COMPUTER SCIENCE

Paper 9618/11
Theory Fundamentals 11

Key messages

There is a list of the command words used in examination questions given on page 41 of the specification. Candidates should be encouraged to look carefully at these command words and understand what each one needs as an answer. For example, a question that begins with the command word 'explain' requires a different kind of response to one that starts with the command word 'state'.

There are other key words that are also regularly used in these examination papers and candidates should familiarise themselves with them. Words such as 'benefits' and 'drawbacks' are examples.

Candidates must think carefully about their answers before beginning to write. Statements that repeat or use the name in a description do not demonstrate any further understanding. For example, '*a monitoring system monitors...* ' or '*a general purpose register is for general purpose*s'.

Candidates should be aware of the need to use terminology correctly. There is often confusion between terms such as *database* and *table* when writing about DBMS systems and between *memory* and *storage* when discussing processing or data transmission.

General comments

Candidates should take care with their handwriting. If an answer cannot be read, then it cannot be given any marks. Multiple crossing out and writing between the lines should also be avoided wherever possible. If candidates write a response in the answer space on the question paper and later decide that the answer is incorrect and needs to be replaced, the new version should be written either on any blank pages in the script or on an additional sheet of paper, rather than trying to squash the new answer into the existing answer space.

When an answer is replaced or is continued elsewhere a note for the examiner such as, '*Please see the bottom of page 10*', is very helpful.

Candidates should avoid writing in the margins of the page as these can sometimes be removed during the scanning process.

Comments on specific questions

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

Question 1

(a) There were many good answers to this question. Some candidates need to take more care. The incorrect statement that a tebibyte was equal to 1024 gigabytes (not gibibytes) appeared frequently and units were often omitted altogether. Some candidates should understand that imprecise statements such as, '*a tebibyte is bigger than a gigabyte*' is not enough for credit at this level of study.

(b) (i) This question was answered very well. Many candidates were able to correctly convert the unsigned binary number to hexadecimal.

(ii) The conversion from two's complement binary to denary was less well done. A common error was a missing minus sign where candidates had correctly flipped the bits and added one to get the positive binary value, then correctly converted this positive value to denary, but had forgotten to replace the minus sign to accurately represent the original negative number.

(iii) This question was also answered very well. Many candidates were able to correctly convert the Binary Coded Decimal to denary. Some candidates need to improve their understanding of the difference between BCD and unsigned binary numbers. A common incorrect answer was 1395 where the conversion was from unsigned binary into denary.

(c) Many candidates found this binary subtraction challenging. The question asked candidates to subtract the denary number 23 from the two's complement binary number. Any appropriate method of working is accepted; however, the method must be carried out correctly. Frequent errors in the working included subtracting the two's complement binary value given from 23 instead of the other way round and converting all the values to denary, completing the subtraction in denary and then converting the answer back to binary. Some candidates should understand that when a question asks for a calculation to be completed in binary no marks will be awarded for a denary calculation.

(d) This question was answered well. Many candidates were able to give a correct reason why binary addition and subtraction can result in overflow. Some candidates should be aware of the need to refer to the number of bits available in their answer. Vague statements such as, '*too big to be stored'* are insufficient, a better response is '*too big to be stored in the number of bits available'*.

Question 2

(a) Almost all candidates were able to give a correct limitation of using a file-based approach to store data, usually data redundancy or data inconsistency. Explaining how this was overcome in a relational database proved to be much more challenging. Many responses simply described the limitation rather than saying how it was overcome, which did not answer the question. There was also considerable confusion with the correct use of the terms '*database*' and '*table*', some candidates writing about, for example, the foreign key of one database linking to the primary key of another database.

(b) (i) There were some good, complete diagrams. Many candidates correctly completed the relationship between the CUSTOMER and REPAIR entities. Quite a few candidates also recognised that the REPAIR_PART table was a linking table designed to implement a many-to-many relationship between the PART table and the REPAIR table, but then incorrectly indicated the degree of the two one-to-many relationships.

(ii) Again, there were a few completely correct SQL scripts. A common error was the omission of any constraints. Some candidates need to improve their understanding of the statement to define a composite primary key.

(iii) There were many good SQL scripts written. The most common cause of error was the use of the COUNT function instead of the SUM function to find the total amount due. Some candidates need to take care when copying attribute names. Amount occurred frequently instead of AmountDue.

(c) Definitions of the terms Candidate Key and Tuple were mostly correct. Here too, some candidates need to take care with the terminology. Statements such as, *'a tuple is a row in a database'* were frequently seen rather than *'a tuple is a row in a table'*. Giving a correct definition for Referential Integrity proved to be more challenging. Vague statements such as, '*referential integrity ensures the integrity of the data*' are not enough for credit at this level.

Question 3

(a) Answers for the system clock were better than answers for the control unit. Here again, some candidates should understand that this is a technical subject and at this level some technical knowledge and understanding is expected. Answers such as, '*the control unit controls the other components*' which simply expand the name do not demonstrate any technical expertise. Something like, '*the control unit manages communication between the components in the CPU*' shows a better understanding.

(b) (i) Most candidates were able to identify a feature that would affect the performance of a computer system. Clock speed and bus width were easily the most popular correct answers. Some candidates need to take care that their reasons clearly state why the performance is affected and do not just describe the feature chosen.

(ii) Almost all candidates correctly identified a USB port as a suitable connection for the solid state memory drive. Many candidates need to improve their understanding of the way in which the port provides an automatic connection. It is not enough at this level to simply say that the device is 'plug and play' or to give a description of a USB connector. These do not answer the question.

(c) This question was answered very well. Easily the most popular correct answers referred to the need for constant refreshing and lower access speed.

(d) (i) There were a small number of good answers to this question. Many candidates understood that reading and writing to an optical disc required the use of a laser to create pits and lands. Quite a few candidates found it more challenging to say anything else and need to improve their understanding of the principal operation of an optical disc reader/writer. A number of candidates described the operation of a magnetic disk reader/writer.

(ii) This question was not answered well. Many candidates need to improve their understanding of buffers. A buffer is an area in memory or storage and therefore will store or hold data, but it cannot send or transmit anything. Using phrases like '*the buffer sends*' when describing the use of a buffer demonstrates a lack of understanding of what a buffer is. Very few responses made any reference to a speed mismatch between the computer and the optical disc reader/writer.

Question 4

(a) (i) This question was not answered well either. There was considerable confusion between file management and memory management. Many responses described memory management rather than file management. Here too there is a need for correct use of terminology. Some of the responses that attempted to describe file management tasks used the word 'memory' when referring to secondary storage.

(ii) Again, this question was not answered well. The question did not ask for a description of back-up software, it asked why back-up software was needed. Very few responses gave any reasons for using back-up software.

(b) (i) This question was answered very well. Many candidates were able to give correct benefits to the teacher of the file being compressed. Some candidates need to read the question carefully, it asked for benefits to the teacher, not benefits for the candidate.

(ii) This question was answered very well. Easily the most popular choice of method was Run-Length Encoding. Some candidates need to take care that when expanding the acronym to ensure that it is done correctly. The question asked about compressing a text file, many of the responses described compressing an image file and wrote about consecutive pixels of the same colour rather than repeated strings of the same character.

(c) This question was not answered well. The question asked for the benefits to the candidate of using a program library. Many responses did not refer to the candidate at all and so did not answer the question. When questions are asked in a particular context it is a test of understanding and generic responses are not enough.

(d) (i) Many candidates were able to correctly identify a presentation feature found in a typical IDE. Some candidates need to read the question carefully to ensure that some features are not excluded. In this question candidates are told that prettyprint is a presentation feature, so no marks will be awarded if a candidate repeats that feature.

(ii) Again, many candidates were able to correctly identify another debugging feature found in a typical IDE. Some candidates need to understand that if single stepping is given in the question marks will not be awarded for answers such as double stepping or two stepping.

Question 5

(a) There were some good clear answers to this question, with three appropriate sensors identified.

(b) There were some good answers to this question. The expression for X was often correct, some candidates found it more challenging to determine the expression for Y.

(c) There were many excellent explanations for the system outlined. Some candidates need to be aware that vague answers such as '*it is a monitoring system because it monitors the conditions*' are not enough for an explanation at this level of study.

Question 6

Many candidates found this question very challenging. There were a few correct statements about image recognition or optical character recognition, but many of the responses did not describe the use of artificial intelligence in the identification of the vehicle registration numbers at all, rather they described the use of a DBMS system to manage the registration numbers of the cars entering and leaving the car park.

Question 7

(a) This question asked for the benefits of distributing software using a shareware licence. Many candidates need to take care when reading the question. Generally, responses gave two of the characteristics of a shareware licence, but did not state why this was a benefit to either the distributer or the user.

(b) Similarly, this question asked for the benefits of distributing software using a commercial licence, but responses again usually gave characteristics rather that stating why this was of benefit to either the distributer or the user.

Question 8

(a) Many candidates found describing the purpose of the status register very challenging. Some candidates were able to give an example of a status flag, however, very few responses described the use of bits that could be individually referenced.

(b) This question too, was challenging for many. It is an area where candidates need to improve their understanding. Vague statements such as, '*general purpose registers are used for general purposes and special purpose registers are used for special purposes*' which do little more than repeat the names are not enough as an answer.

COMPUTER SCIENCE

Paper 9618/12
Theory Fundamentals 12

Key messages

There is a list of the command words used in examination questions given on page 41 of the specification. Candidates should be encouraged to look carefully at these command words and understand what each one needs as an answer. For example, a question that begins with the command word 'explain' requires a different kind of response to one that starts with the command word 'state'.

There are other key words that are also regularly used in these examination papers and candidates should familiarise themselves with them. Words such as 'benefits' and 'drawbacks' are examples.

Candidates must think carefully about their answers before beginning to write. Statements that repeat or use the name in a description do not demonstrate any further understanding. For example, '*a monitoring system monitors...* ' or '*a general purpose register is for general purpose*s'.

Candidates should be aware of the need to use terminology correctly. There is often confusion between terms such as *database* and *table* when writing about DBMS systems and between *memory* and *storage* when discussing processing or data transmission.

General comments

Candidates should take care with their handwriting. If an answer cannot be read, then it cannot be given any marks. Multiple crossing out and writing between the lines should also be avoided wherever possible. If candidates write a response in the answer space on the question paper and later decide that the answer is incorrect and needs to be replaced, the new version should be written either on any blank pages in the script or on an additional sheet of paper, rather than trying to squash the new answer into the existing answer space.

When an answer is replaced or is continued elsewhere a note for the Examiner such as, '*Please see the bottom of page 10*', is very helpful.

Candidates should avoid writing in the margins of the page as these can sometimes be removed during the scanning process.

Comments on specific questions

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

Question 1

(a) This question was generally answered well. The most common reason for incorrect answers was the misinterpretation of the positioning of the brackets in the expression given.

(b) This question was also answered well. Some candidates need to take more care when drawing the symbols for the logic gates. It is sometimes very difficult to distinguish between the symbols for an AND gate and an OR gate. Candidates are advised to use the symbols given in *section 3.2* of the specification.

Question 2

(a) The two key words in this question were 'describe' and 'drawbacks'. Some candidates overlooked these words and simply stated the characteristics of embedded systems. For example, a response such as, '*an embedded system can only perform one task*' is a statement of a characteristic not a description of a drawback. A better answer would be, '*an embedded system can only perform one task, therefore they cannot easily be adapted for another function*'.

(b) Similarly, some candidates found this question challenging and simply stated characteristics of DRAM without considering whether it was a benefit or not. Care should be taken with statements referring to DRAM having a higher storage capacity. Such statements need to be explicit that this is per chip, not in general.

(c) There were many good answers to this question. Easily the most popular differences were the methods used to erase data and whether the chip needed to be removed or not. Some candidates should understand that generalised answers such as, '*EPROM is cheaper than EEPROM*' will not be given credit at this level of study. There also seems to be a misconception amongst some candidates that EPROM can only be written once

Question 3

(a) This question was not answered well. Candidates need to improve their understanding of registers. A register is a memory location and therefore will store or hold, for example, an address or an instruction, but it cannot send or transmit anything. Using words like 'send' when describing the purpose of a register demonstrates a lack of understanding of what a register is. A common incorrect description for the Program Counter suggested that it held the number of instructions that had been executed, rather than the address of the next instruction to be executed. Answers such as, '*the Index Register is used in indexed addressing*' are also insufficient for credit at this level of study.

