understand what is being asked of them. Candidates should be careful to answer the question that has been 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 also advised to use the correct computer science language and technical terms when answering questions.

Comments on specific questions

Question 1

(a) Most candidates achieved at least one mark for storing a set of data in a given record data type. Many of these candidates achieved high marks. Common errors seen included using incorrect data types for some of the fields or either missing `Flight1` or not putting this in the correct place with each field.

(b) (i) Many candidates achieved at least one mark for a partially correct declaration statement for the enumerated data type `GateID`, with most candidates achieving both marks.

(ii) Most candidates were able to write the new statement for the declaration of `Gate`. A common mistake seen was `Gate` and `GateID` being the wrong way round in the declaration statement.

Question 2

(a) This question was mostly answered well. Most candidates who gave the correct answer, including full working to show how they had arrived at their answer, achieved full credit.

(b) Many candidates demonstrated correct methods to calculate the denary value of the given negative binary floating-point number and achieved one or two marks for their working. However, some of these candidates did not achieve the correct answer mark due to arithmetic errors, or not taking their calculation far enough to arrive at the correct final answer. Some candidates forgot that their answer should be a negative number.

Question 3

(a) Most candidates correctly named two different layers of the TCP/IP protocol suite.

(b) Candidates whose answers concentrated on describing the TCP/IP protocol suite as being viewed as layers within a stack, with the user interfacing with the Application layer, the network interfacing with the Link layer and each layer only communicating with adjacent layers, achieved some or all of the marks. Many candidates incorrectly described the function of the individual layers of the TCP/IP protocol suite, despite a statement in the question asking candidates **not** to do this.

Question 4

Candidates were generally able to name benefits and drawbacks of circuit switching. Candidates who could name two of each and who did not attempt to compare them to another method of data transmission, such as packet switching, achieved the highest marks.

Question 5

(a) Most candidates achieved at least one mark for naming at least one process state. Many of these candidates were able to name more than one, so achieved higher marks.

(b) Most candidates were able to achieve a mark for stating that the shortest job first scheduling routine processed jobs with the shortest burst time first. Some of these candidates also achieved the mark for giving a benefit, such as faster throughput of processes. The higher scoring candidates gave more detail as to how the jobs were ordered for processing or described the fact that this scheduling routine is non-pre-emptive.

Question 6

(a) Candidates were generally aware that a graph, as used in an Artificial Intelligence (AI) system, consists of nodes and/or edges, and that these can be weighted. Those who made more than one of these points achieved both marks.

(b) Candidates were mostly aware that unsupervised learning uses unlabelled data and supervised learning uses labelled data. Candidates who went further than this, to describe the use of data with known outcomes in supervised learning, or data with unknown outcomes in unsupervised learning, achieved higher marks.

Question 7

(a) Most candidates achieved at least one mark for completing the truth table for the given logic circuit, with many of these candidates achieving more than one mark.

(b) Most candidates were able to write the Boolean logic expression corresponding to their truth table as a sum-of-products.

(c) A range of responses were seen for this question. Most candidates achieved at least one mark with many candidates achieving all or nearly all the marks.

Question 8

Candidates who gave a good explanation of the term lexical analysis including the steps involved during the process achieved the highest marks. The question was generally well answered with most candidates demonstrating knowledge of lexical analysis in program compilation.

Question 9

(a) The question was well answered. Candidates were able to give the clear reason why `9K` is not a valid variable.

(b) Candidates were aware that to answer this question, they needed to list all the acceptable symbols included in the syntax diagram `operator`. Candidates who achieved this without including an 'or' symbol, '`|`', at either the start or end of their list, achieved the mark.

(c) Candidates generally answered this question well. Common errors seen included repeat arrows for each of the last two boxes individually rather than together and a missing final output arrow.

(d) The question was well answered. Candidates achieved the mark if their variable example began with a letter and was at least four characters in length, using only the following:

Letters: A, B, C, D, E, F, G, H, J, K
Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Question 10

Candidates who named the handshake and/or record protocols achieved marks for correctly naming the TLS protocols. A number of these candidates also correctly stated the purpose of one or both of these, therefore also achieving those marks. Common errors seen for this question included candidates incorrectly naming Application layer protocols such as HTTP or SMTP.

Question 11

(a) Candidates who added the four new nodes in the correct positions with the correct connections, and who correctly labelled all the null pointers, achieved full marks. Most candidates achieved at least one mark for this question, but common errors included nodes in the wrong place, missing arrow heads on the pointers or pointers not originating from the correct pointer box.

