COMPUTER SCIENCE

Paper 9618/11
Theory Fundamentals

Key messages

At this level of study candidates are expected to demonstrate more than just general knowledge and should be able to provide more detailed and technical answers than would be expected at GCSE. It is very important that the correct technical terms are used appropriately. Many candidates would benefit from a better understanding of the different command words used in the questions. A question that begins with the command word 'explain' requires a different type of answer to one that begins with the command word 'state'.

Each question should also be read carefully and answered appropriately. For example, if a question asks for the benefits of a particular translator, no marks will be awarded for just describing how the translator operates. Marks will be awarded for describing the advantages of using that translator in the circumstances outlined in the question.

If a question asks for a given number of responses and the answer space is numbered, only the first response in each numbered space will be marked. Candidates should be encouraged to think carefully before starting to write their answer and to plan what they wish to write. In some cases when a question asked for two reasons, candidates listed many more than two.

General comments

Candidates should ensure their final answer is written legibly in pen, in the correct space on the question paper. If candidates wish to write rough versions of their answers first, they should be encouraged to use either the blank pages at the end of the question paper or additional sheets of paper. In either case the rough work should be clearly crossed through.

Some questions required that working be shown. In these questions it is very important that candidates indicate clearly which value is meant to be the answer. Sometimes the working space was covered with numbers but none of them were labelled as the answer.

Questions about number bases and the logic circuit were usually completed successfully, but questions about manging interrupts and hardware were less well answered.

Comments on specific questions

Question 1

(a) Pixel and colour depth were described well. Some candidates found describing the drawing list more challenging. Care needed to be taken with the words used. Describing the drawing list as "*all the shapes that can be used*" was incorrect as this describes the shape library not the drawing list. A better answer was "*all the drawing objects that make up an image".*

(b) (i) This question was answered very well. Almost all candidates were able to correctly identify two items that could be included in the header of a bitmap file.

(ii) This question was answered well but some candidates did not read the question carefully and used 8 bits instead of 8 bytes for the bit depth, thus giving an incorrect answer of 4.5MB.

(c) (i) This question was also answered well. Most candidates were able to give two benefits of compressing the photographs. Some candidates found it more challenging to give either a third benefit or to expand on one of their other answers to gain the final mark.

(ii) Many candidates realised that if colours were not repeated in sequence, then RLE would probably not reduce the file size. The question explicitly referred to a bitmap image but some candidates incorrectly described the compression of a text file and gave an example using characters instead of colours.

Question 2

(a) (i) There were many correct definitions of a data dictionary. Some candidates needed to understand that it is data about the data in a database and not the data itself. The example was usually correct and primary and foreign keys were the most popular answers.

(ii) Some candidates found this question challenging. There were many incorrect statements such as "*data integrity makes sure that the data is correct"*. Data integrity makes sure that the data is consistent and up to date, but it does not ensure accuracy or correctness. These words are used in a technical way so candidates should ensure that they use them appropriately.

(b) (i) There were many correct answers to this question. Some candidates incorrectly included the primary key table as well.

(ii) This question was answered well. Most candidates were able to correctly match the Normal Form to the appropriate definition.

(iii) There were a small number of excellent answers to this question but some otherwise correct responses were not awarded full marks because BirdID was defined as an integer. A significant number of candidates did not have a secure enough understanding of SQL and its use in defining database tables.

(iv) Many candidates found completing this SQL script challenging. Section 8.3 of the syllabus includes the INNER JOIN statement. However, many candidates seemed unaware of the alternative method of joining of two tables used in the SQL script given and tried to rewrite the complete script using an INNER JOIN.

Question 3

(a) Many of the answers to this question did not demonstrate enough technical knowledge for credit at this level of study. It was expected that descriptions would include statements about the installation and updating of drivers, about the management of interrupts from the peripheral and the role of the Operating System in the use of buffers for data transfer.

(b) Many candidates were able to correctly identify two other Operating System management tasks. A small number of candidates repeated the example given in the question.

(c) This was a question that needed to be read carefully. The question asked how defragmentation can improve computer performance. Unfortunately many candidates saw the word 'defragmentation' and simply wrote down a description of how defragmentation software operates without any reference to computer performance and so did not answer the question.

(d) (i) There were some good correct answers to this question. Some candidates needed to understand that vague statements such as "*a kibibyte is bigger than a kilobyte*" were not sufficient for credit at this level. Some candidates confused the two prefixes and stated that a kibibyte was 1000 bytes and a kilobyte was 1024 bytes. Some candidates needed to take care when writing their answers. A frequent error was a statement such as "*a kilobyte is 1000 bits"*.