(b) There were some excellent, complete answers to this question. Many candidates were able to explain that using HDMI enabled the transmission of audio as well as video with a single connection. Some candidates found explaining further benefits of HDMI more challenging and included generalised statements about HDMI being a more up-to-date connection and modern equipment not having a VGA port when the question stated that the computer had both. Here again, the command word in the question is 'explain', so more than simple statements of fact is required.

(c) Again, there were a few very good answers which described the process management tasks in considerable detail. However, many candidates found this question challenging. A popular correct answer described deciding which process received processor time next. There was some confusion between resource allocation as part of process management and the role of memory management. Care should be taken to distinguish between these two operating system management tasks

Question 4

(a) Almost all candidates were able to name two debugging features of an IDE, usually single-stepping and breakpoints. Many candidates found it more challenging to describe each feature. Some candidates should understand that '*breaks the program at a given point*' which simply expands the name of the feature is insufficient for a description of a breakpoint.

(b) This question was not answered well. The question asked for the benefits to the programmers of creating a program library. Many responses did not refer to the team of programmers at all and so did not answer the question. When questions are asked in a particular context it is a test of understanding and generic responses are not enough.

(c) This question was answered well. Encryption was by far the most popular correct answer, and there were many complete explanations. Some candidates need to take care to ensure that they differentiate between being able to intercept the code and the code being understood if it is intercepted. Encryption does not prevent interception, it prevents understanding.

Question 5

(a) There were some excellent, thoughtful answers about why a programmer should act ethically towards their colleagues. Some candidates found it more challenging to give reasons for acting ethically towards the public. Reasons for the public tended to be more relevant to the company or the product rather than the public.

(b) (i) Again there were some very good answers, but there was also considerable confusion between the Free Software Foundation and Freeware as a type of licence. Some candidates need to improve their understanding of the different types of software licence available.

(ii) Many candidates were able to correctly say that copyright gave the author of the software recognition of ownership. Some candidates should understand that copyright does not actually prevent plagiarism, but it makes the act illegal and allows redress for the copyright holder.

Question 6

(a) Almost all candidates were able to correctly identify at least one benefit of a relational database. However, this was a question where candidates needed to ensure that they used the appropriate terminology. There were many instances where the word '*database*' had been used when it should have been '*table*'.

(b) (i) Some candidates found this question very challenging. Some of the better responses were from candidates who had drawn an E-R diagram to represent the tables given in the question and then based their answer on the diagram. Quite a few candidates recognised that the JOB_EMPLOYEE table was a linking table designed to implement a many-to-many relationship between the JOB table and the EMPLOYEE table, but then incorrectly gave the degree of the two one-to-many relationships. There was, too, a general assumption that the primary key of each table would be the table name followed by ID, for example, CustomerID and JobID, but very few candidates actually stated this.

(ii) There were many good SQL scripts written. The most common cause of error was the use of the COUNT function instead of the SUM function to find the total amount. Some candidates need to take care when copying attribute names. DataSent occurred frequently instead of DateSent.

Question 7

(a) (i) This question was answered very well. Many candidates were able to correctly define the two terms.

(ii) There were many good answers to this question, the effects of changing the image resolution were explained quite well. Some candidates should be aware of the need to make it clear whether the image resolution is increasing or decreasing. Statements such as, '*changing the image resolution will change the image quality and file size*' are far too simplistic and will not be given credit at this level. A small number of candidates confused image resolution and sampling resolution.

(iii) This question was also answered very well. Easily the most popular choice was Run-Length Encoding. Some candidates need to take care that when expanding the acronym to ensure that it is done correctly.

(b) Some candidates found this question challenging. The properties were not described well. Many candidates gave correct examples, but on their own this is insufficient for a definition. Many of the definitions of a drawing list actually described a library of shapes, that is, all the shapes available, rather than a list of the subset of shapes from the library used in a particular image.

Question 8

(a) There were many correct answers to this question. The second pair of instructions was the one most likely to have an incorrect value of 34 in the ACC where candidates had applied the instruction ADD #19 rather than the one given. Some candidates did not complete the column of IX values.

(b) (i) Some candidates found this question more challenging. A frequent error in the third pair of instructions was the omission of the INC ACC after a correct OR operation.

(ii) Many candidates understood that an AND operation would be required and gave a correct instruction. Explaining the process proved to be more challenging. Some candidates should understand that statements such as, '*if the number is odd the end bit will be 1*' are not enough. It does not explain which 'end' bit. The use of the correct wording, such as, '*if the number is odd the least significant bit will be 1*' makes it clear.

Question 9

(a) There were some good clear answers to this question, with appropriate sensors identified and their use in the system clearly described. Some candidates need to take care that they understand that the sensors do not take any action. The sensors measure and send the data to the microprocessor that instigates any action.

(b) There were many excellent explanations for the system outlined. Some candidates need to be aware that vague answers such as '*it is a monitoring system because it monitors the traffic approaching the bridge*' are not enough for an explanation at this level of study.

COMPUTER SCIENCE

Paper 9618/13
Theory Fundamentals 13

Key messages

There is a list of the command words used in examination questions given on page 41 of the specification. Candidates should be encouraged to look carefully at these command words and understand what each one needs as an answer. For example, a question that begins with the command word 'explain' requires a different kind of response to one that starts with the command word 'state'.

There are other key words that are also regularly used in these examination papers and candidates should familiarise themselves with them. Words such as 'benefits' and 'drawbacks' are examples.

Candidates must think carefully about their answers before beginning to write. Statements that repeat or use the name in a description do not demonstrate any further understanding. For example, '*a monitoring system monitors...* ' or '*a general purpose register is for general purpose*s'.

Candidates should be aware of the need to use terminology correctly. There is often confusion between terms such as *database* and *table* when writing about DBMS systems and between *memory* and *storage* when discussing processing or data transmission.

General comments

Candidates should take care with their handwriting. If an answer cannot be read, then it cannot be given any marks. Multiple crossing out and writing between the lines should also be avoided wherever possible. If candidates write a response in the answer space on the question paper and later decide that the answer is incorrect and needs to be replaced, the new version should be written either on any blank pages in the script or on an additional sheet of paper, rather than trying to squash the new answer into the existing answer space.

When an answer is replaced or is continued elsewhere a note for the Examiner such as, '*Please see the bottom of page 10*', is very helpful.

Candidates should avoid writing in the margins of the page as these can sometimes be removed during the scanning process.

Comments on specific questions

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

Question 1

(a) This question was answered well. Some candidates need to take care to ensure that their answer covers all possible combinations of inputs.

(b) This question was also answered well. There were a small number of responses where the NOR gate had been confused with an XOR gate.

(c) This question was also answered well. Some candidates need to take more care when drawing the symbols for the logic gates. It is sometimes very difficult to distinguish between the symbols for an AND gate and an OR gate. Candidates are advised to use the symbols given in *section 3.2* of the specification.

Question 2

(a) This question was generally answered very well. Easily the most popular correct answers referred to DRAM needing constant refreshing and having lower access speed.

(b) This question was not answered well. Many candidates need to improve their understanding of buffers. A buffer is an area in memory or storage and therefore will store or hold data, but it cannot send or transmit anything. Using phrases like '*the buffer sends'* when describing the use of a buffer demonstrates a lack of understanding of what a buffer is. Very few responses made any reference to a speed mismatch between the computer and the magnetic disc drive.

Question 3

There were some excellent explanations for the system outlined. Some candidates need to be aware that vague answers such as '*it is a control system because it controls the speed of the car*' are not enough for an explanation at this level of study. Questions like this which are asked in a specific context are designed to test the candidates' understanding and so it is expected that responses refer to the scenario given and not simply repeat generic information or information given in the question.

Question 4

(a) Almost all candidates were able to correctly identify the foreign keys and the correct tables.

(b) There were many good SQL scripts written. The most common cause of error was the use of the COUNT function instead of the SUM function to find the total amount due.

(c) (i) Again, there were a few completely correct SQL scripts. Common errors were the omission of any constraints and using an integer data type for the Size and/or SellingPrice when the data given is clearly not a whole number.

(ii) Many candidates found this question challenging. Quite a few candidates identified the repeating attributes but linking the resulting tables was rarely carried out correctly. Care must be taken to read the question fully, some candidates with tables containing appropriate fields ignored the instruction in the question to identify any primary or foreign keys

(d) Some candidates were able to correctly define the two terms, the definition for an attribute was more likely to be correct than the definition of an entity. Many candidates need to improve their understanding of the terminology associated with databases.

(e) This question was not answered at all well. The question asked specifically about data integrity in a DBMS. Many responses simply listed the generic advantages of a DBMS over a file-based system, others included a lot of irrelevant information about validation rules.

Question 5

(a) This question was answered well. Many candidates were able to correctly identify the three types of software licence.

(b) There were some excellent, thoughtful answers about why a programmer should join a professional ethical body. The most popular reasons were having guidelines to follow and the availability of training courses.

Question 6

(a) Almost all candidates were able to correctly identify a character set and there were many appropriate descriptions. Some candidates need to be careful with the size measurements. Statements such as, '*ASCII has 7 bytes per character*' instead of 7 bits per character were not uncommon.

(b) (i) There were many correct answers to this question, the most common incorrect answer was 255.

(ii) Almost all candidates were able to say that a drawback was increased file size.

(iii) There were some good, complete answers to this question, many candidates were able to explain why lossy compression is suitable for bitmap images. There was some confusion between the types of compression and some candidates tried to explain why lossless compression would be suitable.

(c) (i) Many candidates found this question challenging and there were many descriptions of microphones and mention of analogue to digital convertors. There is a need for candidates to improve their understanding of how analogue data is converted to binary format.

(ii) Removing sounds outside the range of human hearing was a popular correct method of compression, however, the question asked for a description and many candidates found it more challenging to expand on their choice of method to achieve the second mark. Again, there was some confusion between the types of compression and some candidates incorrectly tried to describe Run-Length Encoding.

Question 7

(a) There were many correct answers to this question. The second pair of instructions was the one most likely to have an incorrect value of 105 in the ACC where candidates had applied the instruction ADD #56 instead of the instruction given.

(b) Many candidates understood that an AND operation would be required and gave a correct instruction. Explaining the process proved to be more challenging. Some candidates should understand that statements such as, '*to clear the register*' are not enough, this repeats the wording of the question, it does not explain what clearing the register means. The use of the wording, such as, '*if the register is cleared all the bits will be 0*' makes it much clearer.

Question 8

(a) Almost all candidates were able to convert the hexadecimal number into denary. Some candidates need to take care with the addition of the place values, there were some unnecessary errors.

(b) The conversion from two's complement binary to denary was less well done. A common error was a missing minus sign where candidates had correctly flipped the bits and added one to get the positive binary value, then correctly converted this positive value to denary, but had forgotten to replace the minus sign to accurately represent the original negative number. The question asked for an explanation, but many responses showed just the working with no attempt at any explanation.

(c) Descriptions of a right logical shift were usually correct. Many candidates need to improve their understanding of arithmetic shifts. There was considerable confusion with cyclic shifts.

Question 9

(a) Extending over a large geographical area was a popular choice of characteristic of a WAN. Many candidates found it more challenging to describe a second characteristic in order to achieve the second mark.

(b) There were some good answers to this question with suitable transmission media identified and described. Some candidates should understand that twisted pair cable, for example, is a variety of copper cable and so is excluded from acceptable answers because it is given as a transmission medium in the question.

(c) This question was not answered well. Many candidates need to improve their understanding of how bit streaming is used in real-time communications.

(d) (i) Answers for the public IP address were more likely to be correct than answers for the private IP address. Some candidates should understand that using the words 'public' and 'private' in describing the purpose of the two types of IP addresses is not appropriate. Statements such as, '*a private IP address keeps the address private*' does not say anything extra about the address and so does not answer the question.

(ii) This question was answered well. Many candidates gave appropriate answers in terms of security and the reduction of network traffic.

COMPUTER SCIENCE

Paper 9618/21

Fundamental Problem-solving and Programming Skills

Key messages

The emphasis for this paper is on the application of practical skills. Candidates need to have developed these and be able to apply them to the scenarios presented.

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 need to improve their competence in terms of designing and implementing a solution to a problem.