(b) Candidates found the first part of the question more difficult. However, most candidates were able to name an example that could be programmed using a recursive algorithm.

Question 12

(a) Many candidates gained full or nearly full credit for this question. Common errors included inconsistency of naming across the various parts of the question. For example, the attribute for date of birth had to be `DateOfBirth`, because the setter is given as `SetDateOfBirth`. Similarly, its getter would need to be `GetDateOfBirth()`. Also, any parameters given in the setters are expected to have different names to their respective attributes, so if the attribute is `Priority`, the parameter in the setter would have to be something different, such as `PatientPriority`. Candidates whose answers followed these expectations correctly achieved the highest marks.

(b) (i) Most candidates recognised that the term to describe 'an occurrence of an object' is an instance.

(ii) Most candidates were able to successfully describe polymorphism.

Question 13

The full range of marks was awarded for the completion of this algorithm. Candidates with a good knowledge of file handling methods from the pseudocode guide achieved the highest marks.

COMPUTER SCIENCE

Paper 9618/33
Advanced Theory

Key messages

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, using relevant technical terminology, 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 in order to gain full credit.

Candidates are further advised to make use of the published pseudocode guide when preparing for this examination, especially in the areas of user-defined data types, algorithm construction or file handling. When answering questions that require responses to be written in pseudocode, candidates should use this syntax.

General comments

To maximise their mark, candidates are advised to read questions carefully before beginning their answer in order to understand what is being asked of them. Candidates should be careful to answer the question that has been 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 also advised to use the correct computer science language and technical terms when answering questions.

Comments on specific questions

Question 1

(a) This question was mostly well answered. Most candidates achieved at least one mark and many achieved high marks. Common errors seen included using incorrect data types for some of the fields or either missing `DECLARE` or not putting this in the correct place with each field.

(b) This question was generally answered well. Most candidates recognised the video format field as being the most suitable and were able to give the reason for this.

Question 2

(a) Candidates were mostly able to state the largest normalised positive two's complement binary number that could be stored in the given system, so achieved one of the marks. Fewer candidates were able to state its denary equivalent.

(b) This question was mostly answered well with candidates who gave the correct answer, including full working to show how they had arrived at their answer, achieving all four marks. Credit was given to candidates where correct elements of their working were seen, even if they did not achieve the correct final answer.

Question 3

(a) The vast majority of candidates correctly wrote the Boolean logic expression corresponding with the given truth table, as a sum-of-products.

(b) (i) Candidates who correctly filled the Karnaugh map with both 0s and 1s achieved both marks.

(ii) Candidates who correctly identified two appropriate groups in the Karnaugh map and drew loops around them achieved both marks.

(iii) Candidates who wrote the expected two term simplified Boolean sum-of-products, taken directly from their Karnaugh map, achieved both marks.

Question 4

(a) Candidates who gave answers describing the purposes of the internet and Link layers in terms of their role in preparing data for transmission, or receiving data and preparing it for a user, rather than concentrating on which other layers they passed data to or received data from, achieved the best marks.

(b) Most candidates answered this question well as they were able to describe the function of a router in packet switching.

Question 5

A range of responses were seen for this question. Most candidates achieved at least one mark.

Question 6

(a) Most candidates achieved at least one mark for describing why scheduling is necessary in process management. Common responses seen were to enable multitasking or to ensure fair access to resources, or similar.

(b) This question was generally answered well. Most candidates were aware that the round robin scheduling routine allocated a time slice to each process in turn. Those who stated that each process received a fixed time quantum, achieved the mark. Some candidates also achieved marks for describing the fact that round robin is a pre-emptive routine, or that once a time slice ends, the current process is returned to the queue and the next process receives its time slice. Some candidates also recognised that a benefit was that starvation of resources would not be an issue with this scheduling routine.

Question 7

This question was very well answered by most candidates. Most candidates achieved at least one mark, with candidates who used the correct technical terminology and the correct sequence of steps, achieving the highest marks.

Question 8

Many candidates demonstrated a good understanding of the second stage of program compilation, syntax analysis. Many high scoring responses were seen. Some common errors seen included candidates who described lexical analysis rather than syntax analysis, or candidates who described the process of program interpretation.

Question 9

(a) This question was well answered. Candidates who gave a clear reason why `JJ90` is not a valid passcode, such as, the second character must come from either lower or digit and not upper, where `J` comes from, achieved the mark.