(ii) There were many correct answers to this question. The most common incorrect answer occurred when candidates converted the denary number 964 to an unsigned binary integer instead of to BCD.

(iii) There were also many correct answers to this question. The most common incorrect answer occurred when candidates converted the binary integer given into denary instead of hexadecimal.

(iv) Many candidates found this question challenging. The question asked for the two's complement binary representation of the numbers but there were a number of responses where the correct denary values had been written, but these values had not then been converted to binary.

(v) This question was answered very well. Almost all candidates were able to correctly perform the binary addition. Some candidates needed to make their working more obvious.

(vi) This question was answered very well. Almost all candidates were able to correctly perform the required shift operation. A small number of candidates shifted the original value left instead of right.

Question 4

(a) (i) Many candidates were able to correctly identify one item that would be stored in the ROM, but identifying a second item proved to be more challenging. There was some confusion between the contents of ROM and the contents of RAM and some candidates did not answer the question, and instead just described the ROM.

(ii) Many candidates found this question challenging. Vague statements such as "*DRAM is faster*" or "*DRAM is cheaper*" were not enough for credit at this level.

(b) Many candidates also found this question challenging. Again statements such as "*magnetic storage is cheaper" were* not enough. "*Magnetic storage costs less per storage unit"* was a better answer. There also seemed to be a misconception that solid state drives could not be used for long term backup. Few candidates mentioned anything about the number of read/write operations required.

(c) There were some good, imaginative answers to this question. Many candidates were able to describe how the AI uses image recognition to focus on a person. Some candidates needed to be careful that they are not just describing the use of motion sensors in a generic monitoring or control system. It was sometimes difficult to work out the role of the AI in candidate responses.

(d) (i) Many candidates correctly filled in the values for the IPv4 address. Completing the paragraph about IPv6 was more challenging. A common incorrect answer for the last space was a single colon rather than a double colon.

(ii) This question was challenging for many candidates. Many of the responses described the generalised benefits of any network and did not answer the question which asked for two benefits of subnetting a network.

Question 5

(a) There were many correct answers to this question. A common incorrect circuit was where the NOT A OR B had been incorrectly replaced with A NOR B. Some candidates needed to be careful that they differentiated between the gates. The best way to avoid any confusion is to use the symbols shown in section 3.2 of the syllabus.

(b) Some candidates found describing the operation of the two gates very challenging. Vague statements such as "*a NAND gate is the opposite of an AND gate*" were not enough. The answer needed to include a description of the operation of the gate for each possible combination of inputs.

Question 6

There were a few excellent, complete answers to this question. However, many answers were too general. There was little mention of checking the priority of the interrupt and few candidates stated where in the FE cycle the check for interrupts would occur. There seemed to be an assumption that all current processes would be stopped immediately and the interrupt from the keyboard would be processed without any delay.

COMPUTER SCIENCE

Paper 9618/12
Theory Fundamentals

Key messages

At this level of study candidates are expected to demonstrate more than just general knowledge and should be able to provide more detailed and technical answers than would be expected at GCSE. It is very important that the correct technical terms are used appropriately. Many candidates would benefit from a better understanding of the different command words used in the questions. A question that begins with the command word 'explain' requires a different type of answer to one that begins with the command word 'state'.

Each question should also be read carefully and answered appropriately. For example, if a question asks for the benefits of a particular translator, no marks will be awarded for just describing how the translator operates. Marks will be awarded for describing the advantages of using that translator in the circumstances outlined in the question.

If a question asks for a given number of responses and the answer space is numbered, only the first response in each numbered space will be marked. Candidates should be encouraged to think carefully before starting to write their answer and to plan what they wish to write. In some cases when a question asked for two reasons, candidates listed many more than two.

General comments

Candidates should ensure their final answer is written legibly in pen, in the correct space on the question paper. If candidates wish to write rough versions of their answers first, they should be encouraged to use either the blank pages at the end of the script, or additional sheets of paper. In either case the rough work should be clearly crossed through.

Some questions required that working be shown. In these questions it is very important that candidates indicate clearly which value is meant to be the answer. Sometimes the working space was covered with numbers but none of them were labelled as the answer.

Questions about assembly language and the number base conversions were usually completed successfully, but questions about Ethernet protocol and USB ports were answered less well.