This is a technical subject and makes use of many technical words and phrases. These have specific, defined meanings and it is important that these are used correctly.

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

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

General comments

Candidates need to read each question carefully and it is important to clearly understand the question before attempting to answer it. 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.

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 would not be credit worthy.

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

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

If typed answers are provided, then it is very helpful if these are organised so that individual answers do not span page breaks. This is particularly important for programming answers. If the question involves completing a table, then typed answers should clearly indicate any unfilled rows.

Comments on specific questions

Question 1

(a) Few candidates gave the correct answer for this initial question.

Candidates need to improve their understanding of the `CASE` construct as many referred to a syntax error in their answer. A common example of this lack of understanding was 'if `ItemCost` is

less than 50 then it is also less than 200' or the `OTHERWISE` command word was missing before the '`HigerRate ← TRUE`' statement.

(b) (i) Around half of candidates gained the mark for mentioning a constant.

(ii) Many accessible mark points meaning many candidates gained at least one of the marks points available.

MP3 and MP4 'easier to…' answers were the most common.

(c) Full marks for vast majority of candidates.

The most common wrong answer was to suggest Integer for row two.

(d) More than half of candidates gained the mark for this.

A common mistake was to suggest `ELSE` as the answer.

Question 2

The majority of appropriate solutions approached the problem in a logical sequence following the guidance given in the question stem: increment `SS`, check for 60, if so, increment `MM` etc.

Weaker solutions contained loops, often along the lines of `FOR SS ← 0 TO 59` or initialised the global variables.

Incomplete selection clauses (missing `ENDIF`) prevented a mark in many solutions.

Mark point breakdown:

MP1: Given to many solutions.

MP2: Calls to `CheckAlarm()` were seen regularly, and often correctly formed since this was a procedure rather than a function and had no parameters. The requirement to call the procedure when `SS` was set to zero was missed by many

MP3, MP4 and MP5: were seen regularly, these were usually given as a block, and it was not uncommon for stronger solutions to have gained MP1 to MP5 within the first 5 or 6 lines of pseudocode.

MP6 and MP7: very often missed both parts of the solution to gain the mark.

Common mistakes included:

- Declaring `HH`, `MM` and `SS` as local variables.
- Incorrect use of Boolean operator: `IF HH AND MM AND SS = 0 THEN.`
- Using different variable rather than `HH`, `MM` and `SS` (and not updating the originals before the end).
- Incorrect comparison: `IF MM = MM + 1 THEN.`
- Use of iteration rather than selection. e.g. `WHILE HH = 0 AND SS = 0`...
- Incorrect conditional nesting e.g. so that `CheckAlarm()` would only be called at the start of each hour when `MM` was set to zero.

Question 3

(a) A small number of candidates completed the steps correctly, with most candidates gaining some marks.

Step 1: The file mode was very often omitted making this the rarest mark.

Step 2: Commonly given. Mistakes included testing for `ThisLine` being an empty string or looping until '3 lines have been read'.

Step 3: Weaker answers were seen omitting any reference to 'the line'. Many answers suggested multiple lines being read at once.

Step 7: Commonly given. Mistakes included such statements as 'end the loop and count the lines'.

(b) Gained by just under a third of the candidates.

Although many answers appeared to be on the right track, the mark scheme required a level of precision that eluded many. References to 'sorting' were seen only occasionally.

Many weaker answers such as 'to initialise the variables' or 'to make sure all the variables are used'. Often these attempted to describe 'what' the lines were doing rather than 'why'.

(c) Many candidates found this a challenging question.

Mark point breakdown:

MP1: When a second mark was given (after MP6) this was usually the mark point given.

MP2: Description was often too vague for this to be given (e.g. 'if more lines are not available') and often it was not clear that this test formed part of a loop condition.

MP3: Solutions often included three sequential read operations, ignoring any possible `EOF()` implications.

MP4: Rarely given. If a `READ` operation was described, then the immediate `OUTPUT` was very often omitted.

MP5: As for MP2.

MP6: The most given mark point.

MP7: Seen only on a few occasions.

Common mistakes:

- Assuming it was possible to jump immediately to the required start line in the file.
- Omitting any explicit description of reading the file.
- Checking for blank lines and treating them differently from 'normal' lines.
- Descriptions that did not 'flow' in any meaningful way.

Question 4

(a) Only a small number of two marks answers was seen with around 50 per cent of candidates gaining one mark.

Many candidates appeared not to have understood the scenario so 'syntax error' was seen often.

The most common correct 'cause' mark was given for 'divide by zero'. For the alternative, many answers were heading in the right direction but lacked precision to clearly identify a non-numeric string.

(b) Around 30 per cent of them achieved at least one mark.

'Program easier to understand' and 'no need to re-write the code many times' were perhaps the two most common marks given.

A wide range of incorrect answers were seen including 'checks can occur automatically', 'avoids program crashing', 'easier to read' and 'saves time'.

(c) A wide range of response was seen with the full range of marks being awarded.

Three marks for 'housekeeping' aspects of the solution (MP1, MP2 and MP3).

Weaker candidates struggled to use the user-defined data type, with consequential impact on MP2, MP3, MP7 and MP8.

Mark point breakdown:

MP1: Candidates are improving at translating a scenario into a module header, and in this case MP1 was essentially simply the pseudocode equivalent if the first bullet point from the question and was given to most meaningful solutions.

MP2: As for MP1, although a number of solutions attempted to return both a `REAL` and a `BOOLEAN` rather than a user-defined variable of type `Result`.

MP3: The combination of many candidates' failure to declare local variables together with the complication surrounding the use of a record type put this mark out of the reach of many.

MP4: The function `ISNUM()` was absent from many solutions and was not as common as `STR_TO_NUM()`.

MP5: Function `STR_TO_NUM()` was seen frequently and was often correctly used.

MP6; Several solutions incorrectly used the expression `IF STR_TO_NUM(NumStr1) > 0`, for this test meaning that negative numbers would have been regarded as invalid.

MP7 and MP8: Two challenging marks that relied on correct field assignment as well as field initialisation and correct logic. Not often given. Attempts to return two separate values in place of a single item of type `Result` were seen often, with `OUTPUT` also being used in some solutions.

Common mistakes:

- A common error was to identify the parameters as type Integer.
- Weaker solutions often included `INPUT` statements despite having defined the parameters in the function header.
- Comparing the parameters with integer `0` rather than character `'0'`.
- Assigning the result of the calculation to the record type itself:

```
Result ← STR_TO_NUM(NumStr1)/STR_TO_NUM(NumStr2)
```

- Attempting to return two separate objects:

```
RETURN Value, Done
```

- Invalid Boolean construct:

```
IF STR_TO_NUM(NumStr1) AND STR_TO_NUM(NumStr2) = REAL THEN
```

Question 5

(a) Many full-mark answers seen. A wide range of marks was given for this question.

No discernible pattern seen for wrong answers given.

(b) (i) Many full-mark answers seen with candidates recognising this style of question and understand what is required to 'thoroughly test' an algorithm of this type.

A fairly common mistake was to confuse 'Abnormal' with 'Extreme'.

A small number of candidates lost out by simply giving different values from the 'Normal' range.

Some incorrect test types were suggested.

(ii) Around 40 per cent of candidates gained the mark for this question.

Question 6

(a) An understanding of the scenario was required and generalised answer relating to arrays were not always credit worthy. The mark scheme demanded greater precision for these marks.

Less than 20 per cent of candidates gained the full two marks for this question with many more gaining one mark.

MP1 and MP2 were the marks most often given.

In cases where the use of an index was mentioned, MP3 was not given if the description referred to iterating or 'searching through' the array as this is not the appropriate technique for the given scenario.

(b) A wide range of response was seen gaining the full range of marks.

A small number of excellent solutions were seen.

Many solutions contained some form of unnecessary iteration.

Mark point breakdown:

MP1: Usually included, but often mistakenly using the date function names as identifiers.

MP2: An accessible mark point.

MP3: Correctly addressed in many solutions.

MP4: Attempted in many solutions. Where a loop had been used there was a tendency to use the loop counter as the index to the `DaysInMonth` array rather than the month number as derived from the `ProductionDate` parameter.

MP5, MP6 and MP7: Often correctly implemented in meaningful solutions. Occasionally these omitted MP7.

MP8 and MP9: Use of `SETDATE()` was frequently correct but weaker solutions had often not declared a variable of type `DATE` to for this value to be assigned to. Several better solutions combined these mark points into a single statement and had often reached the 7-mark total by this point.

Common mistakes:

- As stated for MP1, the use of date function names as identifiers.
- Weaker solution often combined the function header for a date function as part of the assignment statement e.g.:

```
ThisMonth ← MONTH(Value: DATE) RETURNS INTEGER
```

- Some use of `OUTPUT` rather than `RETURN`.

Question 7

(a) Weaker answers that consisted only of module names and did not attempt to add parameters.

Mark point breakdown:

MP1: Given in the majority of answers. 'Reset' and 'Enable' occasionally swapped over.

MP2: Additional parameters were often included and return values were often omitted. Additionally, parameter name(s) were also often missing.

MP3: Additional parameters were often omitted.

MP4: Parameter `CC` was often omitted or shown in the wrong direction. Sometimes it was also shown as `BYREF`.

MP5: Rarely added.

(b) The majority of candidates gained MP1 for some reference to iteration. Often the description of exactly what was being repeated was vague. Answers that referred to `Sync` 'calling itself' were not given a mark.

Answers which gained MP1 often lacked the precision for MP2, which was rarely given.

Question 8

(a) Of the two styles of answers, loop-based solutions accounted for around 40 per cent and were usually the more competent, with some excellent solutions seen.

Selection-based answers tended to be weaker and in very many cases used literal grade boundary values taken from the example given in the question.

The scenario appeared to have been well-understood in the majority of cases, however occasional attempts were made to assign a mark to a grade e.g. `A ← '65'`

Weaker answers did not have a discernible structure and based solely on literal boundary values.

Loop-based solution mark point breakdown as follows:

MP1: The count-controlled loop was usually correct. Weaker solutions often attempted to loop through the mark range e.g. `FOR Mark ← 1 TO 75`. Nested loops were seen occasionally; the inner loop often repeating the range.

MP2: Usually correct in average-to-better solutions. A fairly common error used consecutive array values:

```
IF Mark >= GradeBoundary(n) AND Mark <= GradeBoundary(n+1)
```

MP3: Almost always followed MP2

MP4: Normally given as the third mark in the 'set' following MP2 and MP3

MP5 and MP6: As per the mark scheme example, an immediate `RETURN TRUE` within the loop followed by `RETURN FALSE` after the loop was seen in many better solutions.

Common mistakes:

- Invalid use of `LENGTH()` e.g. `FOR N ← 1 TO LENGTH(GradeBoundary)`.
- Attempting to check for values 'outside the range' e.g. `FOR Mark < GradeBoundary(1) AND Mark > GradeBoundary(5)`.
- The inclusion of an `ELSE` clause in the selection statement to reset the variable (MP5) or to immediately return `FALSE` allowing for just one pass through the loop code.

Selection-based solution mark point breakdown as follows:

MP1: Frequently given for 'any' correctly formed structure regardless of functionality. Multi-level `IF` statements were common. These tended to overflow the answer lines available. `CASE`

`... ENDCASE` syntax was often correctly given, although individual ranges were rarely correct and were usually based on literal example values.

MP2, MP3, MP4 and MP5: Rarely given.

MP6: Often given if the overall selection structure made some functional sense.

Common mistakes:

- Invalid use of Boolean operator in `CASE` condition: `>=63 AND <= 61`.

(b) Three marks or less given in around 75 per cent of cases.

As for the previous question part, a small number of very good solutions were seen. A number of candidates made no real attempt to answer the question although a number of mark points were particularly accessible.

Mark point breakdown as follows:

MP1: Given in most cases where a viable attempt had been made. A small number decided the module needed to be a function.

MP2: The lack of missing quotes around the literal filename string and missing filename in the `CLOSEFILE` statement (if there was one) combined to stop this mark being given in very many cases. `WRITE` mode was usually correctly identified.

MP3: The mark was frequently given. Some answers attempted a `REPEAT ... UNTIL` loop, often with problems in the conditional test, where some strange constructs were sometimes seen e.g.:

```
UNTIL Index = Result(Upperbound)
```

MP4: Often given.