(b) Candidates were aware that to answer the first part of this question, they needed to list all the acceptable uppercase letters included in the syntax diagram `upper`. Candidates who achieved this without including an 'or' symbol, '`|`', at either the start or end of their list, achieved the mark. For the second part of the question, candidates listed the acceptable combinations of `upper`, `lower` and `digit` that could be applied to make a valid `passcode`. This question was generally well answered.

(c) Candidates who added a single box with `upper` as an additional option for the third character of `passcode`, with appropriate connections, and who added a return arrow to enable the last digit to be repeated indefinitely, achieved the mark. A common error seen involved the addition of multiple other boxes, including `upper`, as options for characters in `passcode` other than the third.

Question 10

(a) Candidates who stated the purpose of the A* and Dijkstra's algorithms is to find the path between two points on a graph using the algorithm, or similar, achieved the mark.

(b) A good range of differences between the A* and Dijkstra's algorithms were seen, with many candidates achieving one or both marks. Common answers referred to the A* algorithm trying to find a better path by using a heuristic function, or the fact that Dijkstra's algorithm just explores all possible routes.

(c) Candidates were mostly aware that unsupervised learning uses unlabelled datasets to train it and that it identifies hidden patterns or clusters without the need for human intervention. Most candidates achieved one or two marks. Few candidates were awarded the third mark, as they needed to add that unsupervised learning is able to discover similarities and differences in the data.

Question 11

The full range of marks was awarded for the completion of this algorithm. Candidates with a good knowledge of file handling methods from the pseudocode guide achieved the highest marks.

Question 12

(a) Most candidates were able to name two Abstract Data Types, excluding an array, which was given in the question.

(b) Most candidates who wrote pseudocode for an insertion sort achieved at least one mark, with many candidates achieving higher. One common error that was seen was candidates reverting to a bubble sort.

(c) Most candidates identified two factors that would affect the performance of a sort routine, namely, the number of items to be sorted and how sorted the data is at the start. However, many candidates did not describe these ways, so it was unclear how they affected the performance of the sort routine. Candidates who gave answers such as the performance of the sorting routine should improve if the data is already partially sorted and the sort may take longer if the number of data items increases, or similar, achieved both marks.

Question 13

This question was answered very well. One common error that was seen was in the Index column. Some candidates incorrectly stopped it before it got to 9 and others let it continue past 9. Very few errors were seen in the other columns of the trace table.

COMPUTER SCIENCE

Paper 9618/41
Practical

Key messages

Centres should be submitting only one document for each candidate, this is the evidence document where all code needs to be copied and pasted as well as screenshots for testing. Candidates should be submitting black text on white background, without coloured fonts and highlighted sections. The evidence document must contain all work in the correct section for each question being answered, with the centre number and candidate number in header. Screenshots should only be used for the results of testing and these can include white text on a black background where that is standard for the Integrated Development Environment but should avoid all use of coloured font. This is to ensure the answers are legible.

Screenshots must be fully visible and include all inputs and outputs from that test run of the program. Some results may require multiple screenshots depending on the formatting of the outputs.

When copying code into the evidence document, candidates need to make sure the indentation remains when this is part of the syntax of the language, predominantly Python. Indentation should be used for other languages as well to ensure that the program can be followed. Where candidates need to use their own variables, they should be selecting appropriate identifiers that are meaningful to make sure the code can be followed.

General comments

Candidates were able to demonstrate a range of programming skills across the questions.

In Object-oriented programming (OOP), candidates could often declare a class and its constructor. When candidates are required to write a recursive or iterative solution, they need to make sure they are following those requirements and including a recursive call within their function to meet that criteria.

Candidates could often open files and read data from the files but often did not meet all requirements of the question, for example by looping a set number of times instead of taking into account an unknown number of lines in a file.

Comments on specific questions

Question 1

(a) Most candidates were able to accurately declare the variables. Some candidates declared the array but did not initialise each value with the required integer `-1`.

(b) Candidates were often able to declare the function `Enqueue()` and return appropriate Boolean values. Fewer candidates were able to accurately check if the queue was full, by checking an incorrect value for `NumberItems`, or attempting to check each element in the array to see if it contained data.

Some candidates stored the parameter in the appropriate place. Some candidates inaccurately stored the value at the `HeadPointer` position instead of the `TailPointer` position. Some candidates inaccurately incremented both the `HeadPointer` and the `TailPointer`.

Few candidates correctly created a circular queue, with the pointer returning to 0 once it had reached the end of the array.

(c) Most candidates were able to call `Enqueue()` with the correct values. Some candidates made appropriate use of a loop, whilst some manually called `Enqueue()` for each value. A common error was not storing the return value from the function call and therefore this could not be checked to output the appropriate message. Some candidates attempted to manipulate the array direct as, opposed to, or as well as, using `Enqueue()` .