Comments on specific questions

Question 1

(a) While there were some good answers to this question, there was also a lot of confusion between benefits of a LAN and characteristics of a LAN. Benefits needed to be linked to why the computers would be connected together to form a LAN. A significant number of answers compared a LAN to a WAN which was not what was required. A common response was "*a LAN gives each computer access to the internet"*. There is no mention of any internet connection in the question and the LAN could be solely for internal use.

(b) The most usual correct answer to this question referred to the size of the network. However, answers such as "*a LAN covers a local area*" which simply repeated the words of the acronym were too vague for credit.

(c) There were some very good answers to this question. Some answers also correctly included a hub or switch which was not included in the question and was not necessarily required in the answer. Some candidates needed to be aware of the need to fully label diagrams.

(d) Many candidates found this question very challenging. The majority of answers began with the words, "*Ethernet is a cable*". There was little understanding that an Ethernet cable is a cable used to transfer data using the Ethernet protocol.

(e) This question required responses that described a thick-client model, so it was expected that there would be at least one statement about the role of the (thick) client and another about the role of the server in this model. Many responses were too vague and could apply to any general client-server model, not specifically a thick-client model.

Question 2

(a) There were some good answers describing the data dictionary and correctly identifying the logical schema. Some candidates found identification of the developer interface and describing the query processor more challenging. Statements such as, "*the query processor processes queries*" were far too vague for credit at this level of study. Better descriptions of the query processor included a statement about the execution of queries written in SQL.

(b) This question was answered well. Many candidates were able to give at least two reasons why referential integrity is important in a database.

(c) (i) The command word in this question was 'describe', so it was not enough to simply name two validation checks. The question was also very specific in asking for two methods of validating a particular field, `RiderLevel`, in the database. Generic descriptions of validation checks therefore, did not answer the question. Responses needed to refer explicitly to that field.

(ii) There were some good correct SQL scripts. Some candidates would have benefitted from more practice in setting up simple databases and using the various SQL statements in section 8.3 of the syllabus to query and maintain these databases. Some candidates confused the rider level from **(i)** with the horse level in this question and so returned the incorrect result.

(iii) This was another question where candidates needed to read the question carefully. The question asked for identification and correction of the errors. Some candidates identified the errors but did not offer any correction. Section 8.3 of the syllabus includes the INNER JOIN statement. However, many candidates seemed unaware of the alternative method of joining of two tables used in the SQL script given and tried to rewrite the complete script using an INNER JOIN. Some candidates who had correctly enclosed the text in quotation marks in their answer to **(ii)** did not spot the missing quotation marks here.

Question 3

(a) There were some good answers to this question. Many candidates were able to match the generation of the object code, the removal of white space and the addition of labels to the correct pass. The reading of the source code was the action most likely to be incorrectly matched. Many candidates did not join the action to both passes.

(b) There were many completely correct answers to this question, with all three addressing modes correctly identified.

Question 4

(a) This question was answered well. Many candidates correctly stated the number of values. A small number gave the answer of 65535, that is $2^{16} - 1$.

(b) Many candidates needed to improve their understanding of what is meant by the one's complement of a binary number. Often either +120 had been correctly converted to binary, but then the answer given was the two's complement, or candidates converted –120 to two's complement binary (10001000) and then took the one's complement of that value to give (01110111).

(c) There were many completely correct answers to this question. When converting the hexadecimal value A04 to binary, some candidates omitted the central zeros and gave an incorrect answer of 10100100 in binary or 164 in denary.

(d) This question was answered very well. Almost all candidates were able to correctly perform the required shift operation. A small number of candidates shifted the original value right instead of left.

Question 5

(a) Many candidates were able to correctly state that interrupt signals are carried on the control bus but found it more challenging to give a second correct answer. A common answer was "*control signals*". At this level of study marks were not awarded for answers which did not demonstrate any understanding beyond that the control bus carries control signals.

(b) There were some excellent, complete answers to this question. The two most popular hardware upgrades were increasing the number of cores in the processor and increasing the clock speed. Some candidates needed to be aware that when asked for a hardware upgrade an answer of "*upgrade the CPU*" was not enough. There needed to be an indication of how the CPU was to be upgraded.

(c) (i) This question was challenging for most candidates. Many candidates overlooked the word 'port' in the question and described plugging a USB memory stick into a computer to transfer a document to the printer. A popular answer from candidates who did recognise what the question required, included the description of data being transmitted in parallel.