MP5 and MP6: Give as a pair of marks for a single pseudocode line in many better solutions. Incorrect use of a function was widely seen; a common fault being to call the function name without assigning a return value but then to subsequently test something called `Flag` or `CheckMark`.

MP7: Awarded in many cases provided both the filename and variable name were present. Some incorrcet attempts were made using an assignment statement: `GRList.txt ← LineOfText`

MP8: Not commonly seen.

MP9: Often given. A common problem was to omit the separator characters.

Common mistakes:

- Missing quotation marks around the literal filename string.
- Omitting the filename from the `CLOSEFILE` statement.
- Attempting to loop until `EOF()` of a file which is open in write mode.
- Failing to initialise the loop counter in a `REPEAT ... UNTIL` loop.
- Missing separators in `OUTPUT` or attempt to concatenate an integer value.

(c) Correctly answered in just under 40 per cent of cases. Many candidates did not attempt to answer this question.

COMPUTER SCIENCE

Paper 9618/22

Fundamental Problem-solving and Programming Skills

Key messages

Accuracy and precision – these are two key skills of the computer scientist and there are questions on this examination paper where these skills were required to be demonstrated. In **Question 3(b)** the selection statement test required a precise test for the condition. The correct condition `OnStack = 60` is very different from incorrect conditions such as `OnStack >= 60`.

In **Question 5(c)** the task was to write the given pseudocode as a single selection construct. For one of the two available marks this required little more than the copying of two `ReturnValue` assignment statements. Many candidates lost the mark for either, the incorrect number of asterisks or (more commonly) the omission of the final closing bracket on the statement using the `TO_UPPER` and `RIGHT` functions.

Computer Science is a practical subject, and it is hoped that candidates can be exposed to as much of the content as possible is a practical way. **Question 1(b)** required a knowledge and understanding of the testing features of a typical Integrated Development Environment (IDE). Candidates need to improve their competence in this content of using IDE software.

General comments

Candidates need to understand that 'good examination technique' is a key foundation to answering the questions. Candidates can improve on this.

Question 7(c) some candidates had misunderstood the requirement of the question. The earlier **Question 7(a)** required answers which related to the given scenario; data items needed for the implementation of the module. Candidates were then given a lead in to **Question 7(c)** stating in the stem that the algorithms and module designs will be produced. The key element of the question was to state other items which would be defined at this design stage. Many candidates continued where they had left off from **Question 7(a)** incorrectly stating further data items such as a mobile telephone number.

Candidates need to be aware that if a question is shown as two marks, then the answer to gain the second mark must be something in addition to the first – not just a repeat or slight re-wording. This was often seen in **Question 7(a)**. The question asked to 'identify an item' and then give the 'justification' for its selection. An answer such as 'date of last visit' is good for the first mark but a justification 'to check when they last visited' was insufficient for the second mark.

Comments on specific questions

Question 1

(a) Many correct answers stated either 'maintenance' or a more precise answer of 'corrective maintenance'. The frequent incorrect answer given was 'testing'.

(b) (i) See the earlier comment on the 'Key messages' section. A few answers were seen where it was clear that the first task would be to set the breakpoint at the start of the function call. If the answer implied this – as it was the first feature described – then the candidate gained credit. Few answers made it clear that arriving at a breakpoint and stopping execution at a particular statement would be the first step. Good communication required an answer such as 'set the breakpoint at a given statement; at a given point was considered too vague. Similarly for the answer describing single

stepping. A statement such as 'execution of one statement/instruction at a time' would gain the mark; executing one step at a time was insufficient.

Candidates scored better describing the use of the 'watch window' stating it displays the current values of variables and/or expressions.

(c) Well answered. Candidates were able to describe two features which would help the programming at the coding stage. There were many answers considered creditworthy, and most were at the 'detail level' when the programmer was actually typing statements such as dynamic syntax checking, the auto-complete of a statement or automatic indentation of the code. Often the omission of the word 'auto' or 'dynamic' caused the non-award of the mark. At a more general level, an 'integrated compiler or interpreter' was creditworthy, but 'auto-documenter' was not.

Weaker answers sometimes tried to re-cycle the key terms used in the previous **Question 1(b)** and so gained no credit.

(d) Well answered by almost all candidates. The only error if present was to suggest `INTEGER` for the `Result` variable.

Question 2

(a) The majority of candidates were able to score marks with many answers securing the full six marks. Two distinctly different solutions were possible. The majority of answers seen had a loop structure which on each iteration added one more asterisk character to a string. Weaker answers usually scored at least the available mark for a correctly formed function header and ending. A not uncommon fundamental error seen was to correctly show a string variable parameter in the function header but then cancel out any understanding by the inclusion of an `INPUT` statement inside the body of the function for the same original string data value.

Candidates need to improve on their understanding of the Cambridge pseudocode syntax.

Common mistakes included:

- Omitting the assignment symbol from the loop statement.
- Using the function name `LENGTH` as an identifier.
- Use of '+' to concatenate strings.
- A formed statement such as attempting to assign a value to a function:
  e.g. `LEFT(InString, Index) ← '*'`.

(b) (i) Often the answers secured only one of the available two marks. Often the lost mark was the result of the incorrect syntax used for the ranges of the array (e.g. both stated as 1:100) or the incorrect upper bound(s).

(ii) The understanding that the parameter must be passed by reference was required. Candidates stated this either be the description 'by reference' or by using the pseudocode equivalent `BY REF`; either secured the mark.

Question 3

(a) (i) A number of weaker answers were seen. Some answers showing the correct initial values were seen.

(ii) Some correct answers which demonstrated an understanding were seen. The few correct answers seen stated that any value added to the stack will effectively overwrite any existing value at location `SP`, and therefore the implication is that there would be no merit in initialising the complete stack contents first.

(b) Varied levels of answer seen. The weaker answers often scored only the final available mark for the increment of the `SP` value and maybe the second mark for the setting of the return value as `FALSE`.

Question 4

A new approach to a pseudocode question which needed the appropriate design to function correctly. Most candidates, given the guidance in the question stem, correctly identified the need for two loops. Although a design with one loop and a single selection statement to test for the arrival of the warning time was equally acceptable. Most candidates correctly calculated the elapsed time but were then less successful in using this in the algorithm for the test for the two critical times having been reached.

Common errors included:

- Errors in the formulation of the procedure header.
- The same fundamental error described earlier for **Question 2(a)** – the input of the minutes and seconds Values in addition to those shown in the procedure header.
- Failure to convert the elapsed time to milliseconds.
- Omission of adding the start time to the elapsed time and final time.
- An attempt to assign some value to the `Tick` global variable.

Question 5

(a) There was an error in **Question 5(a)**. The question stated that the pseudocode contained three statements, instead of four statements that could generate a run-time error. This has been corrected in the published version of the paper.

Due to the issue with this question, its treatment in marking was considered carefully to make sure that no candidates were disadvantaged.

There were some clear and well explained answers seen. The error caused if the `Count` variable in the use of the `RIGHT` function was inappropriate (e.g. negative or too large). The `Data` array statement was similarly well explained stating that the value for variable `Number` could be outside the upper and lower bounds of the array.

Some weaker answers gave general answers stating that variable(s) may have been declared of the wrong data type.

(b) Stronger answers scored one or the two available marks. The answer required was that the structure would be a conditional loop. Any way in which the candidates stated this was mark-worthy. For example, a conditional loop, `REPEAT...UNTIL`, `WHILE...ENDWHILE`. Some answers suggested the candidate was on the right track but an answer such as 'loop' was insufficient for the first mark.

The second mark required a clear and concise statement that the run-time error would be caused by a failure to never meet the terminating condition. Weaker answers such as *'The program is in an infinite loop*' did not gain credit as this was considered the 'effect', not what causes the run-time error.

(c) A number of answers did not follow clearly the rubric of the question which required 'a single selection construct'. Candidates need to read the rubric of the question carefully.

Many answers lost one of the marks if the construct contained an `ELSE IF` clause which was not required. See the earlier comment in the second paragraph of the Key messages section.

Question 6

(a) This proved challenging for many candidates. A few strong answers were seen.

The basic requirement for the design were three conditional loops followed by testing for the conditions of the item numbers generated.

Most answers realised that the solution required the use three times of the `RAND` function. It was rare to see the correct expressions generating a random number in the required two ranges.

Errors seen included:

- The use of the given array name `Bread` as a new identifier.
- Confusion between the use of the generated array index and the contents of the array at this position.
- Correctly using a conditional loop until a non-blank element was found, but omission of the re-assignment of the array index – i.e. no `RAND()` generated value inside the loop.
- The use of a pre-condition loop that tested an element before the array index value had been assigned.

(b) A number of weaker answers were seen. The following answer would have gained both available marks:

'The creation of a list of either desirable or undesirable pairings. Then when a new pairing is generated, the list is searched.'

Non-Computer Science solutions such as *'Ask the customer if they are happy with the selection'* demonstrated a general misunderstanding as to what answer was expected given the key word 'design' in the question stem.

Question 7

(a) An exercise in abstraction; filtering from the list the data items which would be required to implement the module. A common mistake was to overlook the question directive to focus on the new module and instead suggested more general requirements for the loyalty card scheme.

Some candidates are not distinguishing between items of customer information and modules which would make up a final solution.

See the earlier comment in the General comments section referring to examination technique.

(b) A number of weaker answers were seen.

(c) A few answers scored the full three marks. See the earlier comment in the General comments section outlining the need to read the question and be clear as to what is required.

The most commonly seen correct answers described deciding on the choice of programming language and the drawing up of a test plan and testing strategy; 'testing' on its own was considered insufficient.

(d) A few answers gained the full available three marks. A not uncommon problem was to start the `Init()` module at the top of the hierarchy.

Question 8

(a) A very wide range of responses seen. Most candidates chose a `FOR` loop for the design, but this mark was usually lost as there was a failure to exit from the loop when a matching unassigned character was found. A better choice would have been a conditional loop.

The correct pseudocode syntax was often not present. Candidates realised that the `Character` array would be indexed but this combined with the dot notation of the relevant field was incorrect. However, if the candidate demonstrated some use of an index, then there was an attempt to follow through and gain some of the marks.

The most viable solutions used a flag to indicate the matching character found and the flag was correctly tested after the termination of the loop to determine which message should be output.

Common simple errors were often present and resulted in a loss of mark. This included failure to initialise the flag variable and a missing `ENDIF` statement.

(b) Generally, candidates scored better here on this file handling question than the previous **Question 8(a)**. Marks were lost due to, for example, incomplete/incorrect syntax; missing quotation marks

around the file name string, no `CLOSEFILE` statement, use of the wrong operator symbol to concatenate two strings and the omission of the assignment symbol in the `FOR` loop statement.

Some candidates declared the separator character as a constant which was considered good practice.

The marks were awarded for the construction of the single string for:

- The conversion of the player and level data using the `NUM_TO_STR` functions.
- The correct concatenation.
- The correct use of three separator characters.

(c) A few correct answers were seen. Few answers seen described the need to convert the Boolean value to either a string or character value in order that it could then be concatenated to the main string. The second mark required that the value would be added to the existing string using the same separator character. Some answers incorrectly suggested that this would require a different separator. Other incorrect answers suggested a different file mode would be needed, or the data item (unchanged Boolean) should be stored on a different line.

(d) A number of weaker answers were seen. Good communication was required to describe that some form of suffix would be added to the filename which could be incremented. Answers given for the 'example' were as simplistic as file1.txt, file2.txt, etc. which was enough to gain the example mark. Weaker answers often gave two unconnected filenames.

COMPUTER SCIENCE

Paper 9618/23

Fundamental Problem-solving and Programming Skills

Key messages

The emphasis for this paper is on the application of practical skills. Candidates need to have developed these and be able to apply them to the scenarios presented.

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 need to improve their competence in terms of designing and implementing a solution to a problem.

This is a technical subject and makes use of many technical words and phrases. These have specific, defined meanings and it is important that these are used correctly.

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

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

General comments

Candidates need to read each question carefully and it is important to clearly understand the question before attempting to answer it. 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.

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.

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

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

If typed answers are provided, then it is very helpful if these are organised so that individual answers do not span page breaks. This is particularly important for programming answers. If the question involves completing a table, then typed answers should clearly indicate any unfilled rows.