(d) Some candidates implemented the queue as a stack, so in this response attempted to access the data at the tail pointer. Some responses treated the queue as a linear queue and did not loop back to the start of the array after `HeadPointer` was incremented.

Candidates were provided with the variable `NumberItems` to keep track of the quantity of elements in the queue. This was to be used to determine if the queue was full or empty. Few candidates made appropriate use of this, and instead tried to check the values of the pointers to determine if the queue was empty.

(e) (i) Candidates were often able to call `Dequeue()` twice, fewer candidates stored and output the value returned from both function calls.

(ii) Some candidates were able to gain the correct output from their program.

Question 2

(a) Candidates were often able to open the appropriate file. However, fewer candidates also closed this file in an appropriate place.

This question required candidates to read data in from the text file when there was unknown number of elements. This meant the loop could not be a predefined value, some candidates still looped a set number of times. Candidates who implemented a loop dependent on the file input did this by looping until the end of the file, by checking for a null value when reading data, or by reading all the data first and then iterating through it one element at a time. Some candidates also made appropriate use of exception handling.

Some candidates stored the data in the array but did not return the populated array from the function as per the requirements.

(b) This question required candidates to create 6 arrays to store the processed data from the array. Candidates then needed to loop through each of the elements in the array parameter and use an appropriate method to split the data by commas. Some candidates used an inbuilt function, whilst some candidates manually iterated through each character and extracted the data up to the comma character. Fewer candidates accurately compared the second value, i.e. the colour and not number, to each of the colours to store in the appropriate array. Candidates often stored the split data into one array and did not make use of the 6 individual arrays.

(c) Candidates were required to write a program that stored data to an external text file, one whose filename was passed in as a parameter. Some candidates attempted to take the filename as input or extract from the data, some candidates took a parameter but then did not use this write the data to the file. Some candidates opened the file to write but did not close the file.

Candidates then needed to loop through the elements in the array parameter and write the data to the file, making sure there was a new line break between each item. Some responses wrote all the array data to the file without inserting the line break.

This question also required the use of exception handling; some candidates did this appropriately by including all file access code within the try and then having an appropriate output message for when an error did occur.

(d) Candidates often did not attempt this question. Candidates would have been able to gain marks for correct code here even if they had not completed or attempted **Question 2(c)** for demonstrating the skill of calling the procedure correctly.

Some candidates called the procedure with the filenames without identifying them as strings, for example passing Blue.txt instead of 'Blue.txt' as a parameter.

(e) (i) Candidates were often able to call the functions but did not always make appropriate use of the return values and parameters. The function `ReadData()` was to return the populated array, therefore this needed to be stored and used in the main program. This return value was then to be passed to `SplitData()` so that it had the correct values to work with.

(ii) When producing evidence of data stored in a text file candidates need to make sure the screenshot shows the content of the file and the filename of that document. This is to show that the data has been stored in the correct file. Some candidates cropped the screenshot, or it was not clear that it was the data in the text file, for example showing an output from a program.

The question also required a screenshot of the outputs from the program. This needed to show the user entering the filename and then the result of this program.

Question 3

(a) (i) Many candidates were able to declare the class, attributes and the constructor. Fewer candidates assigned null values accurately to `LeftNode` and `RightNode`, instead often storing a parameter. Candidates that are responding in Python need to provide evidence of their data types in their attribute declarations using comments, or other appropriate means, to demonstrate their understanding.

(ii) Most candidates did well at this question, writing accurate get methods and returning the appropriate values.

(iii) Most candidates did well in this question, taking a parameter and assigning it appropriately within the procedure.

(b) Many candidates were able to accurately declare instances of type `Node` and stored the correct data in each node.

(c) (i) Most candidates were able to declare the class and the constructor. Some candidates attempted to pass an integer or a value for a node as a parameter, and then create a new node instead of assigning the parameter.

(ii) Many candidates were able to write a working get method.

(iii) Some candidates found this question challenging. Candidates needed to understand that each `Node` object was stored within the previous `Node` as the left or right node, instead of using pointers an array. Some candidates attempted to implement an array structure instead of a dynamic OOP tree.

Some candidates attempted to compare the nodes, for example if the parameter was less than the head node, instead of accessing the data in the nodes and comparing this. Comparing the nodes would not produce an appropriate result from the comparison.

Some candidates attempted to move left or right by using the correct get methods, but fewer were able to do this within a loop that stopped at an appropriate time such as when the correct position was retrieved.