(ii) This question was answered very well and the most popular correct answer was an HDMI port.

(d) (i) Section 5.1 of the syllabus lists the management task of an operating system with which candidates are expected to be familiar. One of these is process management. There were a small number of very good answers. However, very few candidates recognised that the word 'processes' was being used in the technical sense and the most usual answer was a list of all the different management functions the Operating System.

(ii) This was a question that needed to be read carefully. It asked for a description of the purpose of utility software. In other words, why utility software is necessary. Some candidates found this very challenging and many answers simply gave a list of common utility software.

Question 6

(a) This question was answered very well. Some candidates needed to be careful that they differentiate between AND and OR gates and between NOT and NOR gates. The best way to avoid any confusion was to use the symbols shown in section 3.2 of the syllabus.

(b) There were some very good correct answers to this question. The most frequent incorrect answer was where the output was completely reversed because candidates had applied the XOR gate incorrectly.

Question 7

(a) (i) Many candidates were able to define what was meant by a program library using the appropriate terminology. Some candidates needed to be aware that generalised answers such as "*programs that anyone can use*" were too vague for credit at this level of study.

(ii) Many of the answers seen here were far too generic and applied to any library, not specifically to a DLL. Vague statements about saving space or saving time were not enough. Answers should have said explicitly whether it was disk space or memory that was being saved and why time was saved.

(b) This was a question that needed to be read carefully. It asked candidates to identify which translator should be used when writing a program and then asked that their choice be justified. In other words, candidates were required to explain why they had chosen that particular translator. Many responses simply described the operation of a compiler or an interpreter without any reference to the coding stage or to why their chosen translator would be suitable. Vague answers

such as "*an interpreter makes debugging easier"* did not explain how or why it should be the translator of choice.

(c) Many candidates found it challenging to describe the four features of an IDE. There was little or no distinction made between writing the code and running the program or which features would be used at each stage. Many of the answers simply reworded the name of the feature, for example, "*a breakpoint is a point at which the program breaks"*. Answers for single stepping could often have applied equally to dry-running the code.

(d) There were some good, imaginative answers to this question. Some candidates needed to be careful about using brand names in their answers. Many of the responses included statements about training and/or learning and the use of a database of commands.

COMPUTER SCIENCE

Paper 9618/13
Theory Fundamentals

Key messages

At this level of study candidates are expected to demonstrate more than just general knowledge and should be able to provide more detailed and technical answers than would be expected at GCSE. It is very important that the correct technical terms are used appropriately. Many candidates would benefit from a better understanding of the different command words used in the questions. A question that begins with the command word 'explain' requires a different type of answer to one that begins with the command word 'state'.

Each question should also be read carefully and answered appropriately. For example, if a question asks for the benefits of a particular translator, no marks will be awarded for just describing how the translator operates. Marks will be awarded for describing the advantages of using that translator in the circumstances outlined in the question.

If a question asks for a given number of responses and the answer space is numbered, only the first response in each numbered space will be marked. Candidates should be encouraged to think carefully before starting to write their answer and to plan what they wish to write. In some cases when a question asked for two reasons, candidates listed many more than two.

General comments

Candidates should ensure their final answer is written legibly in pen, in the correct space on the question paper. If candidates wish to write rough versions of their answers first, they should be encouraged to use either the blank pages at the end of the script, or additional sheets of paper. In either case the rough work should be clearly crossed through.

Some questions required that working be shown. In these questions it is very important that candidates indicate clearly which value is meant to be the answer. Sometimes the working space was covered with numbers but none of them were labelled as the answer.

Questions about software licencing and the use of software to prevent threats to computer systems were usually completed successfully. Parts of the questions about relational databases and components of the CPU were answered less well.

Comments on specific questions

Question 1

(a) Some candidates found this question very challenging. There were many unnecessarily complicated incorrect expressions given as the answer.

(b) This question was generally answered well. Some candidates needed to improve their understanding of the operation of the XOR gate. The most frequent incorrect answer was the complete opposite of what was expected.

Question 2

(a) The difference in the area of coverage was frequently the first answer seen. Some candidates needed to understand that statements that simply repeat words from the acronym, such as "*a LAN covers a local area while a WAN covers a wide area*" were insufficient for credit at this level of

study. Candidates found the identification of a second difference more challenging. Again, vague statements such as "*a LAN is faster than a WAN*" were not enough. A better answer was "*the rate of transfer of data in a LAN is greater than the rate of transfer of data in a WAN*".

