COMPUTER SCIENCE

Paper 9618/11
Theory Fundamentals

Key messages

Candidate responses are expected to include more than simply general knowledge. There should be some demonstration of technical knowledge and appropriate use of the terminology associated with the subject.

Some candidates need to be careful with the use of terminology. For example, information was used frequently instead of data, and memory instead of storage.

Candidates should be encouraged to look carefully at the command words in the question. A question that begins with the word 'state' or 'identify' requires a different response to one that starts with the word 'explain'.

Some questions may ask for a justification of, for example, the use of application A rather than application B. This requires candidates to describe **why** application A would be used rather than application B. Simply describing **how** application A and application B are used does not answer the question.

General comments

Questions about low level languages and 3D printers were usually completed successfully, the questions about embedded systems and internal components of the CPU were more challenging.

Comments on specific questions

Question 1

(a) This question was generally answered very well. Many candidates were able to correctly match each term to the corresponding description.

(b) There were many correct definitions. Some candidates need to understand that bit depth means bits per pixel not the number of bits used to represent the image. The explanation required candidates to think carefully about their answers. It asked how changing the bit depth affects the image. Many responses included statements about file size which did not answer the question. Some candidates need to understand that it needs to be made clear in their answers whether the bit depth is being increased or decreased. Statements such as *'increasing the bit depth increases the number of bits per pixel'* which simply repeat the definition are not enough. There needs to be an explanation of how the image is affected.

(c) There were some very good answers to this question. The most popular correct reasons were reduced bandwidth usage and the limits imposed on attachments. Some candidates need to be careful when describing a reduced transmission time so that it is clear that it is the total time taken that is reduced and not the speed of data transmission.

Question 2

(a) There were some good answers to this question. Many candidates need to improve their understanding of the function of a router in a network. Answers to questions such as this are expected to include more technical detail, rather than general knowledge.

(b) This question was generally not answered well. Answers were expected to demonstrate some technical knowledge of the devices given in the table. Statements such as '*allows a device to connect wirelessly*' for the purpose of a Wireless Access Point are not clear enough.

(c) Many candidates were able to state that the requirement for an internet connection was a drawback of storing files on a public cloud. Candidates found it more challenging to give a second drawback. Many of the responses applied equally to a LAN or a private cloud. There seemed to be a general misconception that because the files were on a public cloud, they were accessible to anyone.

(d) This question was generally answered well. The most popular advantage given was the reduction in the number of collisions because of the direct connections. Some candidates need to understand that in a bus network, a device going down does not necessarily affect the rest of the network. If the main cable is broken, the whole network fails.

Question 3

(a) This question was generally answered well. Some candidates need to use a recognised convention for indicating the relationships.

(b) This question was generally answered well. Most candidates were able to correctly identify three advantages of a relational database.

(c) (i) There were many correct answers to this question. The most frequent incorrect answer given was to use the `CREATE TABLE` statement instead of the `CREATE DATABASE` statement.

(ii) There were a small number of completely correct SQL scripts. Many candidates found joining the two correct tables challenging. A frequent error was the use of `COUNT` instead of `SUM` in the `SELECT` clause. Some responses with an otherwise correct statement for the identification of the `CustomerID`, did not include the required quotation marks.

Question 4

(a) This question was generally answered well. Many candidates were able to correctly carry complete the truth table.

(b) This question was generally answered well. Many candidates were able to draw an appropriate logic circuit.

Question 5

(a) Many candidates were able to correctly identify two other special purpose registers. Some candidates found it more challenging when stating their roles.

(b) This question was generally not answered well. Many candidates need to improve their understanding of what is meant by the Immediate Access Store. Answers such as '*immediate access store is storage that can be accessed immediately*' which simply return the name are vague and imprecise.

(c) (i) Candidates were given specific values for the clock speed of the computer. Their answers were expected to refer the values given. Some candidates need to take care in their use of terminology. Instruction was seen frequently instead of F-E cycle.

(ii) There were a small number of excellent answers to this question. Many candidates need to understand that when using multiple cores, each core has its own processor and that there is a need for communication between these individual processors. Answers such as '*there will be an additional load on **the** processor*' were seen frequently.

Question 6

(a) This question asked for the advantages of using an interpreter rather than using a compiler. Many of the responses simply described the operation of an interpreter without any comparison to a compiler. Candidates should understand that in questions like this, it is necessary include both translators in the answer.

(b) This question was generally not answered well. Many candidates need to improve their understanding of why some high-level languages are partially compiled and partially interpreted.

(c) (i) This question was generally answered well. The most popular correct answers were colour coding and the ability to expand/collapse code blocks. The question asked for features to support the visual presentation. A significant number of responses included features of an IDE which were not presentation features.

(ii) This question was generally answered well. The most popular correct answers were single stepping and breakpoints. A few responses given were not debugging features.

Question 7

(a) There were some excellent complete answers to this question describing various 3D printing techniques.

(b) This question was generally answered well. Many candidates were able to say something about overheating of either the printer or the material being used. Some candidates found it more challenging to describe the purpose of the sensor in greater detail required.

(c) This question was generally not answered well. Imprecise statements such as '*DRAM is cheaper*' or '*DRAM can store more data*' are not enough for credit. DRAM has a lower cost per unit and a higher bit density is more precise.

Question 8

(a) This question was generally not answered well. Many candidates need to improve their understanding of the purpose of each pass of a two-pass compiler.

(b) (i) The question stated that only instructions from the given table should be used and that each instruction should have a suitable operand. Many of the responses seen did not have any suitable operands and used instructions not given in the table provided.

(ii) This question required exact use of terminology; some candidates found this challenging. Statements for the similarity such as '*both load data*' and for the difference such as '*indexed addressing uses the index register, direct addressing does not*' are not precise enough at this level.

(iii) This question was generally answered well. Most candidates were able to correctly identify another mode of addressing.

(c) (i) This question was generally answered well. Many candidates were able to correctly carry out the given AND instruction.

(ii) This question was generally answered well. A small number of candidates shifted left instead of right.

(iii) This question was generally answered well. Many candidates were able to correctly carry out the given XOR instruction.

Question 9

(a) This question asked about the importance of feedback in a control system. Some candidates found this challenging and wrote about feedback in general everyday terms, describing verbal feedback on errors in computer applications and similar.

(b) Many candidates found this question challenging and described standard control systems or remote-controlled devices without explaining why there would be an embedded system involved. Some candidates described embedded systems, but in purely generic terms, without any reference to the example they gave.

COMPUTER SCIENCE

Paper 9618/12
Theory Fundamentals

Key messages

Candidate responses are expected to include more than simply general knowledge. There should be some demonstration of technical knowledge and appropriate use of the terminology associated with the subject.

Some candidates need to be careful with the use of terminology. For example, information was used frequently instead of data, and memory instead of storage.

Candidates should be encouraged to look carefully at the command words in the question. A question that begins with the word 'state' or 'identify' requires a different response to one that starts with the word 'explain'.

Some questions may ask for a justification of, for example, the use of application A rather than application B. This requires candidates to describe **why** application A would be used rather than application B. Simply describing **how** application A and application B are used does not answer the question.

General comments

Questions about logic gates and truth tables and utility software were usually completed successfully, the questions about networks and control systems were more challenging.

Comments on specific questions

Question 1

(a) This question was answered well. The most popular correct answer was an infra-red sensor.

(b) This question was not answered well. Many candidates need to improve their understanding of the role of an actuator in a control system. There was considerable confusion between the role of the microprocessor and the role of the actuator. Candidates should understand that the actuator does not do any processing of sensor readings. It initiates an action when prompted by the microprocessor.

(c) (i) There were some good answers to this question. Some candidates need to understand that statements such as '*it* (*an embedded system) is embedded into a larger system*' which simply re-word the description are not sufficient at this level. A better answer is '*it is integrated into a larger system*'.

(ii) The most popular correct answer to this question included a statement about the difficulty of upgrading the device so it was often thrown away. Some candidates must be careful not to repeat answers from previous part questions.

Question 2

(a) Answers for the primary key and referential integrity were much better than those given for an entity. Some candidates need to understand when describing the primary key that it is the record that is uniquely identified and not the table.

(b) The question asked for a description of methods other than authentication used by a DBMS to improve the security of a database. Some answers gave just a list of methods which is not enough for a description. Other responses included methods such as passwords or biometrics, both of which are forms of authentication. There was also considerable confusion between access rights and the provision of views.

(c) Many candidates were able to give correct generic statements about removing repeating groups of attributes and making all fields atomic. Applying this to the database given in the question proved to be more challenging. Many answers attempted full normalisation which was not required.

Question 3

(a) This question was generally answered well. Some candidates need to understand that statements such as '*a kibibyte is smaller than a megabyte*' are insufficient at this level. Care should be taken with the units. Some candidates wrote *bits* instead of *bytes*, and others omitted any units.

(b) (i) This question was generally answered well. Some candidates need to take care that when the question asks for 12-bit binary, any leading zeros or ones must be shown.

(ii) There were many correct answers to this question. Some candidates interpreted the binary value as an unsigned or signed integer rather than as Binary Coded Decimal (BCD) and gave incorrect answers of 2149 or – 1947. A small but significant group of candidates gave the incorrect answer of –865, presumably because the binary value had a 1 as the most significant bit.

(iii) This question was answered very well by most candidates. Some candidates should be more careful when adding the place values together.

(c) Most candidates were able to correctly identify an application of BCD. The most popular answer was some form of electronic display. Justifying the use of BCD in the given application proved to be more challenging. Imprecise statements about numbers rather than digits are not enough at this level.

Question 4

(a) This question was generally answered very well. Many candidates were able to write an appropriate Boolean expression.

(b) This question was generally answered very well. Many candidates were able to correctly complete the truth table.

Question 5

(a) This question was answered well. Most candidates were able to correctly state what was meant by privacy of data.

(b) This question was answered less well. Some candidates gave methods of ensuring the integrity of the data rather than stating what was meant by integrity of data.

(c) There were some excellent complete answers to this question. Some candidates need to improve their understanding of the difference between phishing and pharming.

Question 6

(a) This question asked for the justification of a method of compression in the context of the live stream of a music concert. Candidates were expected to state **why** a particular method would be chosen not **how** the chosen method works. Many candidates need to improve their responses to this type of question which requires the application of knowledge to a given situation or scenario. Answers about reducing the download time were not applicable to this scenario; the duration of the streaming would be the duration of the concert.

(b) The question asked candidates to explain the impact of changing the sampling resolution on the accuracy of a sound recording. Many responses included statements about file size, which did not answer the question. Some candidates need to understand that it needs to be made clear in their answers whether the sample resolution is being increased or decreased. Imprecise statements such as *'increasing the sampling resolution improves the accuracy of the recording'* which simply statement given in the question are not enough. There needs to be an explanation of how the accuracy is improved.

(c) This question was generally not answered well. Many candidates need to understand that the bit depth is given in **bits** and the answer is required in (mebi)**bytes**, so there is a requirement to divide by 8 in the working. The most popular incorrect answer was 20 where this division had been omitted. Some candidates should also be aware that in questions of this kind, it is unlikely that complicated calculations will be required. The values given will usually cancel out and leave a straightforward calculation for the answer.