(d) This question required candidates to write a recursive procedure, some candidates used iteration and did not attempt recursion.

Candidates were often able to compare a node to an appropriate null value and then produced the recursive call with the correct node. Some candidates attempted to compare the data following on from the previous question, instead of following the nodes.

(e) (i) Candidates were often able to create an instance of `Tree` but fewer passed the correct `Node` object to the constructor.

Some candidates were able to accurately call `Insert` with the correct values for the `Tree` object. Some responses called a function `Insert` and not a method related to the class.

(ii) Some candidates were able to gain the correct output for the program.

COMPUTER SCIENCE

Paper 9618/42
Practical

Key messages

Centres should be submitting only one document for each candidate, this is the evidence document where all code needs to be copied and pasted as well as screenshots for testing. Candidates should be submitting black text on white background, without coloured fonts and highlighted sections. The evidence document must contain all work in the correct section for each question being answered, with the centre number and candidate number in header. Screenshots should only be used for the results of testing and these can include white text on a black background where that is standard for the Integrated Development Environment but should avoid all use of coloured font. This is to ensure the answers are legible.

Screenshots must be fully visible and include all inputs and outputs from that test run of the program. Some results may require multiple screenshots depending on the formatting of the outputs.

When copying code into the evidence document, candidates need to make sure the indentation remains when this is part of the syntax of the language, predominantly Python. Indentation should be used for other languages as well to ensure that the program can be followed. Where candidates need to use their own variables, they should be selecting appropriate identifiers that are meaningful to make sure the code can be followed.

General comments

Candidates were able to demonstrate a range of programming skills across the questions.

In Object-oriented programming (OOP) candidates could often declare a class and its constructor. Candidates did find inheritance more challenging and often could not call a parent constructor. Fewer candidates could accurately create an instance of an object, especially when it is from a child class.

Candidates could often open files and read data from the files but often did not meet all requirements of the question, for example by looping a set number of times instead of taking into account an unknown number of lines in a file.

Where a question provides the data, for example '`-1`', candidates must ensure they are using the appropriate data type. '`-1`' is a string data type and not an integer, whilst `-1` is an integer and not a string.

Some candidates were unable to select the appropriate mathematical operators for their language, predominantly for power of and modulus division. Many candidates used the same symbol or function as given in the pseudocode, without converting it to their language.

Comments on specific questions

Question 1

(a) Candidates were often able to store `-1` in the variable `TopOfStack`. Fewer candidates were able to store the string `-1` in all 20 elements in the array, a common error was storing the integer value instead of a string.

(b) Candidates often declared the function accurately taking a parameter. Some candidates correctly identified that the stack is full when the value of `TopOfStack` is 19, a common error was using 20 instead of 19 for the check. Many candidates were able to increment `TopOfStack` and store the parameter in the correct index in the array. Some candidates did not use `TopOfStack` and incorrectly tried to append the data to the array.

(c) Many candidates created the `Pop()` function header but fewer ensured that data was returned from the function, with some candidates having a function with return statements only in specific scenarios.

Some candidates inaccurately determined the `TopOfStack` value as being `0` for empty, instead of `-1` or returned the integer `-1` instead of the string '`-1`'.

Candidates often decremented `TopOfStack` correctly, but some candidates then returned the value from the stack at the decremented position.

(d) Many candidates were able to make appropriate use of exception handling when reading data from a file. Some candidates attempted to perform some of the file handling tasks outside of the exception so they would still run even if the file was unable to be opened. Candidates need to make use of appropriate exception messages so that the reason for the exception is given to the user.

The function needed to open the file with the name passed as a parameter. Candidates often hard-coded the filename into the file opening statement and ignored the parameter.

The function was required to work for a file with an unknown number of lines, this meant that candidates could not use a loop for a set number of lines. Some candidates implemented a set number of iterations, for example 20 and only read in that quantity of lines. The stronger responses looped until the end of file, or read in all the data and had a counter for the quantity of data items that was then used in the loop.

Each line read in the from the file needed to be sent to `Push()` as a parameter. Some responses did not call `Push()` for every line read in, for example one line was read in within the loop condition and then a second line was read in and passed to `Push()`. Some candidates called `Push()` twice with each data value, calling it once in a selection condition to determine the return value and then called it again if that return value was not `-1`.

(e) This question required candidates to use the functions already developed to access the data from the stack appropriately. Some candidates reverted to using the stack as an array without making use of the functions and accessing each element directly.