(b) (i) Many candidates were able to correctly identify the connections in this topology but found adding additional information for the second mark more challenging.

(ii) This question was answered well. The most common correct answers referred to the availability of alternative routes and the reduction in collisions.

(c) This question required responses that described a thin-client model, so it would be expected that there would be at least one statement about the role of the (thin) client and another about the role of the server in this model. Many responses were too vague and could apply to any general client-server model, not specifically a thin-client model.

(d) There were some excellent answers to this question. Most candidates were able to give some benefits of allowing students to use both wired and wireless connections.

(e) This question was challenging and many candidates needed to improve their understanding of IP addresses in subnetworks. There was considerable confusion with Public and Private IP addresses. Frequently the terminology was incorrect. Network IP and Host IP instead of Network ID and Host ID was a common error.

Question 3

(a) This question was answered well. There was some confusion between the two types of touchscreen and some candidates had them the wrong way round. Care needed be taken that exact terms were used. There were some very vague words used instead of 'co-ordinates'.

(b) There were some good, imaginative answers to this question. Many candidates were able to describe how the AI uses image recognition to focus on the pattern of a face.

(c) (i) The idea of sampling the sound and converting the samples to binary was described well. Candidates found it more challenging to describe how the samples were converted to a limited number of binary values, given by the sampling resolution, and then stored sequentially.

(ii) This was a question that needed to be read carefully. The question asked for reasons why increases in the sampling rate and sampling resolution would improve the precision of the sound recording. However, many candidates saw the two terms and simply wrote down the definitions without answering the question.

Question 4

(a) This question was answered well. Many candidates were able to correctly identify two ways in which a relational database addresses the limitations of a file-based approach. Some candidates found it more challenging to then expand on their answer.

(b) The foreign key was usually correctly identified and many candidates were also able to correctly describe what is meant by a secondary key. The entity was frequently incorrectly identified as an attribute and when describing a tuple there was considerable confusion between rows and columns. In questions like this candidates should be aware of the need to use correct terminology in the descriptions. It was not unusual to see the word 'database' used when candidates were actually referring to a table.

(c) There were a small number of completely correct answers to this question. Many candidates who gave the correct tables gave an incorrect primary key for the booking table because the start date was not included. Better answers included the booking ID field in the booking table. Some candidates did not remove the car attributes from the booking table into a separate table and instead implemented a CUSTOMER_BOOKING table which led to confusion.

(d) (i) This was a question that needed to be read carefully. The command word was 'describe' and the context was that of a car registration number. Just identifying a check was not enough for a description and generic descriptions of validation checks did not answer the question in context. "*A*

length check to ensure the registration number has the correct number of characters" was not enough. The format of the registration number was given in the question, so the length was known. A better answer was "*a length check to ensure the registration number has six characters*".

(ii) Similarly, just identifying a verification check was not enough for a description. Some candidates found it challenging to differentiate between double entry and a visual check and their descriptions could have applied equally to both.

(iii) This question was answered well. Most candidates were able to state that it would be because the original data was incorrect although it passed all the checks.

Question 5

(a) (i) This was a question that needed to be read carefully. Candidates were required to explain why the programmer would choose to use an interpreter while writing code. Many responses simply described the operation of an interpreter without any reference to the coding stage. Vague answers such as "*an interpreter makes debugging easier*" did not explain how or why it should be the translator of choice.

(ii) In this question too, candidates were required to explain why the programmer would choose to use a compiler when the program was complete. Many responses simply described the operation of a compiler without any reference to the fact that the program was complete. Vague answers such as "*a compiler is faster*" did not explain how or why it should be the translator of choice.

(b) This question was answered well. Most candidates were able to correctly identify an appropriate licence and say why it would be chosen. The most popular correct choice was a commercial licence. There was some confusion between freeware and free software.

Question 6

(a) Section 6.1 of the syllabus lists the security measures designed to protect computer systems which candidates are expected to be familiar. One of these is digital signatures. There were a small number of very good answers. However, very few candidates were able to explain how a digital signature can be used to authenticate a document and the most usual answer described some sort of scanned image of a traditional signature rather than the hashing of the document and its subsequent encryption and decryption.

(b) Almost all candidates were able to correctly identify two types of software that could be used to prevent threats to a computer system over a network. Very few candidates were able to give complete descriptions. At this level of study, it is expected that descriptions will be complete and inclusive, so it was not enough, for example, to simply say that an anti-virus software checks for viruses. The description needed to include something about scanning the computer, comparing against a database of viruses and deleting or putting anything suspicious into quarantine. Similarly, a description of a firewall needed to include a statement about incoming and outgoing traffic, comparison with set criteria and blocking or allowing the transmission.