Comments on specific questions

Question 1

(a) The average mark awarded for this question was two. With a significant number gaining all four marks.

All candidates made an attempt at this question with candidates gaining most mark for row one and row four.

(b) Many candidates gained all three marks. The most common mistake was suggesting that the data type for row 2 was `DATE` instead of `STRING`.

(c) Many candidates gaining full marks. No discernible pattern was seen for mistakes made.

Question 2

(a) This question attracted a full range of responses. Many full-mark solutions were seen with the majority gaining mid-range marks. A few responses were not credit worthy.

Most candidates attempted to use a for-loop solution, but a smaller number attempted conditional loops to varying success with many of these failing to initialise the loop counter.

There was no discernible pattern to the mark points awarded.

Common mistakes included:

- Failing to declare all variables used with the loop counter often being omitted.
- Missing a prompt.
- Missing `ENDIF` from the selection statement.
- Not initialising the variable used to sum the positive integers.

(b) This style of question is where candidates have to describe why a construct was used in their solution to the problem set. To gain the second mark they needed to make reference to the specific context described in the question.

Candidates could either describe the use of iteration or selection. Most candidates chose the iterative construct to describe but many did not specify that 100 iterations were required.

A significant number of candidates did not use the correct term for the construct using 'while loop' or 'for loop' instead which was not enough to gain the first mark, but these candidates could still gain the 'use' mark if this was correct.

Question 3

This question is assessing the candidates understanding of the use of an array to implement a linked list. The first two parts of the question are specifically about the use of array indices in the context set out in the question.

(a) (i) Just under 40 per cent of candidates were awarded this mark.

Example of typical wrong answers:

- Value not in the list.
- Value not used.
- Smallest value allowed.

(ii) Around 5 per cent of candidates gained the mark for this question. The vast majority of candidates gave the range as 200.

(b) The marks awarded for this question were spread across the full range of marks available with a significant number gaining full marks. Weaker responses did not gain any marks.

No discernible pattern was detected to the mark points awarded.

(c) A reasonable attempt to answer this question with mark point three being the one being awarded most often.

Few candidates referred to `HeadPointer` to determine the start of the linked list.

Question 4

(a) This was generally a well answered question with three marks being the average mark awarded.

The most common mistakes made was the choice of conditional operators for identifying the range of marks to determine which papers needed to be checked.

(b) (i) Just under half of the candidates gained a mark for this question.

Most candidates did not describe that the variable `Mark` had to be used as the index to the `Check` array.

(ii) A wide range of response was seen with the full range of marks being awarded.

Two different approaches were possible either the use of just loops or the combination of a loop(s) and a selection structure.

Most candidates gained the mark for the procedure heading and ending along with at least one of the loop marks. The other accessible mark was for extracting a value from the `GB` array.

Common mistakes:

- Not using a mechanism to correctly identify a mark 2 above or 2 below a grade boundary.
- Not setting the elements of the `Check` array to `FALSE` if they did not need to be checked.

Question 5

(a) A substantial number of full-mark answers were seen with two being the average mark awarded for this question.

No discernible pattern seen for wrong answers given.

(b) A question to assess the ability to identify and describe the correct testing method to use for a given scenario.

Weaker responses were seen with one being the average mark awarded. Only a few candidates gained full marks for this question. A number of candidates did not attempt the question.

Question 6

(a) A wide range of response was seen gaining the full range of marks.

Two different approaches were given in the mark scheme with the vast majority of candidates using a loop solution. Around 2 per cent of candidates attempted the non-loop solution and these were of a high standard.

No candidates attempted solutions that used the `DIV` operator where no loop or selection constructs were needed.

There were a number of weaker responses where candidates did not make a valid attempt to answer the question.

Mark point breakdown for the loop solution:

MP1: The most common mark point awarded. Many candidates used `Year` as a parameter name which should not have been used as it is date function name.

MP2: Was the second most commonly awarded mark.

MP3: Commonly awarded but some candidates lost this mark point for incorrect syntax for the loop structure.

MP4: Many candidates lost this mark point for not attempting to use both `SETDATE()` and `DAYINDEX()`.

MP5: Often lost for not assigning the date created to a variable of type `DATE`.

MP6: If candidates made use of the `DAYINDEX()` function this mark point was also usually awarded.

MP7: Many candidates lost this mark point for not initialising their count variable.

MP8: This was often awarded as a follow-through when MP7 had been lost for missing initialisation.

Common mistakes:

- Incorrect use of `SETDATE()` – often for missing parameters or parameters of wrong type.
- `DAYINDEX()` being passed a parameter of the wrong type.
- Some use of `OUTPUT` rather than `RETURN`.

Question 7

(a) Weaker responses were seen with many candidates failing to gain any marks. Where candidates correctly identified the analysis stage, they often did not give an appropriate document that would be produced with many candidates giving 'a planning document' as their answer.

(b) Many candidates gained full marks with most candidates gaining two or three marks.

A common mistake was to suggest a module was needed to generate a customer ID as this was not part of the new module required.

Question 8

(a) A linear search question that incorporated the assessment of a candidates understanding of accessing specific fields in an array of records.

The average mark was just under four with the full range of marks given. Some excellent solutions were seen that closely matched the example solution given in the mark scheme.

Mark point breakdown as follows:

MP1: This was a mark which many candidates gained.

MP2: One of the most common marks points achieved.

MP3, MP4 and MP5 required candidates to have an understanding of how to use dot notation to access fields in a record. MP3 was available to candidates who had some understanding of dot notation but made an error in the syntax required.

MP3: Many candidates achieved this mark although they did not use the correct dot notation syntax required but the attempt contained all the required parts. Examples of these errors where MP3 was given as an attempt:

- ```
  IF Character.Player(Index) = ThisPLayer AND __
  Character.Role(Index) = ThisRole THEN
  ```
- ```
  IF Player.Character(Index) = ThisPLayer AND __
  Role.Character(Index) = ThisRole THEN
  ```

MP4 and MP5: Most candidates who gained MP4 also gained MP5 (and also MP3), but a number of candidates only checked one of the two fields and so only gained one of these MPs and were not awarded MP3.

MP6: Candidates who lost MP1 could gain this mark.

MP7 and MP8: Many candidates gained both these marks, but a number of candidates lost one of these mark points due to a missing `ENDIF` or for only have one `OUTPUT` statement.

Common mistakes:

- Failing to initialise both required variables.
- Incorrect dot notation.
- Missing `ENDIF` on selection statements.
- Missing separator in `OUTPUT` statements or using `&` to concatenate an integer value.

(b) A small number of very good solutions were seen. A number of weaker solutions were also seen. Many candidates did not make a valid attempt to answer the question although a number of mark points were particularly accessible.

Mark point breakdown as follows:

MP1: Missing quotes around the literal filename string and missing filename in the `CLOSEFILE` statement (if there was one) combined to stop this mark being given in many cases. `READ` mode was usually correctly identified.

MP2: The mark point most frequently given. Some answers attempted a conditional loop, often with problems in the conditional test or not initialising the loop counter. Some candiadtes looped until `EOF()` which was given if the index for the array was also iniilaised outside the loop and incremented in the loop.

MP3: Where a viable solution was attempted this mark point was often given.

MP4: Another mark point that was commonly given where a viable try to produce a solution had been made. This mark point was for an attempt to use the function `Extract()`.

MP5: This mark point was less accessiable than MP4 requiring all four fields to be extracted.

MP6: This was another step up from MP5 and was awarded on fewer occasions than MP5.

MP7: This was a furher step up and again was less frquently awarded than MP6.

Common mistakes:

- Missing quotation marks around the literal filename string.
- Omitting the filename from the `CLOSEFILE` statement.
- Failing to initialise the loop counter where a conditional loop was used.
- Incorrect use of dot notation.

(c) A small number of fully correct answers were seen. A number of weaker answers that were seen did not gain any marks for this question. Most answers lacked the precision and did not mention what was being saved to the file i.e. the `Character` array and that this would allow the game to be continued from the point it was previously saved.

COMPUTER SCIENCE

Paper 9618/31
Advanced Theory 31

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, for example in the areas of user-defined data types or algorithm construction, and answer questions requiring responses to be written in pseudocode using this syntax.

General comments

Candidates are advised to read questions carefully before beginning their answer in order to understand what is being asked of them so as to maximise their mark; making sure they answer the question that is asked.

Candidates must always make sure that they answer questions in the context of any scenario described in the question, rather than in generic terms, to receive maximum credit. In some cases, marks may be awarded for how an answer applies to the given scenario.

Candidates are also advised to use the correct computer science language and technical terms when answering questions.

Comments on specific questions

Question 1

(a) The vast majority of candidates achieved at least one mark here. Candidates who correctly wrote the truncated mantissa and correct exponent in the given grid or commented that the mantissa would need to be truncated, and also provided working to show how they arrived at their solution achieved the best marks.

(b) Candidates who gave the correct decimal number equivalent to the given binary number, along with sufficient working to show how the solution was achieved, gained the highest marks. Most candidates achieved at least one mark here.

Question 2

Candidates who only correctly identified features of a RISC processor, rather than trying to compare RISC with something else, such as a CISC processor, or stating what RISC does not have or is not good at, achieved the marks.

Question 3

(a) Most candidates achieved some marks for recognising that circuit switching made use of a dedicated channel, which was set up before transmission could begin. Candidates who also added that the circuit lasted for the duration of the transmission, achieved all three marks.

(b) Candidates were able to state one benefit or one drawback of circuit switching as a method of data transmission. Many candidates achieved both marks.

Question 4

This question required candidates to describe how the various layers of the TCP/IP protocol stack interact with each other. Many candidates achieved one or two marks for recognising that interaction could only take please between adjacent layers and/or data is added to packet headers as the packets pass through the layers. However, a significant number of candidates misunderstood the question and simply described the purpose of each layer.

Question 5

(a) Most candidates understood that a hashing algorithm was used in direct access methods, or that it was applied to the key field of a record in order to calculate an address for the record to be stored or located. Candidates were therefore credited with marks for their responses. However, many responses lacked sufficient detail or clarity to earn all three marks.

(b) Candidates were awarded marks for up to two solutions to overcome the use of a hashing algorithm, resulting in the same storage location being identified for more than one record. Correct solutions included: searching linearly for the next available space from the one identified by the hashing algorithm or storing the record in the next available space in an overflow area. Simply stating using 'open hash' or 'closed hash' was not considered to be a detailed enough answer at this level.

Question 6

(a) Candidates were mostly aware that a set is a composite data type and set theory operations, such as intersection and union, could be applied. Candidates who added a little more detail, such as, a list of unordered elements is included, or all the elements are of the same data type, achieved all three marks.

(b) The full range of marks was seen for this question, in which candidates declared a set data type to match some given requirements. Some good responses were seen and candidates are advised to refer to the A Level Computer Science Pseudocode Guide for more information on this topic.

Question 7

(a) This question was generally well answered, with many candidates achieving two or three marks. Only responses with six correct terms in the solution achieved three marks.

(b) Candidates who fully and correctly completed every square of the Karnaugh map with either a 0 or a 1 achieved both marks.

(c) Candidates who correctly added two loops to identify two groups of four 1s to their Karnaugh map achieved both marks.

(d) (i) Candidates who wrote the correct two term simplified sum-of-products taken directly from their Karnaugh map achieved both marks.

(ii) Candidates who correctly simplified their sum-of-products in **part 7(d)(i)** to its simplest form achieved this mark.

Question 8

(a) Candidates generally found this question difficult, but many did achieve one or two marks for describing the process of segmentation for memory management. Common correct points seen

included 'the logical address space is broken into varying sized blocks called segments', 'segments are numbered' and 'segment numbers are used as an index in a segment map table'.

(b) Candidates generally understood the meaning of disk thrashing and that it may occur when virtual memory is being used. Most candidates achieved at least one mark, but many good answers describing disk thrashing were seen and achieved all three marks.

Question 9

(a) Candidates who stated that the attributes would be declared as `PRIVATE` so that they are only accessible using the class's own methods', or similar achieved the mark. Many candidates incorrectly interpreted the use of the word `PRIVATE` to indicate that the data must not be shared with unauthorised users.

(b) The vast majority of candidates achieved at least one mark for this question, however, marks higher than three were rare. Most candidates managed to correctly declare two appropriate attributes. Many candidates also created two correct getters and used names matching those given in the question or the one created by the candidates for the attributes mark. For the setters mark, appropriate attributes and data types were required, which was rarely seen.