Question 7

(a) There were some excellent answers to this question. Some candidates need to understand that there are eight groups of digits in IPv6 not six. There was also considerable confusion between a semi-colon (;) and a colon (:) as the separator between the groups of digits.

(b) (i) This question stated that there were four devices in the network and required a diagram of the network configured as a star topology. Many of the diagrams seen incorrectly included other devices. There was also some confusion between a star topology and a mesh topology.

(ii) Candidates with a suitable central device were able to correctly explain how data would be transmitted between the two laptops. Some candidates need to improve their understanding of the role of the router in a network.

(iii) This question was generally not answered well. There were some vague answers about transmission speed and collisions, which did not really answer the question why subnetting is used. Better answers included statements about improved security or more efficient management.

(c) This question was generally answered well. Some candidates need to understand that it is the devices, as stated in the question, that are performing the actions and not the CSMA/CD protocol.

(d) Many candidates were able to correctly identify the public and private IP addresses. Describing static and dynamic IP addresses proved to be more challenging, especially the dynamic IP address. Answers such as '*an IP address that randomly changes*' are not enough. There needs to be a statement about when or why it changes.

Question 8

(a) This question was generally not answered well. Many of the answers stated a management task performed by the Operating System rather than a purpose of the Operating System.

(b) Many candidates were able to correctly identify appropriate utility software. The most popular choice was disk defragmentation. Some candidates need to read the question more carefully. It asked for utility software that was **not** intended to improve security. Incorrect answers seen included software that is intended for security such as virus checkers and firewalls. Some candidates who correctly identified appropriate software, found it more challenging to explain why this software would be needed.

(c) (i) This question was generally answered very well. Most candidates were able to correctly name an appropriate port to provide a connection for the optical disc reader/writer.

(ii) Many candidates found this question very challenging. It asked for the roles of the components in the process of writing to the optical disc reader/writer. Few candidates were able to successfully apply their answers to the given scenario. There were many generic statements about the components, which did not answer the question.

Question 9

(a) There were some good answers to this question. The opcode most likely to be incorrectly placed was JPE; identified in the compare group rather than the unconditional and conditional group. Some candidates should understand that just copying the explanation of the opcode from the table does not identify the instruction group for the command.

(b) There were many excellent complete answers to this question. Some candidates with an otherwise correct trace table incorrectly included quotation marks around the output. Some candidates started correctly but then struggled with implementing the code from the LDX command onwards.

COMPUTER SCIENCE

Paper 9618/13
Theory Fundamentals

Key messages

Candidate responses are expected to include more than simply general knowledge. There should be some demonstration of technical knowledge and appropriate use of the terminology associated with the subject.

Some candidates need to be careful with the use of terminology. For example, information was used frequently instead of data, and memory instead of storage.

Candidates should be encouraged to look carefully at the command words in the question. A question that begins with the word ‘state’ or ‘identify’ requires a different response to one that starts with the word ‘explain’.

Some questions may ask for a justification of, for example, the use of application A rather than application B. This requires candidates to describe **why** application A would be used rather than application B. Simply describing **how** application A and application B are used does not answer the question.

General comments

Questions about logic gates and truth tables and representation of data were usually completed successfully. Questions about using SQL and types of RAM proved more challenging.

Comments on specific questions

Question 1

(a) Candidates found this question challenging. Answers such as ‘*data that is not digital*’ were too vague at this level of study.

(b) Most candidates were able to correctly match each term to its corresponding description.

Question 2

(a) This question asked candidates to describe the impact of increasing the image resolution on the quality of a bitmap graphic. Many responses included statements about file size, which did not answer the question. There was considerable confusion between image resolution and sampling resolution.

(b) Many candidates need to understand that the bit depth is given in **bits** and the answer is required in (kibi)**bytes**. Therefore, there is a requirement to divide by 8 in the working. A frequent incorrect answer omitted this division.

Question 3

(a) (i) There were some good answers to this question. The most popular reason given was the requirement for the courses to be available over the internet for access anywhere in the world.

(ii) Candidates found it more challenging to describe two disadvantages of using a public cloud. Many of the responses applied equally to the LAN. There seemed to be a general misconception that because the files were on a public cloud, they were accessible to anyone and that a public cloud was owned by the public not a third-party supplier.

(iii) There were many good answers for the firewall and passwords. Many candidates found it more challenging to state how encryption protected computer systems. Some candidates should understand that encryption does not prevent access to the data, the encryption makes the data incomprehensible if accessed.

(b) (i) This was a standard breakdown of a many-to-many relationship. Many candidates recognised it as such and correctly completed the entity-relationship diagram.

(ii) There were several answers where the basic `SELECT`, `FROM` and `WHERE` clauses were correct. Many candidates found it more challenging to give an appropriate field name to the value returned.

(c) There were some excellent answers to this question with complete descriptions of appropriate verification checks. There was also some confusion between verification and validation. Some candidates need to improve their understanding of the difference between the two.

Question 4

(a) This question was answered well by most candidates. Some candidates need to improve their understanding of the operation of an XOR gate.

(b) This question was answered very well by many candidates. Many candidates were able to correctly draw an appropriate logic circuit.

Question 5

(a) Some candidates found this question challenging. Descriptions of single stepping were often better than descriptions of context-sensitive prompts. There was some confusion between context-sensitive prompts and auto-completion.

(b) (i) This question required candidates to explain the reasons **why** the files would be compressed and not **how** a particular compression method works. Many candidates need to improve their responses to this type of question which requires the application of knowledge to a given situation or scenario.

(ii) This question required candidates to describe the reasons **why** a hard disk formatter is needed and not **how** the utility software works. While there were some better answers, there was also considerable confusion with disk defragmentation software.

Question 6

(a) The question required two benefits to a programmer. Some candidates need to ensure that they read the question carefully; responses were often given in terms of benefits to the user. There was considerable confusion with open source software. Responses describing modification and re-distribution were frequently seen. Some candidates need to understand that shareware does not mean that the source code can be shared.

(b) There were some very good answers to this question. Most candidates could give at least one valid reason why a programmer should join a professional ethical body.

Question 7

(a) This question was answered well. Some candidates need to understand that inexact responses such as '*static RAM is faster*' are not enough at this level. Better statements include '*static RAM has a faster access time*'.

(b) Many candidates found this question challenging. Responses were often given in the context of a comparison with EPROM which was not what the question asked. The most popular correct answer was the ability to change the contents of the EEPROM without removing it from the headset.

(c) This question was generally not answered well. There was considerable confusion between the buffer and the print queue and statements about instructions being transferred to the buffer rather than the data to be printed. Some candidates described the principal operations of a laser printer with no mention of a buffer.

(d) Most candidates were able to correctly identify a suitable port. Many candidates found it more challenging to justify why this port would be appropriate. A justification should say **why** the port is used and not how the port operates.

Question 8

(a) This question was answered well by most candidates. Some candidates need to ensure that their descriptions of a validation check do more than just repeat the name of the check. For example, responses which name a validation check as a format check, and then give the reason for use as to check that the data is in the correct format, will not gain credit for both the name and the description.

(b) This question was answered well by most candidates. Many candidates were able to give a difference and a similarity between pharming and phishing. Some candidates need to understand that at this level answers should contain more than just general knowledge and that when describing a difference between two things, a statement about both is required.

(c) This question was also answered well by most candidates. Some candidates must ensure that they take care with the precision of their answers. For example, just downloading an anti-virus program is not enough; it must be clear that the anti-virus program is installed and running.

Question 9

(a) (i) There were some excellent answers to this question. Some candidates need to read the question carefully, as answers that included buses and registers were seen when candidates were told not to include them in their answers.

(ii) Some candidates found this question more challenging. Responses such as '*special purpose registers have a special use; general purpose registers are for general use*' which simply return the names of the registers are not enough at this level.

(b) The missing steps in the table were often inserted correctly. The missing descriptions proved to be more challenging. A frequent incorrect statement for the first missing description was '*the contents of the MAR are copied to the MDR*' rather than '*the contents of the address held in the MAR are copied to the MDR*'.

(c) This question was answered well by most candidates. The most popular correct answer was division by zero.

(d) A minority of candidates provided a completely correct answer. A popular incorrect order for the missing statements was D-A-E.

COMPUTER SCIENCE

Paper 9618/21
Fundamental Problem-solving and Programming Skills

Key messages

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

- A correct procedure/function header but the end statement is omitted.
- An IF statement with the `ENDIF` omitted.
- A loop with the terminator statement omitted; for example, `ENDWHILE`.
- File handing statements incorrect:
    - The file name omitted from the `CLOSEFILE` statement.
    - The file name omitted from the `EOF()` function (**Question 8(a)**)
- Using keywords for a variable identifier. For example, String (for the parameter in **Question 6(a)**).
- `OUTPUT` statements where the <ampersand> or <comma> have been incorrectly used or omitted.

General

In English the verb ‘to increment’ is used to mean to increase some value and it can be by one only or a stated number e.g. ‘an increment of 5’. In Computer Science we always assume the increment is by one. For example, increment `TopOfStack` (**Question 3(b)**).

Candidates seem unsure when the algorithm is asking for the calculation of ‘a total of …’ and have trouble distinguishing this from ‘a count of …’. **Question 4** required the algorithm to input a sequence of integer values and calculate a running total of the numbers which satisfied some condition. Candidates need to be aware this is a very different requirement to an algorithm asking for a count of the numbers that satisfied some condition, as asked for in **Question 4(a)**.

‘Empty string’ does not mean a string populated with <space> characters.

A statement such as 30 to 40 (**Question 1**) or ‘between 30 and 70 inclusive’ (used in **Question 2**) would always assume to include the two boundary values.

Question 1

(a) A common error was for the delimiter speech marks to be omitted from row 1 and row 3. Some answers for row 2 were shown as 12/3 which did not gain credit. Another frequent error on row 4 was to state the answer as 10. A general point candidates should be aware that a statement such as ‘30 to 40’ does include the two boundary values.

(b) Some answers lost the first available mark by not stating the relevant assignment statement. Many correct answers however successfully followed. Candidates stating that if the level was less than 20 then it would satisfy the first clause in the case structure. It was expected that the answer clearly stated, or implied, that the clauses in a case structure are processed in sequence.

(c) The majority of answers gained the mark by stating that all possible values for the Level variable are already covered by the existing clauses.

(d) Well answered with the major of candidates gaining all full three marks. Candidates should be reminded that if an answer such as 'integer/real' is presented only the first answer shown will be marked.

Question 2

(a) The standard of answers seen were very variable. See the earlier comment in the General section concerning the interpretation of the question wording '*a total of all numbers …'.*

The answers which gained 4 or all 5 marks were describing something happening to a single number.

The key wording of the question was '*steps …. which could be used to produce pseudocode*'. The key steps were:

- initialise a running total to zero
- the algorithm would need a loop to iterate 100 times
- a number is input
- the number is compared to the given criteria, and if true …
- the number is added to the running total
- when the loop ends the final total is output.

Answers which throughout described something happening to numbers scored few if any of the available five marks. The use of an array to store the numbers was a common irrelevance.

(b) Generally, well answered, by stating the constructs of iteration and selection. Peripheral terms such as 'repetition' and 'condition(al)' did not gain credit.