Candidates were often able to check the multiplication symbol for data held in a variable, but fewer candidates were able to use the appropriate symbol or function for power of in their chosen programming language, for example candidates in Python often attempted to use ^ instead of converting it to **.

The stronger responses made appropriate use of `Pop()` within a loop to access each operator and data item. Some candidates had a correct loop criteria, ending when the string '`-1`' was returned from a `Pop()` call, whilst many candidates attempted to compare it to the integer `-1`.

Some candidates used `Pop()` but did this inappropriately, for example calling `Pop()` in the loop condition and then not being able to make use of this return value again within the loop.

The final value from the calculation needed to be returned from the function. This needed to be after the last value was removed from the stack. Some candidates returned a value at the end of the loop but inside this loop, so the value was returned after the first calculation.

(f) (i) Candidates were often able to take an input and send it to the function `ReadData()`. Candidates often called `Calculate()` but did not output the return value from this function call.

(ii) Some candidates were able to gain the correct outputs from the program. Some of these responses did not show the input of the filename.

Question 2

(a) Some candidates were able to make use of an appropriate data structure for the record format. Candidates who answered in VB.NET often used a structure. Candidates who answered in Python or Java commonly used a class, although dictionaries and 2D arrays were also used appropriately.

(b) (i) Many candidates were able to create an array for `HashTable` and `Spare`. Some candidates only gave a response as a comment which meant there was no actual array created.

(ii) This question required an empty record to be stored in each element in both arrays. An empty record has a key value of `-1` and two data items of `-1`. A common misconception was that `-1` should be stored in each array element instead of a record with `-1` in each field.

(c) Candidates were often able to declare the function with a parameter and return a generated value from this parameter. Fewer candidates were able to calculate the modulus value. Some candidates were able to use the appropriate operator for their chosen programming language, whilst some candidates created a manual calculation for it. Many candidates attempted to use `Mod` as the operator when it was not a valid operator for their language.

Some candidates used a function that returned two values, the integer division and the modulus, but then returned these values as a set instead of just returning the modulus.

(d) Many candidates were able to create the procedure taking a parameter. Fewer candidates extracted the key field from this record and passed it to `CalculateHash()`, many candidates sent the whole parameter. Fewer candidates stored the return value from the function call to make use of this to check if that position was vacant.

Few candidates accurately checked if the position in `HashTable` contained an empty record, with many candidates comparing it to `-1` and not an empty record or an empty key field.

Some candidates made appropriate use of a spare counter that they incremented each time a new record was added to the array `Spare` so that they knew where to store the record instead of having to perform a linear search through the array. A common misconception was that the data should be stored in the calculated index in the second array.

(e) Candidates were often able to open and close the file accurately, some candidates did not close the file in an appropriate place, for example, before the data had been read. Many candidates looped through the file and read in the data; fewer candidates split the data by the commas. Some candidates were able to create a record for the data read in and then often called the function appropriately. Candidates often had exception handling, but some candidates did not have this in an appropriate place, for example, only including a limited amount of file handling so the program would still crash if data could not be read.

(f) (i) Candidates were often able to iterate through each element stored in the array `Spare`, but fewer responses made an appropriate check to determine if an empty record was stored in each index, for example comparing it to an empty record or a key field that stored `-1`. Candidates who had created a counter for the number of elements in `Spare` often iterated through these elements accurately.

(ii) Candidates were often able to call all three functions appropriately.

(iii) Some candidates were able to gain the correct output of the elements in spare. Some responses were cut off and did not include all the values in the output.

Question 3

(a) (i) Many candidates were able to declare the class accurately. Candidates often used the appropriate constructor for their programming language and assigned the appropriate values to the attributes.

(ii) Some candidates sent parameters to the `Description()` function and then attempted to use these to create the messages instead of using the attributes.

A common error was not using an appropriate method to create a string, for example using a comma in Python to separate each element of the message which does not produce a single string.

Candidates often outputted the message generated instead of returning the string.

(b) (i) Some candidates were able to use inheritance appropriately to define the class `Parrot` as inheriting from `Animal`. Candidates often found the constructor more challenging, for example taking only `WingSpan` and `NumberWords` as parameters and not taking the four attributes for `Animal`. Fewer responses called the parent constructor appropriately to assign these values.

This question also required the method `ChangeNumberWords()` to be written, which some candidates did not attempt. Some candidates accurately added the parameter to the attribute. Some candidates who wrote in Python did not include the appropriate self parameter to identify that this was a method within the class and then often did not update the attribute.