Question 10

(a) Most candidates achieved at least one mark for this question, but common errors included misplaced 'or' symbols '`|`'. Some candidates incorrectly began or ended their list of operators with one, for example, `| + | - | * | / |` and other candidates incorrectly used them between every item in `<label>`, for example, `<letter>|<digit>|<letter>|<digit>|<digit>`.

(b) (i) Most candidates achieved at least one mark for their syntax diagram. Candidates are reminded that diagrams should be drawn in pencil therefore allowing errors to be erased and redrawn neatly. Common errors seen here included not beginning with either a letter or symbol; not ending with either one or two symbols; not all arrowheads were present or correct.

(ii) Many candidates achieved marks for this question and a wide range of valid and creative solutions were seen. Candidates who accurately followed the description of the new syntax rule for `password`, at the start of the question, whilst constructing their Backus-Naur Form (BNF) solution, achieved the marks.

Question 11

(a) This question part was generally well answered. Marks were awarded for correctly adding the four new nodes; correctly applying joining arrows between nodes; correctly adding null pointers as appropriate with no additional data in the pointer boxes, or anywhere else on the diagram. Some common errors included missing arrowheads, pointers not originating from the correct left or right pointer box and numbers other than the null pointer written in the left and/or right pointer boxes.

(b) Candidates who correctly annotated the left and right pointer spaces in the table, as well as the free pointer achieved the marks. A common error seen was data written in row 7 of the table. As this represents the free pointer, which is not yet populated, this must be left blank.

(c) This question part was well answered, with most candidates achieving marks. Common errors seen here included the use of `RETURN` instead of `RETURNS` in the function definition line; the wrong value being assigned to the variable `NowPtr`, such as –1, in the second answer; use of the wrong array index variable such as `NewPtr` instead of `NowPtr`, in the third answer; reference to the left pointer instead of the right pointer in the final answer.

COMPUTER SCIENCE

Paper 9618/32
Advanced Theory 32

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, for example in the areas of user-defined data types or algorithm construction, and answer questions requiring responses to be written in pseudocode using this syntax.

General comments

Candidates are advised to read questions carefully before beginning their answer in order to understand what is being asked of them so as to maximise their mark; making sure they answer the question that is asked.

Candidates must always make sure that they answer questions in the context of any scenario described in the question, rather than in generic terms, to receive maximum credit. In some cases, marks may be awarded for how an answer applies to the given scenario.

Candidates are also advised to use the correct computer science language and technical terms when answering questions.

Comments on specific questions

Question 1

(a) The vast majority of candidates demonstrated knowledge of packet switching and how it is used to transmit messages across the network. Candidates who gave their answer using correct accurate terminology associated with packet switching achieved the highest marks.

(b) The vast majority of candidates achieved at least one mark for this question. Candidates who stated actual benefits, such as 'packets are more likely to arrive because if one is lost it can be retransmitted' and/or drawbacks such as 'there will be a time delay because packets need to be reassembled at the destination' rather than simple statements of fact, that could have answered **part 1(a)**, achieved the highest marks.

Question 2

(a) Candidates who described serial file organisation as 'records being stored chronologically' rather than 'files being stored chronologically', achieved the mark. Those who also stated that records are added at the end of the file achieved both marks.

(b) Many candidates gave a good example of the use of serial file organisation, such as 'to create unsorted transaction files', and achieved the mark.

Question 3

(a) Most candidates achieved one or two marks for this question; usually for describing records as a composite data type referencing other data types in its definition. Some candidates gave further information, such as: the data types referenced may be primitive or user-defined, or a group of related items can be included in the definition. These candidates achieved all three marks.

(b) The vast majority of candidates demonstrated a good ability to define a record data type using pseudocode and the data supplied in the question. Many high scoring responses were seen.

Question 4

(a) Most candidates achieved at least one mark, often for showing valid working, many of these candidates also achieved the second mark for giving the correct final answer.

(b) Many high scoring responses with good working were seen, and the full range of marks were awarded. Candidates who clearly showed how they achieved their final answer and then gave a correct mantissa and exponent in the answer grid achieved all four marks.

Question 5

(a) The vast majority of candidates achieved at least one mark for naming at least one protocol used by the Application Layer in the TCP/IP protocol suite, with many candidates correctly naming two. Candidates who then went on to describe the protocols they had named achieved all four marks.

(b) Many candidates demonstrated an understanding of where the Application Layer sits within the protocol suite and that it communicates with the Transport Layer. Some candidates also recognised that the Application Layer provides access to all the programs that exchange data, or that it interacts directly with the user. However, very few candidates gave a sufficiently detailed response to enable all three marks to be achieved.

Question 6

(a) Generally, candidates who understood how to answer this question achieved two or three marks, with two-mark answers usually having either one or more terms missing or having a repeated term within the response.

(b) This question was well answered with candidates who correctly completed every square of the Karnaugh map with either a 0 or a 1 achieving both marks.

(c) Candidates who correctly added two loops to identify two groups of four 1s to their Karnaugh map achieved both marks.

(d) Candidates who wrote the correct two term simplified sum-of-products taken from their Karnaugh map achieved both marks.

Question 7

(a) Candidates who gave appropriate reasons for the two given numbers to be invalid achieved the marks. For example, `21` is not valid because it begins with an even number and should begin with an odd number. `123` is invalid because it consists of three digits and it can only have one or two digits.

(b) Most candidates achieved at least one mark for this question, but common errors included misplaced 'or' symbols '`|`'. Some candidates incorrectly began or ended their list of symbols with one, for example, `| % | £ | # | @ | $|` and other candidates incorrectly used them between every item in `<number>`, for example, `<odd>|<odd>|<odd>|<odd>|<even>`.

(c) (i) Most candidates achieved at least one mark for their syntax diagram. Candidates are reminded that diagrams should be drawn in pencil therefore allowing errors to be erased and redrawn neatly. Common errors seen here included misspelling `number` as `numbers`, missing arrow heads or not ensuring that `number` had to be present once or twice only.

(ii) Many candidates achieved marks for this question with a wide range of valid solutions seen. Candidates who accurately followed the description of the new syntax rule for `code`, at the start of the question, whilst constructing their Backus-Naur Form (BNF), achieved the marks.

Question 8

Candidates who only correctly identified features of a CISC processor, rather than trying to compare CISC with something else, such as a RISC processor, or stating what CISC does not have or is not good at, achieved the marks.

Question 9

(a) Candidates who were able to outline how the kernel of an operating system acts as an interrupt handler with reference to such aspects as priority of the interrupt compared with the current task, the use of the interrupt dispatch table, and the storing and restoration of the process state using a stack, before and after the interrupt is handled, achieved the marks.

(b) (i) The vast majority of candidates were able to state the meaning of the term 'multi-tasking' as it occurs within an operating system.

(ii) Most candidates achieved at least one mark for describing the role of scheduling in relation to the implementation of multi-tasking in an operating system. Some of these candidates achieved another mark for describing the need for processor time or hardware and resources to be shared between tasks.

Question 10

(a) The vast majority of candidates were able to name one or more items that form the structure of objects and classes, such as, attributes, methods and a constructor, enabling candidates to achieve one or two marks. Candidates who were able to provide an expansion for at least one of these to identify its purpose, what it means or how it is used, achieved another mark.

(b) Candidates were asked to give three differences between an object and a class. Many candidates achieved one mark for a response similar to 'a class is a template from which objects are created'. Unfortunately, many candidates then re-wrote this answer in alternative ways for one or more of their other differences, so did not achieve any additional marks. However, a small number of candidates did give one or two additional differences for which they achieved additional marks.

Question 11

(a) This question part was mostly well answered. Marks were awarded for correctly adding the five new nodes; correctly applying joining arrows between nodes; correctly adding null pointers as appropriate with no additional data in the pointer boxes, or anywhere else on the diagram. Some common errors included missing arrowheads, pointers not originating from the correct left or right pointer box and numbers other than the null pointer written in the left and/or right pointer boxes.

(b) Candidates who correctly annotated the left and right pointer spaces in the table, as well as the free pointer achieved the marks. A common error seen was data written in row 8 of the table. As this represents the free pointer, which is not yet populated, this must be left blank.

(c) This question part was well answered, with most candidates achieving marks. Common errors seen here included the use of `RETURN` instead of `RETURNS` in the function definition line; the wrong variable being assigned the value of `RootPtr` in the second answer; use of the wrong comparison in the third answer, such as < or >= instead of >; reference to the right pointer instead of the left pointer in the final answer.

COMPUTER SCIENCE

Paper 9618/33
Advanced Theory 33

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, for example in the areas of user-defined data types or algorithm construction, and answer questions requiring responses to be written in pseudocode using this syntax.

General comments

Candidates are advised to read questions carefully before beginning their answer in order to understand what is being asked of them so as to maximise their mark; making sure they answer the question that is asked.

Candidates must always make sure that they answer questions in the context of any scenario described in the question, rather than in generic terms, to receive maximum credit. In some cases, marks may be awarded for how an answer applies to the given scenario.

Candidates are also advised to use the correct computer science language and technical terms when answering questions.

Comments on specific questions

Question 1

(a) The vast majority of candidates achieved at least one mark here. Candidates who correctly wrote the truncated mantissa and correct exponent in the given grid or commented that the mantissa would need to be truncated, and also provided working to show how they arrived at their solution achieved the best marks.

(b) Candidates who gave the correct decimal number equivalent to the given binary number, along with sufficient working to show how the solution was achieved, gained the highest marks. Most candidates achieved at least one mark here.

Question 2

Candidates who only correctly identified features of a RISC processor, rather than trying to compare RISC with something else, such as a CISC processor, or stating what RISC does not have or is not good at, achieved the marks.

Question 3

(a) Most candidates achieved some marks for recognising that circuit switching made use of a dedicated channel, which was set up before transmission could begin. Candidates who also added that the circuit lasted for the duration of the transmission, achieved all three marks.

(b) Candidates were able to state one benefit or one drawback of circuit switching as a method of data transmission. Many candidates achieved both marks.

Question 4

This question required candidates to describe how the various layers of the TCP/IP protocol stack interact with each other. Many candidates achieved one or two marks for recognising that interaction could only take please between adjacent layers and/or data is added to packet headers as the packets pass through the layers. However, a significant number of candidates misunderstood the question and simply described the purpose of each layer.

Question 5

(a) Most candidates understood that a hashing algorithm was used in direct access methods, or that it was applied to the key field of a record in order to calculate an address for the record to be stored or located. Candidates were therefore credited with marks for their responses. However, many responses lacked sufficient detail or clarity to earn all three marks.

(b) Candidates were awarded marks for up to two solutions to overcome the use of a hashing algorithm, resulting in the same storage location being identified for more than one record. Correct solutions included: searching linearly for the next available space from the one identified by the hashing algorithm or storing the record in the next available space in an overflow area. Simply stating using 'open hash' or 'closed hash' was not considered to be a detailed enough answer at this level.

Question 6

(a) Candidates were mostly aware that a set is a composite data type and set theory operations, such as intersection and union, could be applied. Candidates who added a little more detail, such as, a list of unordered elements is included, or all the elements are of the same data type, achieved all three marks.

(b) The full range of marks was seen for this question, in which candidates declared a set data type to match some given requirements. Some good responses were seen and candidates are advised to refer to the A Level Computer Science Pseudocode Guide for more information on this topic.

Question 7

(a) This question was generally well answered, with many candidates achieving two or three marks. Only responses with six correct terms in the solution achieved three marks.

(b) Candidates who fully and correctly completed every square of the Karnaugh map with either a 0 or a 1 achieved both marks.

(c) Candidates who correctly added two loops to identify two groups of four 1s to their Karnaugh map achieved both marks.

(d) (i) Candidates who wrote the correct two term simplified sum-of-products taken directly from their Karnaugh map achieved both marks.

(ii) Candidates who correctly simplified their sum-of-products in **part 7(d)(i)** to its simplest form achieved this mark.

Question 8

(a) Candidates generally found this question difficult, but many did achieve one or two marks for describing the process of segmentation for memory management. Common correct points seen

included 'the logical address space is broken into varying sized blocks called segments', 'segments are numbered' and 'segment numbers are used as an index in a segment map table'.