Question 3

(a) There were very few answers which scored the three available marks. The first bullet point in the stem of the question meant there was no ambiguity as to the five array cells which would contain the values. Some answers changed the order of the data so that they were in a different sequence (often (D1, D2 …D5). A very common misunderstanding was what values would be used for the two pointers. These were frequently incorrectly shown as two of the data values (for example, D1 and D2) rather that the array index values (correctly as 5 and 1). This was a fundamental error which suggested a lack of understanding of how a stack structure using an array would be implemented.

(b) This question assessed the candidate's understanding of the terminology used using an array to implement the stack structure. Weaker answers tended to gain only the final mark – however many strong answers were seen.

Question 4

(a) Generally, well understood and answered. Most solutions correctly had a count-controlled loop, but a not uncommon error was the omission of the assignment arrow. Some solutions attempted a length calculation/check on the string parameter which was not required.

A correct solution had the return value done either inside the loop when the count condition was met, or outside the loop. A common error for the latter was to omit the final `ENDIF`.

(b) There were many variants seen here producing a correct solution. Some candidates used a two-iteration loop to test each of the array values. This was not needed, and a simpler solution used two conditions joined by the `AND` operator.

However, a common syntax structure error was to write this incorrect statement:

```
Search = Data(Row, 1) OR Data(Row, 2)
```

Many candidates used the correct syntax for the test for an even number. Careless errors which lost marks were:

- not using the variable `Row`.
- not using the identifier name `Data` for the array. (Both as stated in the rubric of the question).

Question 5

(a) Generally, well answered with the majority of the answers seen identifying at least two of the incorrect statements.

Candidates were given no guidance as to how to present their answer. The clearest answers were those which wrote both the original incorrect statement, followed by the correct (both labelled).

Weaker answers which only made a statement such as '*replace the* < *with a* <', '*replace the and with a* +' generally did gain credit but this was not ideal or good examination technique.

The statement which proved most elusive for candidates was the correction to:

```
NextInput = "END"
```

(b) (i) This was poorly answered. Answers which were able to state that a sequence with integers all greater than 999 were rarely seen.

(ii) Some candidates' mis-interpreted the framework given on the question paper for their answer and alongside each value prompt had a complete sequence.

Very few answers were seen which gained any credit. The sequence needed to have all integers above 999, at least one non-numeric value (as per question rubric) and the final '`END`'. Some candidates wanted to explain a solution where the non-numeric values were somehow converted to integers (despite the wording on the question stem that stated '*….non-numeric input is ignored…..')*.

Question 6

(a) Most candidates found this a challenging question and misunderstood the given problem description; few answers were seen which gained the full 7 marks. The most common algorithm error was to concatenate the string using the old `MyString` variable. This meant that there was no longer access to the original string variable if the length of the concatenated string exceeded 255 characters. Furthermore, most candidates forgot to reset `MyString` to an empty string if the Boolean parameter evaluated to true.

Additionally, some candidates made the mistake of redeclaring variables and their `INPUT` inside the procedure instead of passing them as parameters.

(b) These marks were only gained by the strongest candidates. Some answers gave a clear description of the difference between global and local variables, but this did not gain credit. The answer required a description of how this change would impact on this algorithm. The answer wanted was a statement that, as each new instance of `MyString` is created, this would result in the variable's existing contents being lost.

Question 7

The majority of answers gained the full five marks. Only errors seen were the omission of the self-transition for State S5 or a state which was duplicated. The omission of the arrows for each event would have been penalised but was rarely seen.

Question 8

(a) The range in quality of answers seen was wide; weak answers often did not get past a correct procedure header.

Refer back to the 'Key Message' section. There were errors which normally resulted in a loss of marks.

- The omission of a loop.
- A correct `READFILE` statement, but no `CLOSEFILE`.
- The omission of the filename from the `EOF()` function.
- The use of the '`+`' operator to concatenate strings.

The algorithm was not a complex one although it used a scenario of data transfer which had not been used in previous series. Candidates often could not correctly form the message string which was to be sent using the `Transmit()` module.

`<STX>` and `<ETX>` were often incorrectly treated as strings, each of three characters. Solutions which correctly formed the string used for the final file transmit were rarely seen.

(b) Few answers gained the two marks. A statement that 'this string is the indicator to the receiving computer that there is no more data to be sent or it is the end of the transmission. The receiving computer would then close the file' would have secured the marks.

(c) (i) Answers here were weak. A statement that the message cannot be a zero-length data field, therefore blank lines cannot be sent, would have secured the two marks.

(ii) This question proved to be challenging for the majority of candidates. They were required to identify ways to transmit a blank line and recognise the new format by the receiving computer. Some candidates suggested the use of special characters or additional fields, but answers failed to describe how the receiving computer would then reconstruct the original message. In many answers it was not clear whether a suggested change was being made to the contents of the message part only of the transmission, or a change to the format of the protocol.

(d) A very varied standard of answers seen with weaker answers sometimes incorrectly used an unnecessary loop. A sequence of `IF` statements or a `CASE` structure were equally appropriate. For field numbers 1 and 2 the `MID` function was correctly used but often its parameters were incorrect; likewise, for field number 3. Answers often showed a sense of 'recovery' where – despite failing to gain earlier marks – the structure was correct for both the return of the value and the exception for an invalid parameter. Please see the comment in the General section relating to an empty string.

COMPUTER SCIENCE

Paper 9618/22
Fundamental Problem-solving and Programming Skills

Key messages

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

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

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

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

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

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

General comments

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.

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

- Invalid variable names:
  ```
  Length ← LENGTH(Suffix)
  ```
  Using standard function identifiers, found in the Insert, as variable names.

- Invalid string concatenation:
  ```
  Suffix ← '0' + Suffix
  ```

- Invalid boolean statements:
  ```
  IF DNum >= RouteTable(Index, 1) AND <= RouteTable(Index, 2) THEN
  IF RouteTable(Index, 1) <= DNum <= RouteTable(Index, 2) THEN
  ```

- Omitting the assignment in a loop:
  ```
  FOR X 1 TO NunFiles
  ```

- A language specific statement:
  ```
  FOR Index in RANGE(100)
  ```

<u>Comments on specific questions</u>

It is recommended that the following specific comments be read in conjunction with the published mark scheme for this paper. The initials 'MP' stand for 'mark point'.

Question 1

(a) Most of the candidates gaining all four marks available for this question.

One of the most frequent errors was the use of a less than meaningful variable name. A typical example of this was simply using `name` for the name of the customer.

(b) Many three-mark or four-mark answers were seen with over half gaining all four marks available for this question.

Missing quotes accounted for many lost marks and 'error' was seen occasionally despite not being signposted in the question.

Several candidates took `Description` as a string rather than an identifier so row two was evaluated to `'ription'` and row three to `FALSE`.

(c) A more demanding question compared to **1(c)** and **1(b)** with fewer candidates gaining all three marks for this question.

The majority of candidates seemed to recognise the need for a Record for the first mark, generally followed by a reference to 'store many datatypes' for a second. Reference to an array of records was not quite as common but still seen fairly frequently.

Common mistakes included 'database' and simply 'file'. Another common incorrect answer was 'an array' which by itself was not mark worthy.

Question 2

(a) This question tested the candidate's ability to create a flowchart to represent an algorithm.

A full range of marks were seen for this question with most candidates gaining three marks or less. Many candidates were able to correctly create a flowchart that checked for the entry of the first 27 or summed the remaining inputs until a 0 entered but not both. Many candidates did not gain the initialisation mark for the sum.

(b) This is a commonly asked question where candidates are asked to name the type of loop used in an algorithm. Candidates need to use the correct terminology from the syllabus when describing a loop, which in this case was either a 'post-condition' loop or a 'pre-condition' loop. Many candidates lost a mark for just stating 'repeat' loop or 'while' loop which was not enough to gain a mark.

The justification mark was often awarded when the name of the loop was not enough. Candidates often gave mark worthy answers using a specific example of a termination condition from the question. The generic answer of 'unknown number of iterations' was also awarded a mark.

Question 3

This question required candidates to understand how a linked list is implemented using two arrays.

(a) Most candidates obtained some marks for this question with only a few gaining full marks.

Candidates were expected to complete all the missing values not just some.

Most candidates failed to give the `Start_Pointer` value although this was the first missing value in the question.

The Index 2 mark was the most frequently given, followed by the one for Index 5.

The two 'null pointer' marks seemed the most challenging, with the second (Index 7 and 8) being given more than the first (Index 3 and 4).

(b) This question assessed the candidates understanding of how a new item is added to a linked. Candidates are not expected to create their own code from scratch to do this but should be able to complete the steps required when structured in the way presented in the question.

Candidates generally found this more challenging than **Question 3(a)** with fewer gaining any marks and only a few gaining all fours marks available.

A number of zero-mark answer were seen along with the use of non-linked list terminology e.g. head and tail pointers.

Question 4

(a) A good spread of marks was seen with many candidates gaining all the six marks available.
MP1: awarded frequently although several solutions passed in `ThisNum`. A number of candidates still miss out on this mark through omitting the `END` statement.

MP2: Seen frequently.

MP3: Most solutions recognised the need for some kind of loop. Many `WHILE` loops referenced a `ThisNum` that had no initial value. Rather than a simple `ThisNum <> 99` condition a number of solutions opted for an incorrect `Num < 99 AND Num > 0` (the question does not say that 99 is the max value).

MP4: Often input just before the loop and then again as the first line inside the loop. Many solutions omitted it altogether.

MP5: Correct use of `MOD` was often seen, although per cent was seen on many occasions (which was not mark worthy). The `MOD` operator is included in the Insert.

Several elaborate solutions attempted to strip off the rightmost character and compared it with 0, 2, 4, 6 and 8, often incorrectly:

e.g. `IF RIGHT(ThisNum,1) = 0 OR 2 OR 4...`

Some candidates incorrectly used conditions such as `IF Num = Even`

MP6: Many missed the requirement to have initialised both count variables and to **not** count the value 99. Some solutions correctly added an IF statement to skip the value 99.

MP7: This mark was often lost for an attempt to concatenate an integer by simply using `&` operator. Comma separators were correctly used by most candidates.

A few candidates lost this mark for not including a comma separator
e.g. `OUTPUT "Number of even numbers" EvenCount`

(b) The term 'test data sequence' is not widely understood.

Most solutions omitted the terminator value at the end, although many answers went on to suggest a markworthy sequences with 99 as the only value or 99 included somewhere in the sequence.

A common mistake was using test data that was too similar to the example given in the question.

Question 5

Many good answers were seen, although some of these answers missed the initial values for MP1.

Most candidates gained at least three marks but there were a number of zero-mark or one-mark solutions seen.

It is important when completing trace tables that candidates keep the values of variables distinct for each iteration of the loop. A number of candidates failed to do this.

Question 6

(a) As is usual with this type of question, a wide range of marks was given.

It was pleasing to see a range of different solutions gaining full marks for this question.

MP1: Many solutions gained this mark. In some cases, the solution consisted of little more than the module 'wrapper' suggesting that this is a recognised tactic.

MP2: Most solutions included some form of attempted loop. The simple `FOR...NEXT` loop was often replaced by a `REPEAT...UNTIL` despite the added requirement that this approach needed a statement to increment the loop counter. A significant number of solutions mistakenly looped from 1 to 999.