(ii) Candidates who wrote in Python often missed the required attribute self for the creation of a method within the class. Candidates in Python often used a comma to separate the elements of the message and did not produce a single string to be returned.

Candidates who wrote in VB.NET needed to implement polymorphism by showing that this method overrides the one from the parent class. Most candidates did this by using overloads or overrides.

(c) (i) Some candidates correctly defined the class with inheritance. Fewer candidates created the appropriate constructor that took all required values. Some candidates who wrote in Python did not include an attribute declaration for `TerritorySize`. Few candidates made appropriate use of the parent constructor.

This question also required the writing of the method `SetTerritorySize()` which was often missing from responses.

(ii) Candidates responding in Python did not always produce methods for `Description()`, instead creating a subroutine. Candidates also often used a comma to join elements instead of creating a single string.

(d) (i) Some candidates were able to create three instances of the classes using the correct values. Candidates need to carefully check the data they are using to make sure it is accurate to the question paper.

Some candidates created three objects from the class `Animal` instead of one from each class.

Some candidates created one or more objects but did not store these within the program, for example in variables.

Candidates should not be using the same identifiers for the variables that are used for the class, for example a class `Parrot` cannot have an instance created and stored in the variable `Parrot`.

(ii) Some candidates were able to call the appropriate methods to change the territory size and number of words for the objects. Some candidates did this manually by changing the attributes for each object.

A common error was calling the methods but not for each of the objects.

Some candidates called the methods to generate the description for each object, but they did not output the returned description.

(ii) Candidates were often able to generate the correct outputs for the program showing the updated values.

COMPUTER SCIENCE

Paper 9618/43
Practical

Key messages

Centres should be submitting only one document for each candidate, this is the evidence document where all code needs to be copied and pasted as well as screenshots for testing. Candidates should be submitting black text on white background, without coloured fonts and highlighted sections. The evidence document must contain all work in the correct section for each question being answered, with the centre number and candidate number in header. Screenshots should only be used for the results of testing and these can include white text on a black background where that is standard for the Integrated Development Environment but should avoid all use of coloured font. This is to ensure the answers are legible.

Screenshots must be fully visible and include all inputs and outputs from that test run of the program. Some results may require multiple screenshots depending on the formatting of the outputs.

When copying code into the evidence document, candidates need to make sure the indentation remains when this is part of the syntax of the language, predominantly Python. Indentation should be used for other languages as well to ensure that the program can be followed. Where candidates need to use their own variables, they should be selecting appropriate identifiers that are meaningful to make sure the code can be followed.

General comments

Candidates were able to demonstrate a range of programming skills across the questions.

In Object-oriented programming (OOP) candidates could often declare a class and its constructor. When candidates are required to write a recursive or iterative solution, they need to make sure they are following those requirements and including a recursive call within their function to meet that criteria.

Candidates could often open files and read data from the files but often did not meet all requirements of the question, for example by looping a set number of times instead of taking into account an unknown number of lines in a file.

Comments on specific questions

Question 1

(a) Many candidates were able to declare the global variables with the correct values. Candidates also often declared the array accurately and assigned integer `-1` to each of the 50 positions. Some candidates did not initialise the values in the array, or initialised an inaccurate quantity.

(b) Candidates were often able to declare the function and take a parameter. Fewer candidates were able to accurately check if the queue was full. The queue was a linear queue, not a circular queue, a common error was treating the queue as a circular queue throughout the question.

Candidates were often able to correctly increment the `TailPointer`. Some candidates did incorrectly increment the head pointer instead, treating the queue as a stack. The tail pointer stored the index of the last element in the queue, therefore this pointer needed to be incremented before the data was stored in the queue. Some candidates did this in the incorrect order.

(c) Some candidates continued to treat the queue as a circular queue instead of a linear queue. This meant that they were not checking if the queue was empty using the appropriate values. Some candidates identified one condition for checking if it was empty, commonly `HeadPointer = -1` which catches the initial state of the queue, but did not include a condition for if some data had been inserted into the queue and then removed, making the condition incomplete.

Candidates were often able to access and return the appropriate value from the queue, although some candidates incremented the head pointer before accessing the element.

When a function returns a value, no code after that statement will execute, some candidates included additional parts of their answer after a return statement, for example incrementing the head pointer after returning the value.

(d) This question required candidates to read each line from a text file and insert in into the queue. Many candidates were able to open the text file; fewer candidates also closed the text file in an appropriate place. Candidates often looped a set number of times which was acceptable, some candidates looped until the end of the file, or read in all the data and then iterated through this data.