(b) Candidates generally understood the meaning of disk thrashing and that it may occur when virtual memory is being used. Most candidates achieved at least one mark, but many good answers describing disk thrashing were seen and achieved all three marks.

Question 9

(a) Candidates who stated that the attributes would be declared as `PRIVATE` so that they are only accessible using the class's own methods', or similar achieved the mark. Many candidates incorrectly interpreted the use of the word `PRIVATE` to indicate that the data must not be shared with unauthorised users.

(b) The vast majority of candidates achieved at least one mark for this question, however, marks higher than three were rare. Most candidates managed to correctly declare two appropriate attributes. Many candidates also created two correct getters and used names matching those given in the question or the one created by the candidates for the attributes mark. For the setters mark, appropriate attributes and data types were required, which was rarely seen.

Question 10

(a) Most candidates achieved at least one mark for this question, but common errors included misplaced 'or' symbols '`|`'. Some candidates incorrectly began or ended their list of operators with one, for example, `| + | - | * | / |` and other candidates incorrectly used them between every item in `<label>`, for example, `<letter>|<digit>|<letter>|<digit>|<digit>`.

(b) (i) Most candidates achieved at least one mark for their syntax diagram. Candidates are reminded that diagrams should be drawn in pencil therefore allowing errors to be erased and redrawn neatly. Common errors seen here included not beginning with either a letter or symbol; not ending with either one or two symbols; not all arrowheads were present or correct.

(ii) Many candidates achieved marks for this question and a wide range of valid and creative solutions were seen. Candidates who accurately followed the description of the new syntax rule for `password`, at the start of the question, whilst constructing their Backus-Naur Form (BNF) solution, achieved the marks.

Question 11

(a) This question part was generally well answered. Marks were awarded for correctly adding the four new nodes; correctly applying joining arrows between nodes; correctly adding null pointers as appropriate with no additional data in the pointer boxes, or anywhere else on the diagram. Some common errors included missing arrowheads, pointers not originating from the correct left or right pointer box and numbers other than the null pointer written in the left and/or right pointer boxes.

(b) Candidates who correctly annotated the left and right pointer spaces in the table, as well as the free pointer achieved the marks. A common error seen was data written in row 7 of the table. As this represents the free pointer, which is not yet populated, this must be left blank.

(c) This question part was well answered, with most candidates achieving marks. Common errors seen here included the use of `RETURN` instead of `RETURNS` in the function definition line; the wrong value being assigned to the variable `NowPtr`, such as –1, in the second answer; use of the wrong array index variable such as `NewPtr` instead of `NowPtr`, in the third answer; reference to the left pointer instead of the right pointer in the final answer.

COMPUTER SCIENCE

Paper 9618/41
Practical

Key messages

Candidates need to submit all of their program code in the evidence document. This must include code copied from their IDE and not screenshotted because a screenshot often includes illegible text especially if there is a background colour applied with coloured font. The screenshots for the outputs from programs should be either a white background with black font, or the reverse. Candidates need to make sure all of the text is contained in the evidence document and is legible.

When writing Python code candidates need to make sure indentations have copied correctly and are correct in each response.

Some candidates start a question and then leave later parts blank, almost all questions can be attempted in isolation and the code is marked in isolation, therefore candidates should be attempting all programs and including their partial solutions for each question as these could gain additional marks. The answers do not need to be complete and fully working to gain some of the marks.

General comments

Candidates often demonstrated a good understanding of object-oriented programming (OOP), declaring classes and get methods correctly. Some candidates were then unable to create instances of the objects, for example storing the data in an array instead of instantiating an object. Few candidates could then use these created objects and manipulate them within the program.

Candidates were often unfamiliar with 2D arrays, often creating 1D arrays or using incorrect syntax for 2D arrays in their programming language. These errors were often repeated with few candidates being able to manipulate the 2D arrays i.e. storing data, accessing data, or iterating through the elements.

Comments on specific questions

Question 1

(a) Candidates were often able to declare the function and array. Some candidates opened the text file successfully, but fewer proceeded to close the file, or they closed the file in an inappropriate place for example within the loop through each element in the file.

Some candidates looped until the end of file whilst some looped a set number of times which was appropriate as the number of elements was known from the file. Some candidates did not return the populated array as per the requirements in the question.

Some candidates made appropriate use of exception handling to manage the file not being found.

(b) (i) Many candidates were able to declare the appropriate function and provide a parameter. Fewer candidates were able to provide the appropriate formatted data, common errors included missing the required space between each element of outputting the data in the function instead of returning the formatted string.

(ii) When calling functions candidates need to consider which functions require parameters and which functions return values. Returned values need to be used appropriate, for example being stored. In this question the returned array from the first function call needed to be used as the parameter to

the second function call, candidates often called both functions without using this returned value correctly. Some candidates did not output the value returned from the second function call.

(iii) Some candidates produced a screenshot with the correct output, showing each element from the text file in order with a space between each element. Common errors included outputting a list data structure and therefore not having the data in the required format or missing one or more of the strings. Some candidates had screenshots that were cut off and did not show the full output from their program.

(c) Some candidates were able to declare an appropriate function that took the required parameters. Fewer candidates were able to iterate through each character in both of the parameter strings, some candidates compared the whole strings or included an error that meant each character in turn was not checked. Some candidates returned the parameters 1 and 2 in the incorrect order, returning 1 when the first parameter was greater than the second instead of the requirements of it being smaller.

(d) (i) Candidates often attempted to write a sorting algorithm but fewer made use of their function from **Question 1(c)** to compare the strings, instead comparing the strings directly in the bubble sort.

(ii) Responses often called both functions but did not make appropriate use of parameters and return values, some candidates did not output the final return value.

(iii) Some candidates produced the correct output showing the sorted data, some responses had sorted data but had one or more value missing for example the last string 'yellow' was missing from the output.

Question 2

(a) (i) Candidates often had a good understanding of OOP and were able to declare the class and constructor appropriately. Some candidates had syntax errors in their languages, for example incorrect indenting in Python that meant the constructor was not within the class declaration, or missing the end class statement in VB.NET. Some candidates missed the requirement for private attribute and declared them as public or did not include a comment or other means to identify they were private.

(ii) Candidates often demonstrated good knowledge of get methods and were able to write appropriate code for each method. Some candidates in Python missed the required self (or equivalent), some candidates sent parameters to their functions and then attempted to use these.

(b) (i) Some candidates were able to create instances of the classes, common errors included creating arrays for each horse with the values instead of an object for the class. Some candidates created tow objects but did not store them in the array. With private attribute candidates needed to use the get methods to access and then output the name of both horse objects, some candidates output the text of each name or accessed the attribute directly.

(ii) Some candidates were able to produce the output of the two names.

(c) (i) Candidates were often able to declare the class but some candidates attempted to introduce inheritance which was not appropriate in this scenario and not included in the question. This often led to additional errors in the constructor, for example calling a parent constructor which was not a requirement. Candidates were often able to declare the constructor with the appropriate parameters, some candidates did not include a declaration for the private attributes or declared them as public. A common error was missing the requirement to also write the get methods in this question with some responses only declaring the class and constructor.

(ii) Some candidates took the required data as input, but fewer did this the correct quantity of times. Some candidates attempted to validate the inputs. A common error was stopping the loop as soon as the data was valid which prevented four heights and risks being input, or the validation not stopping the counter from incrementing so if invalid data was entered once then only three heights and risks would be input and used overall.

Some candidates were able to create instances of the class with the values input, fewer candidates stored these in an appropriate array (or equivalent) data structure.

(d) Candidates often found this question challenging. Some candidates wrote a function and not a method, then using data that was input by the user instead of accessing the attributes from the class. Some candidates declared new variables within the function and then attempted to use these variables in the comparisons when they did not contain any data.

Candidates often attempted to compare the data and return value based on the data, some of the return values were strings or percentage values for example '0.8' being returned instead of performing the calculation. When the fence height was more than the max attribute some candidates returned 20 instead of multiplying the success by 0.2 to calculate the result.

(e) (i) Some candidates were able to call the correct method for each horse object with each fence object's attributes. These candidates were often able to output the correct values for each horse at each fence. Some candidates produced an output of the horse name for each fence but did not include the chance of success. Candidates often did not produce a response for this question especially where they had partial attempts at **Question 2(d)**.

(ii) This question required the code written in **Question 2(e)(i)** to be modified. Some candidates were able to calculate a running total for each horse and then calculate and output the average. Candidates often did not attempt this question.

(iii) Some candidates generated the correct output, showing the result for each horse in each fence and the final result with the winning horse. Some candidates showed the output for each fence but did not show the data being input for each fence.

Question 3

(a) Candidates were often able to declare an array, but fewer could create a 2D array. Some candidates wrote a comment to indicate the creation of an array but did not actually create a data structure. The question required each element to be initialised to `-1`, some candidates initialised an array of 2 elements each with 20 elements, instead of 20 elements with 2 elements each. More candidates declared the two variables accurately, but some candidates stored the incorrect initial values in each variable.

(b) Candidates were often able to write partial solutions to this ask, commonly the header and storing the data input in the correct position. Some candidates did not make use of the 2D array accurately, accessing a 1D array or other structure repeatedly. Few candidates were able to create a fully functional insert function.

(c) (i) Some candidates were able to access each element in the 2D array in turn and output each value. Candidates often did not use a 2D array appropriately or accessed a 1D array.

(ii) Candidates were often able to call both of the subroutines even if they had not attempted or succeeded at the previous question parts.

(iii) Few candidates generated the correct output.

(d) (i) Candidates often attempted this question, common errors were the inappropriate use of the 2D array, for example using a 1D array. Some candidates were able to produce partial solutions, for example comparing the data in the parameter to the element in `FirstNode` or attempting to follow the pointers. A small number of candidates were able to produce working solutions.

(ii) Some candidates called the appropriate subroutines with the parameter for the first call. Some candidates missed the required output the string 'After' before the list was output.

(iii) Few candidates were able to produce the correct output.

COMPUTER SCIENCE

Paper 9618/42
Practical

Key messages

Candidates need to submit all of their program code in the evidence document. This must include code copied from their IDE and not screenshotted because a screenshot often includes illegible text especially if there is a background colour applied with coloured font. The screenshots for the outputs from programs should be either a white background with black font, or the reverse. Candidates need to make sure all of the text is contained in the evidence document and is legible.

When writing Python code candidates need to make sure indentations have copied correctly and are correct in each response.

Some candidates start a question and then leave later parts blank, almost all questions can be attempted in isolation and the code is marked in isolation, therefore candidates should be attempting all programs and including their partial solutions for each question as these could gain additional marks. The answers do not need to be complete and fully working to gain some of the marks.

General comments

Candidates often demonstrated a good understanding of object-oriented programming, declaring classes and get methods correctly. Some candidates were then unable to create instances of the objects, for example storing the data in an array instead of instantiating an object. Few candidates could then use these created objects and manipulate them within the program.

Candidates were often unfamiliar with 2D arrays, often creating 1D arrays or using incorrect syntax for 2D arrays in their programming language. These errors were often repeated with few candidates being able to manipulate the 2D arrays i.e. storing data, accessing data or iterating through the elements.

Comments on specific questions

Question 1

(a)(i) Candidates were often familiar with classes and constructors and were able to write the appropriate code to declare these. Some responses declared the attributes without the required private condition or did not include the required parameters in the constructor.

(ii) Many candidates were able to declare the required get methods. Some responses included parameters or attempted to read a value from the user to then return.

(b)(i) Candidates were often able to declare a new array, some candidates had a comment to state the intention of an array but did not actually create a data structure. Some candidates used an inappropriate data type, for example initialising an array with string or integer values when the array needed to store data of type `EventItem`.

(ii) Candidates were often unable to create new instances of an object. Common errors included assigning the values for each event to a variable or creating an instance and then storing it in a variable instead of in the array.

(c) This question required a new class creating that was not related in terms of inheritance to the previous class. A common error was including inheritance in both the class and constructor

declaration. This question required the creation of the get method which was often missed by candidates.

(d) Some candidates found this question challenging and often wrote a function that read in or initialised new values, instead of a method that used the data for an object. The solutions needed to make use of the two parameters and then access the appropriate skill level for the type of event. A common error was not including a check on the type of event and then declaring a new variable for the skills value that did not hold any data. Some candidates were able to produce a working solution that made use of the correct attributes within the class and perform the correct calculation.