MP3: Many solutions gained this mark through the use of `NUM_TO_STR(LoopCounter)`

MP4: Few solutions gained this mark. Often the dot was absent. Some ingenious solutions attempted to divide the file number by 1000 (so file 3 became file 0.003) but then failed to convert the result to a string and take just the rightmost 4 characters.

MP5: Very many instances of `CLOSEFILE` appeared without a filename. The use of `CREATEFILE` was common although this is not mentioned in the Pseudocode Guide for Teachers.

MP6: This mark was often gained as a follow through. A very common mistake was for the `WRITEFILE` command to have just one parameter (rather than `<filename>,<data>`) and this was exacerbated by the frequent use of a comma instead of the correct `&` operator to concatenate the filename with the suffix.

(b) (i) Most candidates gained the one mark for this question but over a third did not gain this mark. Incorrect answers were drawn from general computing terminology.

(ii) Under half the candidates answered this question correctly. Mistakes seen occasionally included providing a return identifier in addition to the data type and providing an integer parameter in addition to the string.

(iii) As for **6 (b)(i)** most candidates gained the one mark for this question but over a third did not gain this mark. A few candidates incorrectly gave `WRITE` or `APPEND` as the file mode.

Question 7

(a) The question required candidates to complete a structure chart. A small number of candidates gained all four marks for this question.

MP1 and MP3 were the mark points most often given. Common mistakes were omitting parameter labels particularly on the lower level and not shading in the Boolean parameter arrow to Module-Y.

(b) Most candidates correctly identified the use of selection so gained one mark for this question. A few candidates correctly named all four modules involved.

Question 8

(a) A full range of marks was given for this question a significant number of scripts gaining full marks was seen. However, many non-viable attempts were seen.

It was pleasing to see a range of different solutions gaining good marks for this question. Two of the common approaches gaining full marks were those included in the mark scheme as example solutions. By far the most common was the looped based solution.

MP1: Although not vastly different to MP1 in **Question (6)(a)**, this corresponding mark was less often given and where only one mark was awarded this was generally the mark given.

MP2: Not seen often but candidates who failed to get this mark could still gain the rest of the marks available.

MP3: Many solutions rejected the straightforward `FOR...NEXT` loop in place of a more sophisticated conditional loop. Unfortunately, the termination condition was often incorrect (e.g. an incorrect number of iterations) or the loop lacked the counter increment statement or the values in the conditional test had not been initialised. Even a `REPEAT...UNTIL` loop that includes the loop counter increment fails to gain the mark if that counter has not been initialised.

Some solutions included an inner iteration loop for `Col`, which was usually unsuccessful due to the need to implement a logical `OR` of the two comparison results.

MP4: Not addressed often. Occasional attempts at testing `DestinationID` rather than `RouteTable(Row, 1)` were seen.

MP5: Many solutions gained this 'attempt' mark. A common mistake was to use `=` rather than `<=` or even just `<`.

A significant number of solutions attempted to use the `RouteTable` values from the example in the question which was not mark worthy.

An occasional fault was to include an `ELSE` clause that immediately returned –1 if the `DestinationID` was not found in the range. This impacted on MP8.

MP6: The usual reason for this not being given following MP4 was that the solution missed the final value in each range e.g. `<` instead of `<=`

MP7: Many solutions gained this mark. Immediate `RETURN` of the value was often seen. Many solutions attempted to terminate the loop when a matching range was found although this was not considered necessary for such a small number of iterations.

MP8: Usually given where a reasonable attempt had been made, except where an `ELSE` clause had been used (see MP5 above).

(b) A few very good solutions were seen for question. Some weaker solutions included the use of an unnecessary loop.

MP1: Not well addressed. Solutions that did attempt this often simply compared the whole message with an empty string. Solutions that attempted to extract a possibly non-existent substring did not gain this mark
e.g. `IF MID(Message, 4, LENGTH(Message) – 3)`

MP2 and MP3: These mark points were the ones most often awarded.

MP4 and MP5: Some very elegant solutions were seen but the majority of candidates struggled with the use of this user-defined function. A common mistake was not assigning the return value. Another common mistake was not including either one or both of the parameters.

MP6: This was attempted in many solutions. Often it included a further use of the function `StackMsg()` with no parameters which was not condoned.

MP7: Frequently given, even when MP6 was not awarded as long as the attempt at this this mark point was reasonable.

(c) (i) Many incorrect solutions referenced a whole range of possible problems, such as ‘run-time error’, ‘network problem’ and ‘using the wrong stack’.

Scenario one and two were the most popular, with marks commonly given for ‘FILO’ and ‘Stack is full’.

For scenario two, the most common suggestion was that ‘all the lines are reversed’ but some candidates answers did hint at the interaction between the PUSH and POP actions resulting in ‘some of the lines being out of sequence’.

Very few three-mark solutions were seen as the final mark, in all cases, wanted a reference to the received file and this was usually lacking.

(ii) Just under two-thirds of the candidates gained the mark for this question.

COMPUTER SCIENCE

Paper 9618/23
Fundamental Problem-solving and Programming Skills

Key messages

Many marks are lost by candidates who do not pay attention to key points of detail when writing pseudocode statements. The list is extensive but some of the more common errors seen – which would all result in a loss of marks – are listed below:

- A correct procedure/function header but the end statement is omitted.
- An `IF` statement with the `ENDIF` omitted.
- A loop with the terminator statement omitted; for example, `ENDWHILE`.
- File handing statement incorrect:
    - The file name omitted from the `CLOSEFILE` statement.
- Using keywords for a variable identifier. For example, Length (**Question 8(a)**).
- `OUTPUT` statements where the <ampersand> or <comma> have been incorrectly used or omitted.

General

In English the verb ‘to increment’ is used to mean to increase some value and it can be by one only or a stated number e.g. ‘an increment of 5’. In Computer Science we always assume the increment is by one. For example, increment `NumItems` (**Question 3(b)**).

Candidates seem unsure when the algorithm is asking for some total and have trouble distinguishing this from ‘a count of …’. **Question 8(a)** required the algorithm to read successive lines from the file and maintain a character count of the numbers of characters read. Another way of expressing the requirement is to calculate the ‘running total’ for the characters read.

Question 1

(a) Generally, well answered by the majority of candidates.

(b) Again, well answered.

(c) The majority of candidates suggested the programmer would look for the required functions in a program library. Answers which did not gain credit stated vaguely that the functions would be found ‘*from a search of the internet*’.

(d) Well answered with candidates stating this would require adaptive maintenance. This was followed for the second mark with the suggestion that the user requirements had changed which would also require adaptive maintenance. Some answers tried to expand on the question scenario of ‘*new hardware*’ and this did not gain credit.

Question 2

(a) Very few clear and correct solutions were seen. Candidates often gained credit for a solution with a loop which iterated 30 times, but with little else correct. The algorithm required the tracking of the current minimum value and its index position; the second part was often not present. There were many basic errors which were frequently seen all of which would have resulted in a loss of marks:

- The incorrect flowchart symbol used for a decision box.

- Failure to have two branches from a decision box.
- Failure to label the branches True/False.

Some candidates were clearly thinking about a bubble sort algorithm and a comparison was made between elements i and i+1 in the array.

(b) Most candidates were able to score at least one of the available marks. The most popular answers were that an array allows for all data items to be referenced by a single identifier name or that the arrays can be iterated through using a loop and the index value for each element. There was a more accessible mark available for stating that using an array should simplify the algorithm and make the solution easier to understand, but this was often not stated in the answer.

(c) Generally, well answered.

Question 3

(a) This whole area of the syllabus for abstract data types is one which candidates find challenging and the standard of answers seen reflected this. The data items could appear in five consecutive locations in the given order. Many candidates wrongly decided to change the order to D1, D2, ….D5. A more fundamental error demonstrated a basic misunderstanding about how a queue structure is implemented. Only the stronger candidates showed the two pointers as an array index number. All too frequently the pointers were incorrectly labelled with two data items.

(b) This was assessing the candidate's understanding of the terminology associated with arrays. A range of responses was seen including many which secured four or five of the available marks. See the comments made earlier in the 'General' section of the report concerning the use of the word 'increment'.

Question 4

(a) A straightforward algorithm requiring a count-controlled loop iterating 25 times. Most candidates used successfully the `RAND` and `INT` functions to generate an integer number. There were two different approaches to ensure that each subsequent number generated was larger than the previous. The first method used a 'step value' (say 10) as the parameter for the `RAND` function, and then added this number (after conversion to an integer) to a base number; the base number therefore increasing as each new number was generated.

A different approach was to generate the next number (and repeat doing this) until the number was for the first time larger than the previous. Candidates successfully implemented this with a `WHILE … ENDWHILE` loop.

Refer back to the 'Key messages' section. Three of the available six marks here required correct detail; for the procedure heading and ending, a correctly formed loop and a declared local loop counter variable.

(b) Well answered with solutions eliminating the need for the `OR` operator by combining the two clauses using a `<=` or `>=` comparison.

Question 5

Very few solutions were seen which gained the full six marks. Most candidates gained the first mark for the initialisation stage but often then quickly lost their way with the trace.

Question 6

Most candidates secured the first mark for a correctly formed function header and ending. The easiest mark seemed then to be the test for a match between the right three and left three digits (return value 3). Strong answers did this test first, then followed by the test for the rightmost three digits matching with '000'. This then addressed the requirement in the question stem that 'the function will return the highest possible value for the given string'. The final test (return value 1) proved trickier. The error was often an incorrectly formed `IF` statement which required a test for digit 4 equal to digit 5 `AND` digit 5 equal to digit 6.

Question 7

(a) Candidates answers often correctly stated that this symbol indicated iteration, but then failed to follow this with the detail of the sequence in which the three modules would be called.

(b) (i) Generally well answered with candidates demonstrating the correct syntax for the definition a record data type with identifier `MyType`.

(b) (ii) The inclusion of `BYREF` to define the parameter was rarely seen.

Question 8

(a) Candidates seemed to have no problem understanding the scenario of 'communications' used for the question. The question contained some familiar concepts; file handling, maintaining a count of the number of characters read and a conditional loop testing for the message containing the terminator string. Some candidates did not understand the use of the `STX` and `ETX` fields and wrongly assumed these were each a three-character string.

Refer back to the 'Key messages' section. There were many basic syntax errors made which lost marks – including the use of `Length` as a variable identifier, and incorrect syntax in forming the output statement.

(b) Generally weak answers seen here. The starting point should have been that there is an issue if one of the data fields being sent is indeed the terminator string; the consequence of this is then that the file transfer will terminate prematurely.

(c) Some strong answers were seen for what was a challenging question. Once the answer had the basic structure of the conditional loop, there were some accessible marks to be had – the use of the `GetData()` function to receive the message, its output and the input by the receiver of a reply. Weaker answers often had the basic design but were unable to form a completely correct string to be transmitted with the `Transmit()` function.

(d) Candidates were able to state that the main issue was what would happen if the receiver failed to make a response. Stronger answers were often inventive suggesting solutions such as introducing a timeout if no response was received.

COMPUTER SCIENCE

Paper 9618/33
Advanced Theory

Key messages

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

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

General comments

Candidates are advised to read questions carefully before beginning their answer in order to understand what is being asked of them 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. Marks could be awarded for how a given answer applies to the given scenario.