Candidates were also required to use exception handling. Candidates were often able to do this accurately, with all file access statements within the try and an appropriate output message for if an error did occur. Some candidates had the opening and reading from the file within the try but then closed the text file outside of the exception.

Candidates were often able to call `Enqueue()` with each value read in from the file. However, some candidates called `Enqueue()` multiple times with each value, for example, calling it once and then calling it again within a selection statement and sometimes a third time if the selection statement condition was true.

(e) (i) Candidates were often able to call `CreateQueue()` accurately. Fewer candidates were able to call `Dequeue()` repeatedly until the queue was empty; this required an understanding that `-1` would be returned from `Dequeue()` when the queue was empty. Some candidates attempted to manually check the pointer values to determine if the queue was empty. Candidates were often able to output the total generated.

(ii) Some candidates were able to generate the correct output from the program.

Question 2

(a) Many candidates were able to accurate create the array and assign the correct values.

(b) Some candidates found this question challenging. Candidates often wrote a bubble sort algorithm instead of an insertion sort algorithm, or mixed the two sorts. A common error was not including a stopping condition that checked when the value being inserted was in the correct position, and instead continuing to compare elements to the start of the array.

The function was required to return the sorted array. Some candidates did not return the array.

(c) Candidates were often able to create the procedure and output each element in the array, some candidates did not output it in the correct format, for example they output each element on a new line, or output the array as a whole without any separation.

(d) (i) Some candidates did not make appropriate use of the return values and parameters in this question, for example `OutputArray()` takes an array as a parameter and then `InsertionSort()` returned an array which then had to be used in the next call of `OutputArray()`.

(ii) Some candidates were able to get the correct output for this program. A common error was missing a number on the output of the sorted array.

(e) There were a range of approaches to this question, with some candidates writing iterative solutions and some writing recursive solutions. Candidates often calculated the mid point appropriately and then accessed the element at this position. Some candidates did not update the mid value appropriately if the element was not found, for example updating lower to be mid, instead of mid+1. Some candidates had inaccurate looping conditions, for example they did not loop until low was less than or equal to high.

Some candidates did not attempt a binary search and instead write a linear search.

(f) (i) This question required candidates to call the search function four times with different set values. Some candidates took values as input and then attempted to search for these instead of the hard-coding the specific values they were asked to search for. Candidates often output a message if found and if not found, but not all candidates included the index of the data when it was found.

(ii) Some candidates were able to gain the correct output for the program.

Question 3

(a) (i) Many candidates were able to declare the class, the constructor and assign `TheData` to the parameter. Fewer candidates were able to assign an appropriate null value to `NextNode`.

(ii) Most candidates were able to write accurate get methods.

(iii) Most candidates were able to write an appropriate set method. Some candidates attempted to reuse the `Node` class identifier as a parameter which was not appropriate.

(b) (i) Many candidates were able to declare the linked list class appropriately with the constructor. Fewer candidates assigned an appropriate null value to `HeadNode` in the constructor.

(ii) Candidates were often able to define the method `InsertNode()`. Some candidates answering in Python created a procedure and not a method by missing the required self (or equivalent) as a parameter and then use of the correct attributes.

Some candidates were able to create an instance of `Node` with the parameter value. A common error was treating the parameter as a `Node` object instead of an integer value.

In this list, each new node is inserted at the head of the linked list. This means that it replaces the `HeadNode`. Some candidates attempted to traverse the linked list to find the final node and then insert the `Node` as the last node in the list.

(iii) The `Traverse()` method needed to iterate through each `Node` object and store the data in one string. This required candidates to start at the `HeadNode` and then access the `NextNode` repeatedly until it is null. A variable was required to store the data from each node and finally return this outside of the loop. Some candidates did not iterate through each element, instead accessing the `HeadNode` and just returning this data. Some candidates did not attempt this question.

(iv) Candidates often found this question challenging. Candidates were required to traverse the linked list starting at the `HeadNode` until the data was found or until the end of the list was reached. Some candidates attempted to compare `Node` objects with the integer data they were looking for, instead of accessing the data value within each node.

Some candidates produced effective solutions where they made appropriate use of the get and set methods to access the update the data and nodes.

Some candidates did not attempt this question.

(c) (i) Candidates were often able to create an instance of the linked list. Fewer candidates were able to use the appropriate method to insert the required nodes or call the function to traverse the tree and remove the correct node.

(ii) It was pleasing that candidates were able to identify the need to amend their program to generate an output from the program. Candidates were required to analyse the question and determine that additional code was required to produce the output. Many candidates were able to perform these actions and produce the correct output.