(e) (i) Some candidates were able to create new instances of the correct object with the required data, some candidates created an array for each character and then stored each value as an element within the array. Some responses did not include the name of the character in the object creation, instead naming the variable the character name to store the object within.

(ii) Candidates found this question challenging. Some common errors included manually using the data for each event instead of accessing each of the objects. Candidates often called `CalculateScore` but fewer did this as a method for each of the characters, more candidates were able to compare the values that were returned from these function calls. The stronger responses made appropriate use of the objects throughout and identified the need to use the declared get methods due to the attributes initially being created as private.

(iii) Many candidates did not produce a response to this question. Some candidates were able to get the correct output from the program.

Question 2

(a) This question required the creation of a record structure or an equivalent where a programming language does not contain a record structure, such as a class. A common error was the creation of individual data structures for each of the array and values instead of attempting a record structure. Some candidates created an appropriate class in Python and Java and created the attributes. In VB.NET candidates often created a structure or a class. In Python some candidates provided a comment for the array declaration but did not actually create a data structure for the array.

(b) Candidates that created a record or class in **Question 2(a)** were often able to edit this and produce the structure with the initialised values, or they created an instance of the class or record and used the appropriate values to initialise the data. Some candidates declared variables and a new array for each of the values but did not show that these were part of a record structure (or equivalent).

(c) Candidates were often able to identify the missing code for the first space and the last space. The middle comparison was less often correctly identified. Candidates were still required to use a record structure and if they followed the example pseudocode, they were often able to do this by amending it for their class or structure. Some of the weaker responses replicated the pseudocode exactly with converting it into their chosen language, for example using `ENDIF` when their language did not have an `ENDIF`.

There were many different ways that candidates dealt with the use of passing the data by reference in the pseudocode which were often successful, for example passing data as parameters and then returning the data.

(d) Responses to this question often output every value in the array without consideration of the data in the queue. The question required data to be iterated through starting with the value at the head pointer and concatenating all values insert into the array, this meant that the loop had to stop when it was one element before the tail pointer. Some responses output the data in each element within the function instead of returning the final concatenated string.

(e) (i) Candidates were often able to take data as input. Some responses validated this data appropriately, common errors in validation included stopping the loop as soon as invalid data was input or including each invalid input as one of the 10 integers to include in the queue.

Some candidates called the `Enqueue` function multiple times with the same data, for example calling it once to store the data then using the function call a second time in a conditional statement

to check the return value; this meant that the data was inserted into the queue twice instead of once.

Some responses called the function `ReturnAllData` but did not output the data that this function returned.

(ii) Some candidates produced an output of the data in the queue but did show the data being inserted. Some responses had an output of an array or list without the data in the required format.

(f) Candidates often had an understanding of how dequeue should work, few candidates were able to identify all of the conditions whereby the queue would be empty. Responses often checked if the head pointer was –1 but did not consider when the queue had already had data enqueued and dequeued.

Some candidates used the `Enqueue` function provided in pseudocode and accessed the data at the tail pointer instead of at the head pointer. Candidates more often accessed the correct data and returned this as well as incrementing the head pointer.

(g)(i) Some candidates were able to call the correct function twice and stored the return value each time for use in the comparison. Some candidates called `Dequeue` more than twice, for example calling it once, then a second time in a comparison, then a third time and a fourth time in a comparison. This led to the more elements being removed from the queue than required.

(ii) Candidates often did not produce an output from their program. Some candidates had the correct output showing the data being input and then the correct return values.

Question 3

(a) Candidates were often unable to create a 2D array with the required number of elements. Some candidates created a 1D array or created it with incorrect elements, for example 3 elements each containing 7 spaces. Some candidates provided a comment to declare the intention of an array but did not actually create a data structure or initialise any data within it.

(b) Candidates were often able to open the file to read data from, but fewer proceeded to also close the file. Some candidates created the function header but did not return the array that they had populated at the end of the function. Candidates were often able to loop the required number of times, either to the end of the file or 7 times and then read data within the loop. Some candidates made good use of exception handling, a common error was not including all of the file access code with the try clause, for example opening the file within the try but then reading the data and closing the file outside of the exception.

(c) Candidates who had created a 2D array were often able to iterate through each element. Candidates often attempted to use a 1D array and only accessed the single dimension.

(d) This question required candidates to sort the 2D array based on two of the elements. Some candidates wrote a bubble sort for a 1D array but were often able to return their attempt at the sorted array. The stronger responses often checked both conditions for swapping in one condition and then swapped all required elements, a common error was only swapping the index that was being used in the comparison, for example the game level was swapped for two of the players but not the score and player ID.

Some candidates attempted to produced two sorting algorithms, one for the game level and one for the score, which often worked for that element but did not fully sort the array as required.

(e) (i) Candidates often attempted to call the data and output the required messages. Some of these did not make use of the parameters and return value correctly, for example they did not store the return value from the function calls.

(ii) Candidates often did not produce an output from their program. Some responses showed the data before sorting accurately but the sorted data was in the incorrect order.

COMPUTER SCIENCE

Paper 9618/43
Practical

Key messages

Candidates need to submit all of their program code in the evidence document. This must include code copied from their IDE and not screenshotted because a screenshot often includes illegible text especially if there is a background colour applied with coloured font. The screenshots for the outputs from programs should be either a white background with black font, or the reverse. Candidates need to make sure all of the text is contained in the evidence document and is legible.

When writing Python code candidates need to make sure indentations have copied correctly and are correct in each response.

Some candidates start a question and then leave later parts blank, almost all questions can be attempted in isolation and the code is marked in isolation, therefore candidates should be attempting all programs and including their partial solutions for each question as these could gain additional marks. The answers do not need to be complete and fully working to gain some of the marks.

General comments

Candidates often demonstrated a good understanding of object-oriented programming (OOP), declaring classes and get methods correctly. Some candidates were then unable to create instances of the objects, for example storing the data in an array instead of instantiating an object. Few candidates could then use these created objects and manipulate them within the program.

Candidates were often unfamiliar with 2D arrays, often creating 1D arrays or using incorrect syntax for 2D arrays in their programming language. These errors were often repeated with few candidates being able to manipulate the 2D arrays i.e. storing data, accessing data, or iterating through the elements.

Comments on specific questions

Question 1

(a) Candidates were often able to declare the function and array. Some candidates opened the text file successfully, but fewer proceeded to close the file, or they closed the file in an inappropriate place for example within the loop through each element in the file.

Some candidates looped until the end of file whilst some looped a set number of times which was appropriate as the number of elements was known from the file. Some candidates did not return the populated array as per the requirements in the question.

Some candidates made appropriate use of exception handling to manage the file not being found.

(b) (i) Many candidates were able to declare the appropriate function and provide a parameter. Fewer candidates were able to provide the appropriate formatted data, common errors included missing the required space between each element of outputting the data in the function instead of returning the formatted string.

(ii) When calling functions candidates need to consider which functions require parameters and which functions return values. Returned values need to be used appropriate, for example being stored. In this question the returned array from the first function call needed to be used as the parameter to

the second function call, candidates often called both functions without using this returned value correctly. Some candidates did not output the value returned from the second function call.

(iii) Some candidates produced a screenshot with the correct output, showing each element from the text file in order with a space between each element. Common errors included outputting a list data structure and therefore not having the data in the required format or missing one or more of the strings. Some candidates had screenshots that were cut off and did not show the full output from their program.

(c) Some candidates were able to declare an appropriate function that took the required parameters. Fewer candidates were able to iterate through each character in both of the parameter strings, some candidates compared the whole strings or included an error that meant each character in turn was not checked. Some candidates returned the parameters 1 and 2 in the incorrect order, returning 1 when the first parameter was greater than the second instead of the requirements of it being smaller.

(d) (i) Candidates often attempted to write a sorting algorithm but fewer made use of their function from **Question 1(c)** to compare the strings, instead comparing the strings directly in the bubble sort.

(ii) Responses often called both functions but did not make appropriate use of parameters and return values, some candidates did not output the final return value.

(iii) Some candidates produced the correct output showing the sorted data, some responses had sorted data but had one or more value missing for example the last string 'yellow' was missing from the output.

Question 2

(a) (i) Candidates often had a good understanding of OOP and were able to declare the class and constructor appropriately. Some candidates had syntax errors in their languages, for example incorrect indenting in Python that meant the constructor was not within the class declaration, or missing the end class statement in VB.NET. Some candidates missed the requirement for private attribute and declared them as public or did not include a comment or other means to identify they were private.

(ii) Candidates often demonstrated good knowledge of get methods and were able to write appropriate code for each method. Some candidates in Python missed the required self (or equivalent), some candidates sent parameters to their functions and then attempted to use these.

(b) (i) Some candidates were able to create instances of the classes, common errors included creating arrays for each horse with the values instead of an object for the class. Some candidates created tow objects but did not store them in the array. With private attribute candidates needed to use the get methods to access and then output the name of both horse objects, some candidates output the text of each name or accessed the attribute directly.

(ii) Some candidates were able to produce the output of the two names.

(c) (i) Candidates were often able to declare the class but some candidates attempted to introduce inheritance which was not appropriate in this scenario and not included in the question. This often led to additional errors in the constructor, for example calling a parent constructor which was not a requirement. Candidates were often able to declare the constructor with the appropriate parameters, some candidates did not include a declaration for the private attributes or declared them as public. A common error was missing the requirement to also write the get methods in this question with some responses only declaring the class and constructor.

(ii) Some candidates took the required data as input, but fewer did this the correct quantity of times. Some candidates attempted to validate the inputs. A common error was stopping the loop as soon as the data was valid which prevented four heights and risks being input, or the validation not stopping the counter from incrementing so if invalid data was entered once then only three heights and risks would be input and used overall.

Some candidates were able to create instances of the class with the values input, fewer candidates stored these in an appropriate array (or equivalent) data structure.

(d) Candidates often found this question challenging. Some candidates wrote a function and not a method, then using data that was input by the user instead of accessing the attributes from the class. Some candidates declared new variables within the function and then attempted to use these variables in the comparisons when they did not contain any data.

Candidates often attempted to compare the data and return value based on the data, some of the return values were strings or percentage values for example '0.8' being returned instead of performing the calculation. When the fence height was more than the max attribute some candidates returned 20 instead of multiplying the success by 0.2 to calculate the result.

(e) (i) Some candidates were able to call the correct method for each horse object with each fence object's attributes. These candidates were often able to output the correct values for each horse at each fence. Some candidates produced an output of the horse name for each fence but did not include the chance of success. Candidates often did not produce a response for this question especially where they had partial attempts at **Question 2(d)**.

(ii) This question required the code written in **Question 2(e)(i)** to be modified. Some candidates were able to calculate a running total for each horse and then calculate and output the average. Candidates often did not attempt this question.

(iii) Some candidates generated the correct output, showing the result for each horse in each fence and the final result with the winning horse. Some candidates showed the output for each fence but did not show the data being input for each fence.

Question 3

(a) Candidates were often able to declare an array, but fewer could create a 2D array. Some candidates wrote a comment to indicate the creation of an array but did not actually create a data structure. The question required each element to be initialised to `-1`, some candidates initialised an array of 2 elements each with 20 elements, instead of 20 elements with 2 elements each. More candidates declared the two variables accurately, but some candidates stored the incorrect initial values in each variable.

(b) Candidates were often able to write partial solutions to this ask, commonly the header and storing the data input in the correct position. Some candidates did not make use of the 2D array accurately, accessing a 1D array or other structure repeatedly. Few candidates were able to create a fully functional insert function.

(c) (i) Some candidates were able to access each element in the 2D array in turn and output each value. Candidates often did not use a 2D array appropriately or accessed a 1D array.

(ii) Candidates were often able to call both of the subroutines even if they had not attempted or succeeded at the previous question parts.

(iii) Few candidates generated the correct output.

(d) (i) Candidates often attempted this question, common errors were the inappropriate use of the 2D array, for example using a 1D array. Some candidates were able to produce partial solutions, for example comparing the data in the parameter to the element in `FirstNode` or attempting to follow the pointers. A small number of candidates were able to produce working solutions.

(ii) Some candidates called the appropriate subroutines with the parameter for the first call. Some candidates missed the required output the string 'After' before the list was output.

(iii) Few candidates were able to produce the correct output.