Comments on specific questions

Question 1

(a) This question was usually well answered, with most candidates achieving at least one mark and many achieving all three for a fully correct solution showing appropriate working.

(b) Most candidates achieved at least one mark, but answers were often not detailed enough for both marks. Candidates generally recognised that not all fractions could be represented exactly in binary or that numbers may lose precision due to rounding.

Question 2

(a) Candidates generally scored high marks on this question, linking each protocol to its most appropriate use. However, not all candidates achieved full marks.

(b) A high proportion of candidates achieved at least one mark for outlining the purpose of the link layer in the TCP/IP protocol suite, with many of these candidates achieving both marks.

Question 3

Candidates generally recognised that both the enumerated and pointer data types are user-defined non-composite data types. They also often knew that an enumerated data type includes an ordered list of all possible values. Candidates who noted that pointer data types stored or referred to a memory address achieved another mark.

Candidates are advised not to use the name of a data type to describe it, for example, a pointer data type points to a memory location is not enough as it does not show understanding of the words ‘point’ or ‘pointer’.

Question 4

(a) Candidates usually demonstrated understanding of the sequential and random file organisation methods. Candidates who correctly wrote about how records are organised within these methods achieved the highest marks. The full range of marks was seen with many candidates achieving high marks. However, some candidates incorrectly wrote about how files are organised rather than how records were organised within files.

(b) A good range of marks was seen, with candidates who described how a record was found by searching linearly from the beginning of the file until the record was found achieving the best marks.

Question 5

Candidates were generally aware of the meaning of SISD and MIMD in terms of computer architectures. A high proportion of candidates were also aware that an SISD computer contains a single processor. However, relatively few candidates expanded their MIMD description sufficiently. Those who did, stated that an MIMD architecture contains many processors that operate independently.

Question 6

(a) A high proportion of candidates correctly completed the truth table for a logic circuit that included a 3-input NAND gate.

(b) Candidates were required to simplify a given expression using the rules of Boolean algebra. This question was answered well by many candidates, with many achieving full marks. Many of the remaining candidates achieved at least one mark for showing some correct working.

Question 7

(a) Candidates who stated a benefit of a user interface, such as that the user interface hides the complexities of the computer hardware from the user, followed up with an example, such as clicking on an icon instead of writing code.

(b) The vast majority of candidates correctly identified blocked state as an additional process state.

(c) Most candidates achieved at least one mark here, usually for stating that a process could change from the running state to the ready state when its current time slice expires, or by describing the impact of the CPU receiving an interrupt.

Question 8

(a) This question required candidates to complete the missing blanks in a pseudocode algorithm which finds a record in a random file and outputs it. Many candidates found this question difficult, so it received a mixed set of responses.

(b) Candidates who defined ‘exception handing’ as a process of responding to an event in a running program so it does not halt unexpectedly, or similar, achieved the mark. A number of candidates incorrectly described the meaning of the term ‘exception’, instead.

(c) Most candidates were able to state at least one example of a cause of an exception, with many of these candidates stating two causes, for example, runtime error, or a specific error, such as division by zero.

Question 9

(a) (i) A large proportion of candidates achieved at least one mark for writing the infix version of a Reverse Polish Notation (RPN) expression. A common error was for candidates to leave out the brackets in their infix expression, therefore making their solution incorrect.

(ii) This question was generally answered well, with many candidates achieving high marks for showing the changing contents of a stack whilst evaluating an RPN expression.

(b) Most candidates were able to explain how a stack can be used to evaluate RPN expressions. The best answers explained the RPN evaluation process in a manner that could be applied to any RPN expression.

Question 10

(a) Candidates had to write the pseudocode to declare a constant, three variables and an array to be used in an algorithm representing a stack. Candidates who used the pseudocode style defined in the pseudocode guide that accompanies this course achieved the best marks.

(b) This question required candidates to complete the missing blanks in a pseudocode algorithm for a function to pop an element from a stack. A mixed range of results was seen with a significant number of candidates achieving high marks.

(c) Candidates were asked to compare and contrast the queue and stack Abstract Data Types (ADT) which meant that both types of ADT needed to appear in each statement for the marks to be awarded as a comparison needed to be shown. Most candidates were aware of the queue operating on a first in, first out basis, whilst a stack operates using first in, last out. A minority of candidates achieved a second mark, most often by discussing the difference between a queue and a stack in terms of where elements are added or removed for each ADT.

Question 11

(a) This question involved writing clauses for a declarative programming language application and was answered well. Candidates who wrote their clauses in the correct order, with no spelling errors, no use of capitals and a full stop at the end of each line achieved full marks. Many high mark answers were seen.

(b) This question required candidates to write the result of a given goal in a declarative language application. Most candidates gave the correct answers and did not add any capital letters and so achieved the mark.

(c) The last part of the question was more challenging and required a rule to be given to match a given statement that would be used in the same declarative language application. Most candidates achieved at least one mark, but the full range of marks was seen.

Question 12

Candidates were asked to explain what is meant by the term ‘artificial neural network (ANN)’. Most candidates achieved at least one mark, usually for recognising how an ANN is modelled on a biological brain. However, a good range of well written answers was seen, with many candidates going on to achieve more marks.

COMPUTER SCIENCE

Paper 9618/32
Advanced Theory

Key messages

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

Candidates are advised to answer each question in an appropriate manner for the command word of the question. For example, a question beginning with 'explain' requires more detail than a question beginning with 'identify'. If a question asks for working to be shown, candidates must also ensure that they do this to gain full credit.

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

General comments

Candidates are advised to read questions carefully before beginning their answer 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. Marks could be awarded for how a given answer applies to the given scenario.

Comments on specific questions

Question 1

(a) The majority of candidates achieved at least one mark for this question, with many achieving all three for a fully correct solution with appropriate working present.

(b) Most candidates achieved some marks for this question. High marks were achieved by candidates who gave good explanations for the fact that not all fractional numbers can be accurately represented using binary, especially within the constraints of the given register.

Question 2

Most candidates described the composite data type referencing other data types, the non-composite data not needing to reference other data types, or both. However, few candidates expanded beyond this, for example, by stating that a composite data type is a collection of data that can consist of multiple elements grouped under a single identifier. A non-composite data type can be a primitive type available in a programming language or a user-defined type.

Question 3

(a) Many candidates correctly stated that a collision in the given context could be when a hashing algorithm generates the same hash value for two records, or the location identified by the hashing algorithm may already be occupied. Very few candidates got both. Some candidates incorrectly described collisions that occur during data transmission.

(b) Candidates who explained that a collision resolution process was required and went on to describe linear searching from the original hashed storage space, the use of an overflow area or the use of chaining to find a free location to store the new record achieved the marks. Most candidates achieved some marks and the full range of marks was seen.

Question 4

Candidates generally found this question challenging. The paragraph to be completed described how protocol implementation can be viewed as a stack, where each layer has its own functionality. The completed paragraph is:

The protocols in a **stack** determine the interconnectivity rules for a **layered** network model such as the **TCP/IP** model.

Question 5

(a) Candidates who stated that virtual memory is used when RAM is running low, and correctly expanded on this point achieved the marks. Most candidates achieved at least one mark.

(b) Candidates who explained that disk thrashing is a problem that may occur when using virtual memory and involves frequent swapping of pages in and out of main memory, leading to a very high rate of read/write head movements, which results in disk thrashing achieved the marks. A wide range of depth of answers was seen giving the full range of marks. Some candidates incorrectly linked disk thrashing to disk fragmentation.

Question 6

(a) This question was generally answered well with many candidates achieving high marks for showing the changing contents of a stack whilst evaluating a Reverse Polish Notation (RPN) expression.

(b) Most candidates achieved at least one mark for explaining how an RPN expression can be evaluated but the full range of marks was seen. The strongest answers described the RPN process in a general fashion that could be applied to any RPN expression.

Question 7

(a) Most candidates correctly completed the truth table for a 3-input NOR gate.

(b) A large proportion of candidates correctly applied De Morgan's laws to the given expression.

(c) Candidates were required to simplify a given expression using the rules of Boolean algebra. A large proportion of candidates managed to partially simplify the expression and achieved at least one mark, with many of these candidates achieving up to and including full marks. Some candidates did a good job of applying the rules of Boolean algebra to substantially simply the expression but did not quite get it to its simplest form.

Question 8

Candidates generally gave a good range of responses to demonstrate the use of Dijkstra's algorithm to find the shortest distance between a start point and each of the destinations in the diagram. A range of techniques for annotating how they did this was seen on the given diagram, as separate working, or a combination of both, with most candidates getting some marks and many candidates achieving four or five marks.

Question 9

(a) (i) Most candidates achieved one mark for identifying the data type of the pointers along with at least one correct description. Candidates who also gave correct descriptions for the other two identifiers achieved both marks. However, candidates often did not give sufficient detail for the description of the stack.

(ii) This question required candidates to complete the missing blanks in a pseudocode algorithm for a stack. It received a mixed set of responses with the full range of marks seen. A common error was candidates initialising the top pointer and base pointer the wrong way round.

(b) Candidates were asked to justify the use of a linked list instead of an array to implement a stack. To achieve both marks, candidates needed to give a statement about each of these Abstract Data Types (ADT) to justify this choice. Most candidates only mentioned one of the ADTs, therefore limiting their response to one mark.

(c) Candidates generally found this question challenging and simply explained recursion incorrectly. Candidates who began by stating that the compiler must produce object code and then explained how this code would be used to push return addresses onto a stack with each recursive call, then pop the return addresses from the stack after the base case is reached achieved the highest marks.

Question 10

Candidates were generally aware of the meaning of SIMD and MISD in terms of computer architectures, so achieved marks for this. However, very few candidates went beyond this, for example, to identify that both architectures make use of parallel computers with multiple processors, or that SIMD computers can take advantage of pipelining.

Question 11

(a) This question involving writing clauses for a declarative programming language application was answered well. Candidates who wrote their clauses in the correct order, with no spelling errors, no use of capitals and a full stop at the end of each line achieved full marks. Many high mark answers were seen.

(b) This question required candidates to write the result of a given goal in a declarative language application. Most candidates gave the correct answers and did not add any capital letters and so achieved the mark.

(c) The last part of the question was more challenging and required a rule to be given to match a given statement that would be used in the same declarative language application. Most candidates achieved at least one mark, but the full range of marks was seen. Some candidates achieved high marks.

Question 12

(a) Candidates were asked to describe an exception. They were expected to describe it as an event that occurs during the execution of a program that can cause the program to unexpectedly halt. Then they had to give an example of an exception, such as a run time error. Many candidates achieved one or both marks. However, some candidates incorrectly described exception handling, which is a remedy for exceptions rather than the exception itself.

(b) This question requiring candidates to complete the missing blanks in a pseudocode file handling search algorithm was generally answered well with many candidates achieving high marks. Most candidates achieved some marks for this question.

COMPUTER SCIENCE

Paper 9618/33
Advanced Theory

Key messages

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

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

General comments

Candidates are advised to read questions carefully before beginning their answer in order to understand what is being asked of them 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. Marks could be awarded for how a given answer applies to the given scenario.

Comments on specific questions

Question 1

(a) This question was usually well answered, with most candidates achieving at least one mark and many achieving all three for a fully correct solution showing appropriate working.

(b) Most candidates achieved at least one mark, but answers were often not detailed enough for both marks. Candidates generally recognised that not all fractions could be represented exactly in binary or that numbers may lose precision due to rounding.

Question 2

(a) Candidates generally scored high marks on this question, linking each protocol to its most appropriate use. However, not all candidates achieved full marks.

(b) A high proportion of candidates achieved at least one mark for outlining the purpose of the link layer in the TCP/IP protocol suite, with many of these candidates achieving both marks.

Question 3

Candidates generally recognised that both the enumerated and pointer data types are user-defined non-composite data types. They also often knew that an enumerated data type includes an ordered list of all possible values. Candidates who noted that pointer data types stored or referred to a memory address achieved another mark.

Candidates are advised not to use the name of a data type to describe it, for example, a pointer data type points to a memory location is not enough as it does not show understanding of the words 'point' or 'pointer'.

Question 4

(a) Candidates usually demonstrated understanding of the sequential and random file organisation methods. Candidates who correctly wrote about how records are organised within these methods achieved the highest marks. The full range of marks was seen with many candidates achieving high marks. However, some candidates incorrectly wrote about how files are organised rather than how records were organised within files.

(b) A good range of marks was seen, with candidates who described how a record was found by searching linearly from the beginning of the file until the record was found achieving the best marks.

Question 5

Candidates were generally aware of the meaning of SISD and MIMD in terms of computer architectures. A high proportion of candidates were also aware that an SISD computer contains a single processor. However, relatively few candidates expanded their MIMD description sufficiently. Those who did, stated that an MIMD architecture contains many processors that operate independently.

Question 6

(a) A high proportion of candidates correctly completed the truth table for a logic circuit that included a 3-input NAND gate.

(b) Candidates were required to simplify a given expression using the rules of Boolean algebra. This question was answered well by many candidates, with many achieving full marks. Many of the remaining candidates achieved at least one mark for showing some correct working.

Question 7

(a) Candidates who stated a benefit of a user interface, such as that the user interface hides the complexities of the computer hardware from the user, followed up with an example, such as clicking on an icon instead of writing code.

(b) The vast majority of candidates correctly identified blocked state as an additional process state.

(c) Most candidates achieved at least one mark here, usually for stating that a process could change from the running state to the ready state when its current time slice expires, or by describing the impact of the CPU receiving an interrupt.

Question 8

(a) This question required candidates to complete the missing blanks in a pseudocode algorithm which finds a record in a random file and outputs it. Many candidates found this question difficult, so it received a mixed set of responses.

(b) Candidates who defined 'exception handing' as a process of responding to an event in a running program so it does not halt unexpectedly, or similar, achieved the mark. A number of candidates incorrectly described the meaning of the term 'exception', instead.

(c) Most candidates were able to state at least one example of a cause of an exception, with many of these candidates stating two causes, for example, runtime error, or a specific error, such as division by zero.

Question 9

(a) (i) A large proportion of candidates achieved at least one mark for writing the infix version of a Reverse Polish Notation (RPN) expression. A common error was for candidates to leave out the brackets in their infix expression, therefore making their solution incorrect.

(ii) This question was generally answered well, with many candidates achieving high marks for showing the changing contents of a stack whilst evaluating an RPN expression.

(b) Most candidates were able to explain how a stack can be used to evaluate RPN expressions. The best answers explained the RPN evaluation process in a manner that could be applied to any RPN expression.

Question 10

(a) Candidates had to write the pseudocode to declare a constant, three variables and an array to be used in an algorithm representing a stack. Candidates who used the pseudocode style defined in the pseudocode guide that accompanies this course achieved the best marks.

(b) This question required candidates to complete the missing blanks in a pseudocode algorithm for a function to pop an element from a stack. A mixed range of results was seen with a significant number of candidates achieving high marks.

(c) Candidates were asked to compare and contrast the queue and stack Abstract Data Types (ADT) which meant that both types of ADT needed to appear in each statement for the marks to be awarded as a comparison needed to be shown. Most candidates were aware of the queue operating on a first in, first out basis, whilst a stack operates using first in, last out. A minority of candidates achieved a second mark, most often by discussing the difference between a queue and a stack in terms of where elements are added or removed for each ADT.

Question 11

(a) This question involved writing clauses for a declarative programming language application and was answered well. Candidates who wrote their clauses in the correct order, with no spelling errors, no use of capitals and a full stop at the end of each line achieved full marks. Many high mark answers were seen.

(b) This question required candidates to write the result of a given goal in a declarative language application. Most candidates gave the correct answers and did not add any capital letters and so achieved the mark.

(c) The last part of the question was more challenging and required a rule to be given to match a given statement that would be used in the same declarative language application. Most candidates achieved at least one mark, but the full range of marks was seen.

Question 12

Candidates were asked to explain what is meant by the term ‘artificial neural network (ANN)’. Most candidates achieved at least one mark, usually for recognising how an ANN is modelled on a biological brain. However, a good range of well written answers was seen, with many candidates going on to achieve more marks.

COMPUTER SCIENCE

Paper 9618/41
Practical

Key messages

Candidates need to submit only the evidence document, all evidence of their code and testing must be included within this document. This should be a single document and not a zip file.

When completing the evidence document candidates should make sure their evidence is in the correct answer space and should not delete any answer spaces.

Program code should be copied into each answer space and checked to make sure all required code is present, including all closing statements and brackets. When using Python candidates must make sure indentation is included and accurate in their answer. Code should not be screenshotted because the resolution of the image often does not allow for it to be read clearly.

When screenshotting the output from a program candidates must make sure all inputs to that program are also included. The screenshot must be clear and legible, some screenshots were too small or pixelated to read the values that were output. Screenshots must be of black text on a white background. A black background with coloured text is often illegible and it will not be awarded marks if it cannot be read.

General comments

Some candidates were able to follow the instructions and produce working solutions that met the criteria.

Candidates often started a task but then did not attempt later parts. The questions can often be answered independently, and marks can be gained even if earlier tasks are incomplete.

Candidates often had a good understanding of object-oriented programming and could create classes and get methods. Candidates found instantiating an object more challenging and this was often inaccurate.

When candidates are following pseudocode, they need to test their program code algorithm. This is because the pseudocode is not specific to one language. Therefore, for each language candidates will need to work out what the pseudocode is doing and how this needs to be changed for their own program.

Comments on specific questions

Question 1

This question required candidates to write iterative and recursive functions.

(a) (i) Candidates needed to convert the pseudocode algorithm into program code. The algorithm needed to be changed to meet their chosen language because the pseudocode is not specific to one language. For example the string manipulation functions needed to be altered to ones in their language and the loop condition needed altering in Python to run the correct number of times.

Some candidates were able to identify the appropriate string manipulation functions to access the data. Some candidates had an appropriate mid function but they used the incorrect values to access the required data. When converting a pseudocode algorithm candidates need to make sure they follow the structure given and do not change how the algorithm works by adding or removing elements.

(ii) Many candidates were able to call the function correctly with the string 'house', some candidates called the function with the variable house without speech marks to be a string. Some of these candidates were able to store and then output the return value.

(iii) Some candidates were able to produce the correct answer from the algorithm and provide this as a screenshot.

(b) (i) This question required candidates to re-create the same algorithm as a recursive algorithm. Many candidates did not produce a recursive solution, the algorithm did not contain a recursive call and often was a repetition of the answer to **part (a)(i)**.

Some candidates were able to produce a recursive algorithm, few of these were fully working. Some candidates retained a loop within the function which meant each call would run several times.

(ii) Some candidates were able to call the function they wrote in **part (b)(i)** and outputting the return value, including some candidates who did not have working solutions or did not attempt this part because the part is marked independently therefore marks were not dependent on having a response to **part (b)(i)**.

(iii) Few candidates were able to gain a correct output for this part.

Question 2

This question required candidates to create a linear queue and read data from an external text file to read into the queue.

(a) (i) This question required candidates to set up an array and variables for the Queue. Candidates were provided with the initial values for the points and told what these point to; some candidates did not follow this implementation and changed the value that the head and tail pointer pointed to which impact their responses throughout the question. There are many different ways that data structures can be set up and candidates need to follow the one given in the question paper to show that they understand how the different implementations work.

Many candidates were able to set up the array accurately with the required number of elements; candidates in Python often commented the number of elements and data type, or initialised the array with 50 strings. Some candidates gave a comment to indicate there was an array, but did not actually create an array.

(ii) This question required candidates to consider what indicates that the queue is full, which was given by the tail pointer. Some candidates tried to loop through the array to find an empty element and some candidate had an inaccurate value for the comparison.

Most candidates were able to declare the procedure header with a parameter, some candidates did not include the parameter. Some candidates inserted the data into the incorrect position in the queue, for example incrementing the tail before inserting at this incremented position.

Few candidates recognised the need to change the head pointer if it was the first element inserted because this would stay pointed at –1.

(iii) Some candidates were able to write an accurate dequeue function. Most candidates were able to create the function header without a parameter.

Few candidates could accurately check if the queue was completely empty. Candidates often checked if the head was –1 which would catch if there had never been an item inserted, but few also checked the head and tail were equal which would indicate items had been added and removed.

Some candidates implemented an additional pointer to store the number of elements in the queue. Where candidates initialised, incremented and decremented this value consistently and accurately throughout their algorithm, they were able to use this to check the queue was empty. This was not part of the instructions but demonstrated the candidates' understanding of how the queue working.

Some candidates returned a value within the function but then had vital actions after this return statement, this code would not run because the function will have already stopped.

(b) In this question the data from an external text file needed to be read into the program and enqueued. Many candidates were able to create the procedure header and also open the correct file, fewer candidates also closed the file in the correct place.

Some candidates looped until the end of the file, for example looping until EOF or reading in each line within a for loop in Python. Some candidates looped too many times reading in non-existent data, for example looping 50 times when there were fewer data items to read in.

(c) (i) Candidates can create record structures in a variety of ways, some languages record or type structures to be created. In other languages candidates could create a class, a dictionary or even a 2D list. Many candidates were able to do this, with the appropriate data types through declaration, initialisation or comments. Some candidates declared variables for the ID and Total but did not contain these within a structure. Where candidates chose to create a class they also needed a constructor to ensure that the fields were created for each new object that was instantiated.

(ii) Many candidates were able to store 0 in the variable NumberRecords, although some candidates attempted to declare this as an array. Some candidates were able to declare an array of the type they used in **part (c)(i)**, for example if they created a class then an array of objects was used. Candidates using Python did often use comments to show the data types, but some candidates only used comments and did not actually create an array or a variable.

(iii) This question provided a pseudocode algorithm for candidates to create in program code. Candidates needed to convert this algorithm into their chosen language and test it because pseudocode is not specific to one language and made use of a record format which not all candidates may have created in **part (c)(i)**. Some candidates were able to do this accurately and used the appropriate notation for their language. Candidates needed to make sure their loops iterated the appropriate number of times, for example if using Python then the end value in for .. in range loop needed to be one greater than in the pseudocode because of how Python differs from pseudocode. The pseudocode contained several comparisons, some candidates copied the pseudocode direct using a single = for comparison whilst their chosen programming language required a double = for comparison.

(d) Candidates needed to access the ID and Total for each of the records and output them in the given format i.e. with the string ID and Total included in the output message. Many candidates output the array in one statement, without accessing each field in each record in turn and outputting it in the appropriate format.

(e) (i) Many candidates were able to call ReadData and OutputRecords in the appropriate place. Few candidates identified that TotalData needed to be called for each of the records and instead only called it once.

(ii) Only a few candidates were able to complete the task fully accurately and gain the correct output for this question.

Question 3

This question required candidates to use object-oriented programming to create and move characters.

(a) (i) Many candidates were able to declare a class and constructor in their chosen programming language. Some candidates did not include the required parameters to the constructor, or assigned values to the attributes instead of the parameters.

(ii) Candidates often had a good understanding of get methods and were often able to create a method header without a parameter and return the correct value. Many candidates could also do this accurately for both get methods.

Some candidates included a parameter, or tried to read in or overwrite the attribute instead of just returning the attribute.

This is a method and therefore the Python answers needed the parameter self.

(iii) Candidates were often familiar with set methods but were unable to limit the values assigned. This validation is often used in set methods to make sure inappropriate values are not assigned to attributes. Many candidates were able to create the set method header with a parameter. Some candidates then checked whether the parameter was more than 10 000 or less than 0 instead of first adding this to the attribute. Some other responses checked the value of the attribute before adding the parameter, and some candidates checked the value but then output a message to state that it was invalid and did not update the attribute as required.

This is a method and therefore the Python answers needed the parameter self.

(iv) This question required candidates to write a method to change the coordinates of the character. The method needed to take the direction as a parameter, some candidates instead tried to read in a value within the method, or use a variable where it was not clear what its contents were or where they are assigned.

The data in the parameter needed to be compared to the four options, some candidates were able to do this and used an appropriate selection statement; some candidates did not use the string for each direction as a comparison, for example comparing the parameter to up instead of 'up'.

Some candidates used the set method appropriately to update the value, and many of these were able to pass the correct value as a parameter for example –10 to move down or left. Some candidates did not identify that Move() was a method and tried to access a specific object and its set method instead of just calling the set method.

This is a method and therefore the Python answers needed the parameter self.

(b) Many candidates found the instantiation of an object challenging. Some candidates were able to create a new instance of their class, but some candidates attempted to create an array or an instance of Jack and store it in a variable Character. Candidates who did create an object successfully were often able to pass the appropriate values to the constructor. Where a constructor is declared, this should be used to instantiate an object, some candidates attempted to assign the values after instantiation.

(c) (i) Candidates were often able to create a class but fewer used inheritance correctly. Some candidates were able to create the constructor with the correct parameters, but fewer then called the parent constructor within this; candidates often assigned the values to the attributes within the constructor.

(ii) Candidates needed to override the parent method in this question, the method of which varied depending on the programming language, for example Python did not require any additional code to indicate overriding. Some candidates who chose Python did not include the required self parameter for this method.

Some candidates repeated their code from the previous part and changed the values, whilst some candidates called the parent method twice which would increase this by the correct amount as it values were just added or subtracted.

(d) Few candidates were able to accurately create an instance of BikeCharacter, with the correct parameters and assign it to a suitable variable. Some candidates attempted to instantiate a Character and then call the constructor for BikeCharacter separately.

(e) (i) Candidates who attempted this question were often able to take the character and direction as input, some candidates took inputs but did not include the required prompt for each one. Fewer candidates validated these two inputs, some candidates who did attempt validation only did so for the character or the direction, whilst some output a message to state the data was valid but then allowed the invalid data to be used in the program. Some candidates used loops to continually read each value as input until it was one of the valid options.

The input values then needed to be used to decide which character was being moved and have the Move method called for that character with the direction input. Some candidates tried to create a copy of the object and then use this object which meant that the position of the actual original character was not actually moved, only the copy was moved.

The question required the outputs in an appropriate format and provided one as a guide, some candidates did output the appropriate values but did these in isolation without any additional context.

(ii) Few candidates were able to complete the solution fully correctly and create the correct output for the two tests.

COMPUTER SCIENCE

Paper 9618/42
Practical

Key messages

Candidates need to submit only the evidence document, all evidence of their code and testing must be included within this document. This should be a single document and not a zip file.

When completing the evidence document candidates should make sure their evidence is in the correct answer space and should not delete any answer spaces.

Program code should be copied into each answer space and checked to make sure all required code is present, including all closing statements and brackets. When using Python candidates must make sure indentation is included and accurate in their answer. Code should not be screenshotted because the resolution of the image often does not allow for it to be read clearly.

When screenshotting the output from a program candidates must make sure all inputs to that program are also included. The screenshot must be clear and legible, some screenshots were too small or pixelated to read the values that were output. Screenshots must be of black text on a white background. A black background with coloured text is often illegible and it will not be awarded marks if it cannot be read.

General comments

Some candidates were able to follow the instructions and produce working solutions that met the criteria. Some candidates were unable to produce working solutions to any tasks, for example they were unable to declare variables and assign values to them.

Candidates often started a task but then did not attempt later parts. The questions can often be answered independently, and marks can be gained even if earlier tasks are incomplete.

Candidates often had a good understanding object-oriented programming and could create classes and get methods. Candidates found instantiating an object more challenging and this was often inaccurate.

When candidates are following pseudocode, they need to test their program code algorithm because the pseudocode is not specific to one language. Therefore, for each language candidates will need to work out what the pseudocode is doing and how this needs to be changed for their own program.

Comments on specific questions

Question 1

This question required candidates to program a stack using a 1D array and read data into the stack from an external text file.

(a) (i) Many candidates were able to create two arrays with the appropriate data type and the correct number of elements. Some candidates declared the number of elements using a declaration statement, candidates who wrote in Python often initialised the arrays with 100 strings or used a comment to give the declaration. This question does require the creation of the arrays, a comment alone is insufficient to show that the arrays have been created. Some candidates used an inappropriate data type or mismatched the data types for example declaring the data type as string but then assigning integers.

(ii) Most candidates were able to create two variables as integers and assign the integer 0 to each.

(b) (i) This question required a function to push data onto the stack until it was full. Few candidates were able to write a completely accurate push function. Most candidates were able to create the procedure header, some candidates did not consider whether the parameter was a vowel of a consonant and instead created an additional stack to store all of the data.

Many candidates attempted to check if the stack was full, but fewer were able to check this accurately; some candidates attempted to count the number of elements and did not use the top pointer. The question required a message to be output a message when it was full, some candidates returned a value instead of outputting it.

(ii) Some candidates found this question challenging and were not able to read the data in from an external file. Some candidates opened the file to read but did not close the file. Many candidates attempted exception handling, but some of these responses did not use it appropriately, for example they opened a file within the try and then closed it outside of the exception handling or closed it in the catch stage.

Candidates who opened the file commonly looped until the end of the file through a variety of ways; in Python it was commonly by looping through each item in the file using a for loop.

(c) This question required two functions, one to remove the next item from each of the stacks.

A common mistake was checking if the stack was empty and returning a value this time; some candidate did not check if the stack was empty and many candidates output a message instead of returning the required string.

Errors were often repeated in the two functions, or the second function was implemented inaccurately by additional errors being introduced, or the wrong stack identifier was used when the functions were copied.

(d) (i) Many candidates were able to call the ReadData() function correctly. Some candidates were also able to read in an input; some candidates did not do this repeatedly or did not include a prompt for the input.

Candidates who did take the input were often able to check if it was a vowel or consonant and call the appropriate function. Many of these candidates also stored the return value.

This question required the return value to be checked to make sure there was data to be returned. It had to continue looping until 5 letters had been successfully popped. Many candidates looped 5 times and did not take into account whether a value had been successfully popped or not, this meant some responses did not have 5 characters, whilst other responses would have included ‘No data’ as one of the characters because it was automatically appended to a list of data to output.

(ii) Few candidates were able to get a fully correct output. Some candidates had an incorrect final string, for example with the incorrect value, whilst other candidates did not have the prompt and input of the 5 required choices.

Question 2

This question required candidates to understand how to convert a pseudocode algorithm into program code and how to create iterative and recursive functions.

(a) (i) This question required candidates to convert the pseudocode into their own language. Depending on their chosen language, this may have required altering aspects including the conditions and the modulus operator. Many candidates were able to do this accurately and correctly converted the algorithm. Some candidates did not know how to calculate modulus division and some candidates added additional code to perform other operations that were not part of the algorithm.

(ii) Most candidates were able to call the function accurately, fewer stored and output the return value. Some candidates output the value inside the function, and some candidates change the function to use the value 10 hardcoded into the algorithm.

(iii) Some candidates were able to get the correct output for the program.

(b) (i) Candidates had to complete the spaces in the recursive algorithm whilst converting it into their chosen programming language. Some candidates were able to do all of this accurately, completing all spaces and converting the rest of the algorithm accurately.

There were a range of different responses for the gaps that did not work accurately, some candidates did not maintain the recursive calls and attempted to introduce additional loops.

When candidates are given a pseudocode algorithm, they need to make sure they are copying the structure i.e. the parameters, the conditions, loops, calls etc.

(ii) Some candidates were able to call the function with the correct values and output the return value. Some candidates only used one parameter of 50 instead of the required two. A common error was not storing and outputting the return value.

(iii) Some candidates were able to get the correct output for the program.

Question 3

This question required candidates to use object-oriented programming.

(a) (i) Many candidates were able to define a class and many of these accurately created a constructor within that class declaration. Some candidates did not include the required parameters in the constructor. Some candidates were able to assign the parameters to the attributes, but some candidates attempted to store initial values instead.

(ii) Many candidates were familiar with get methods and were able to accurate create a get method without a parameter and return the correct attributes. Some candidates who were not familiar with get methods attempted to send a parameter and then return this instead of the attribute. Some candidates overwrote the attribute value before returning it. This is a method within a class, therefore candidates using Python needed to have the parameter self.

(iii) Some candidates did not seem familiar with set methods and attempted to take an input or return a value within the method. Some candidates were able to accurately define a set method, taking the required parameter and assigning it to the attribute.

(iv) Candidates often found this question challenging. Few candidates were able to accurately increase the attribute value by 10 per cent, a common error was multiplying the attribute by 10 per cent which did not increase the attribute value by the required amount. Some candidates attempted to take a parameter or an input and then update this value instead of accessing the attribute. This is a method within a class, therefore candidates using Python needed to have the parameter self.

(v) This question required candidates to consider how the date of birth was stored as a date type and to use appropriate date functions to calculate the age of the character for the year 2023. Some candidates were able to demonstrate an understanding of date functions and they accurately extracted the year from the date of birth. Some candidates were able to calculate a value (inaccurately) and then return this value.

This is a method within a class, therefore candidates using Python needed to have the parameter self.

(b) (i) Some candidates were able to create an instance of the class, but many were unable to store this in an appropriate variable or assign the correct values to the constructor call. A common error was not storing the full date of birth as required, instead only storing the year.

(ii) This question required the calling of a method for the object created in **part (b)(i)**, few candidates could do this and instead called the method independently. The same error occurred with calling ReturnAge which was a method call for the object. Some candidates attempted to output the name and intelligence, but some candidates did not include a suitable message as part of this output.

(iii) Few candidates were able to generate the correct output; some candidates had the incorrect age and intelligence.

(c) (i) Some candidates were able to declare the additional attribute with a string and assign a parameter to this value. Candidates commonly declared the class but few included the required inheritance from the class Character. The constructor was often declared but did not always contain all of the required parameters. Few candidates were able to call the parent class constructor with the required parameters.

(ii) This question required the overriding of the parent method, the means of doing this differed depending on the language. A common error was the use of parameters for the current intelligence and the element, instead of using the object's attributes. Many candidates created an independent procedure instead of a method.

Candidates often had a selection statement, but this was not always checking the correct values, for example taking an input for the element. The calculation of the intelligence was often inaccurate and added values to the intelligence instead of calculating the correct percentage. Some candidates called the parent Learn method first, then recalculated the intelligence which produce an inaccurate final value.

(d) (i) Some candidates attempted this question and often declared the correct instance of the object, fewer were able to send the correct values to the constructor, for example an incorrect or incomplete date of birth.

(ii) Few candidates were able to complete this task fully accurately. Many candidates did not call the methods for the object and some candidates did not use the appropriate methods to access the data values.

(iii) Some candidates were able to produce an accurate output. Some common errors included an incorrect age or outputting a date instead of an age, and the incorrect final intelligence.

COMPUTER SCIENCE

Paper 9618/43
Practical

Key messages

Candidates need to submit only the evidence document, all evidence of their code and testing must be included within this document. This should be a single document and not a zip file.

When completing the evidence document candidates should make sure their evidence is in the correct answer space and should not delete any answer spaces.

Program code should be copied into each answer space and checked to make sure all required code is present, including all closing statements and brackets. When using Python candidates must make sure indentation is included and accurate in their answer. Code should not be screenshotted because the resolution of the image often does not allow for it to be read clearly.

When screenshotting the output from a program candidates must make sure all inputs to that program are also included. The screenshot must be clear and legible, some screenshots were too small or pixelated to read the values that were output. Screenshots must be of black text on a white background. A black background with coloured text is often illegible and it will not be awarded marks if it cannot be read.

General comments

Some candidates were able to follow the instructions and produce working solutions that met the criteria.

Candidates often started a task but then did not attempt later parts. The questions can often be answered independently, and marks can be gained even if earlier tasks are incomplete.

Candidates often had a good understanding of object-oriented programming and could create classes and get methods. Candidates found instantiating an object more challenging and this was often inaccurate.

When candidates are following pseudocode, they need to test their program code algorithm. This is because the pseudocode is not specific to one language. Therefore, for each language candidates will need to work out what the pseudocode is doing and how this needs to be changed for their own program.

Comments on specific questions

Question 1

This question required candidates to write iterative and recursive functions.

(a) (i) Candidates needed to convert the pseudocode algorithm into program code. The algorithm needed to be changed to meet their chosen language because the pseudocode is not specific to one language. For example the string manipulation functions needed to be altered to ones in their language and the loop condition needed altering in Python to run the correct number of times.

Some candidates were able to identify the appropriate string manipulation functions to access the data. Some candidates had an appropriate mid function but they used the incorrect values to access the required data. When converting a pseudocode algorithm candidates need to make sure they follow the structure given and do not change how the algorithm works by adding or removing elements.

(ii) Many candidates were able to call the function correctly with the string ‘house’, some candidates called the function with the variable house without speech marks to be a string. Some of these candidates were able to store and then output the return value.

(iii) Some candidates were able to produce the correct answer from the algorithm and provide this as a screenshot.

(b) (i) This question required candidates to re-create the same algorithm as a recursive algorithm. Many candidates did not produce a recursive solution, the algorithm did not contain a recursive call and often was a repetition of the answer to **part (a)(i)**.

Some candidates were able to produce a recursive algorithm, few of these were fully working. Some candidates retained a loop within the function which meant each call would run several times.

(ii) Some candidates were able to call the function they wrote in **part (b)(i)** and outputting the return value, including some candidates who did not have working solutions or did not attempt this part because the part is marked independently therefore marks were not dependent on having a response to **part (b)(i)**.

(iii) Few candidates were able to gain a correct output for this part.

Question 2

This question required candidates to create a linear queue and read data from an external text file to read into the queue.

(a) (i) This question required candidates to set up an array and variables for the Queue. Candidates were provided with the initial values for the points and told what these point to; some candidates did not follow this implementation and changed the value that the head and tail pointer pointed to which impact their responses throughout the question. There are many different ways that data structures can be set up and candidates need to follow the one given in the question paper to show that they understand how the different implementations work.

Many candidates were able to set up the array accurately with the required number of elements; candidates in Python often commented the number of elements and data type, or initialised the array with 50 strings. Some candidates gave a comment to indicate there was an array, but did not actually create an array.

(ii) This question required candidates to consider what indicates that the queue is full, which was given by the tail pointer. Some candidates tried to loop through the array to find an empty element and some candidate had an inaccurate value for the comparison.

Most candidates were able to declare the procedure header with a parameter, some candidates did not include the parameter. Some candidates inserted the data into the incorrect position in the queue, for example incrementing the tail before inserting at this incremented position.

Few candidates recognised the need to change the head pointer if it was the first element inserted because this would stay pointed at –1.

(iii) Some candidates were able to write an accurate dequeue function. Most candidates were able to create the function header without a parameter.

Few candidates could accurately check if the queue was completely empty. Candidates often checked if the head was –1 which would catch if there had never been an item inserted, but few also checked the head and tail were equal which would indicate items had been added and removed.

Some candidates implemented an additional pointer to store the number of elements in the queue. Where candidates initialised, incremented and decremented this value consistently and accurately throughout their algorithm, they were able to use this to check the queue was empty. This was not part of the instructions but demonstrated the candidates’ understanding of how the queue working.

Some candidates returned a value within the function but then had vital actions after this return statement, this code would not run because the function will have already stopped.

(b) In this question the data from an external text file needed to be read into the program and enqueued. Many candidates were able to create the procedure header and also open the correct file, fewer candidates also closed the file in the correct place.

Some candidates looped until the end of the file, for example looping until EOF or reading in each line within a for loop in Python. Some candidates looped too many times reading in non-existent data, for example looping 50 times when there were fewer data items to read in.

(c) (i) Candidates can create record structures in a variety of ways, some languages record or type structures to be created. In other languages candidates could create a class, a dictionary or even a 2D list. Many candidates were able to do this, with the appropriate data types through declaration, initialisation or comments. Some candidates declared variables for the ID and Total but did not contain these within a structure. Where candidates chose to create a class they also needed a constructor to ensure that the fields were created for each new object that was instantiated.

(ii) Many candidates were able to store 0 in the variable NumberRecords, although some candidates attempted to declare this as an array. Some candidates were able to declare an array of the type they used in **part (c)(i)**, for example if they created a class then an array of objects was used. Candidates using Python did often use comments to show the data types, but some candidates only used comments and did not actually create an array or a variable.

(iii) This question provided a pseudocode algorithm for candidates to create in program code. Candidates needed to convert this algorithm into their chosen language and test it because pseudocode is not specific to one language and made use of a record format which not all candidates may have created in **part (c)(i)**. Some candidates were able to do this accurately and used the appropriate notation for their language. Candidates needed to make sure their loops iterated the appropriate number of times, for example if using Python then the end value in for .. in range loop needed to be one greater than in the pseudocode because of how Python differs from pseudocode. The pseudocode contained several comparisons, some candidates copied the pseudocode direct using a single = for comparison whilst their chosen programming language required a double = for comparison.

(d) Candidates needed to access the ID and Total for each of the records and output them in the given format i.e. with the string ID and Total included in the output message. Many candidates output the array in one statement, without accessing each field in each record in turn and outputting it in the appropriate format.

(e) (i) Many candidates were able to call ReadData and OutputRecords in the appropriate place. Few candidates identified that TotalData needed to be called for each of the records and instead only called it once.

(ii) Only a few candidates were able to complete the task fully accurately and gain the correct output for this question.

Question 3

This question required candidates to use object-oriented programming to create and move characters.

(a) (i) Many candidates were able to declare a class and constructor in their chosen programming language. Some candidates did not include the required parameters to the constructor, or assigned values to the attributes instead of the parameters.

(ii) Candidates often had a good understanding of get methods and were often able to create a method header without a parameter and return the correct value. Many candidates could also do this accurately for both get methods.

Some candidates included a parameter, or tried to read in or overwrite the attribute instead of just returning the attribute.

This is a method and therefore the Python answers needed the parameter self.

(iii) Candidates were often familiar with set methods but were unable to limit the values assigned. This validation is often used in set methods to make sure inappropriate values are not assigned to attributes. Many candidates were able to create the set method header with a parameter. Some candidates then checked whether the parameter was more than 10 000 or less than 0 instead of first adding this to the attribute. Some other responses checked the value of the attribute before adding the parameter, and some candidates checked the value but then output a message to state that it was invalid and did not update the attribute as required.

This is a method and therefore the Python answers needed the parameter self.

(iv) This question required candidates to write a method to change the coordinates of the character. The method needed to take the direction as a parameter, some candidates instead tried to read in a value within the method, or use a variable where it was not clear what its contents were or where they are assigned.

The data in the parameter needed to be compared to the four options, some candidates were able to do this and used an appropriate selection statement; some candidates did not use the string for each direction as a comparison, for example comparing the parameter to up instead of 'up'.

Some candidates used the set method appropriately to update the value, and many of these were able to pass the correct value as a parameter for example –10 to move down or left. Some candidates did not identify that Move() was a method and tried to access a specific object and its set method instead of just calling the set method.

This is a method and therefore the Python answers needed the parameter self.

(b) Many candidates found the instantiation of an object challenging. Some candidates were able to create a new instance of their class, but some candidates attempted to create an array or an instance of Jack and store it in a variable Character. Candidates who did create an object successfully were often able to pass the appropriate values to the constructor. Where a constructor is declared, this should be used to instantiate an object, some candidates attempted to assign the values after instantiation.

(c) (i) Candidates were often able to create a class but fewer used inheritance correctly. Some candidates were able to create the constructor with the correct parameters, but fewer then called the parent constructor within this; candidates often assigned the values to the attributes within the constructor.

(ii) Candidates needed to override the parent method in this question, the method of which varied depending on the programming language, for example Python did not require any additional code to indicate overriding. Some candidates who chose Python did not include the required self parameter for this method.

Some candidates repeated their code from the previous part and changed the values, whilst some candidates called the parent method twice which would increase this by the correct amount as it values were just added or subtracted.

(d) Few candidates were able to accurately create an instance of BikeCharacter, with the correct parameters and assign it to a suitable variable. Some candidates attempted to instantiate a Character and then call the constructor for BikeCharacter separately.

(e) (i) Candidates who attempted this question were often able to take the character and direction as input, some candidates took inputs but did not include the required prompt for each one. Fewer candidates validated these two inputs, some candidates who did attempt validation only did so for the character or the direction, whilst some output a message to state the data was valid but then allowed the invalid data to be used in the program. Some candidates used loops to continually read each value as input until it was one of the valid options.

The input values then needed to be used to decide which character was being moved and have the Move method called for that character with the direction input. Some candidates tried to create a copy of the object and then use this object which meant that the position of the actual original character was not actually moved, only the copy was moved.

The question required the outputs in an appropriate format and provided one as a guide, some candidates did output the appropriate values but did these in isolation without any additional context.

(ii) Few candidates were able to complete the solution fully correctly and create the correct output for the two tests.