Question 7

(a) This was answered well. Many candidates were able to correctly match the description to the correct denary value. Some candidates needed to be aware that the smallest number which can be represented in 8-bit two's complement binary is –128 not –127.

(b) This question was challenging for many candidates. These candidates needed to improve their understanding of how these components operate together inside the CPU. Answers frequently included statements such as "*the control bus carries the data*" or *"the control bus carries the instruction*" both of which are incorrect. A control bus carries (control) signals.

(c) This question was answered very well and there were many completely correct answers. Some candidates needed to take care with the positioning of the brackets in register transfer notation.

COMPUTER SCIENCE

Paper 9618/21

Fundamental Problem-Solving and Programming Skill

Key messages

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

This subject makes use of a technical vocabulary. These specific words and phrases need to be used correctly.

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 ensure they understand what is being asked when preparing their answer.

Candidates should also be aware that marks are unlikely to be gained from answers based on re-arranging words and phrases from the question without the application of computational thinking.

Centres are recommended to concentrate more on the following areas:

- the ability to describe an algorithm as a sequence of clearly defined steps
- the ability to write meaningful pseudocode

Many questions require students to write pseudocode. Candidates are required to write pseudocode and not use a programming language. Candidates need to follow the recommended pseudocode style 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.

A number of specific date functions are mentioned in the insert which candidates are expected to be familiar with and know how to use when writing algorithms to solve problems. In this paper **Question 2(b)** asks candidates to assign a date to a variable, however, many candidates did not use the correct date function provided on the insert.

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

A number of candidates make use of blank pages for rough work when preparing their final answer instead of an additional answer booklet. It is recommended that additional answer booklets are used not blank pages and the original answer is crossed out. The answer in the additional answer booklet should clearly indicate the question number and part being answered e.g. **8(a)**

The following comments relate to the use of pseudocode

Familiarity with fundamental programming concepts is vital. Lack of understanding is often illustrated the misuse of `OUTPUT` in place of `RETURN`.

Candidates need to be familiar with the use quotes when referring to literal strings for example when opening a file to read from / write to:

```
OPENFILE "Stock.txt" FOR READ
```

Functions and Operators

The functions and operators that are available for use in pseudocode answers are described in the insert which accompanies the paper and candidates are strongly encouraged to refer to this information when writing their answers. Candidates should be aware that the use of programming language-specific functions or methods that do not appear in the insert will not gain credit.

Comments on specific questions

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) Candidates found this question challenging with just over 15 per cent making no attempt to answer the question.

Candidates could have gained three of the four marks by just mentioning the specific IDE features that a typical Integrated Development Environment has that enables logical errors to be identified.

(b) Nearly half the candidates gained all three marks for this question.

(c) (i) Just over 75 per cent of candidates gained both marks for this question.

(c) (ii) Please note that due to an issue with **Question 1(c)(ii)**, full marks have been awarded to all candidates for this question to make sure that no candidates were disadvantaged.

Question 2

(a) Most candidates who attempted this question gained at least one mark with 25 per cent gaining all four marks.

The mark point that was most commonly awarded was for correctly obtaining the two required inputs. Many candidates found the most challenging part of the question was forming the loop so that it iterated the correct number of times.

(b) Around 40 per cent of candidates obtained all three marks available for this question with nearly 95 per cent of candidates making an attempt to answer the question.

Many candidates were not able to correctly declare a DATE variable, and fewer were able to use the `SETDATE()` function correctly although an example of how to do this was provided in the insert.

Question 3

(a) (i) This was a question regarding the advantages of using records. Just over 10 per cent of candidates obtained all three marks for this question with many candidates wrongly giving an array as the answer.

(a) (ii) Just over 10 per cent of candidates gained both marks available for this question. A small number of candidates lost marks for mentioning the use of a 2D array instead of just an array or preferably a 1D array.

(b) The majority of candidates gained one or two marks and over 40 per cent gaining all five marks available. A common omission from good answers seen was not mentioning that only the whole part of the amount entered is used in the calculation.

Question 4

This was the first question on the paper that required a pseudocode algorithm to be written. This was an accessible question with marks available for declaring variables and using straightforward loop and selection structures.

Just over 15 per cent of candidates did not attempt this question and a quarter of candidates gained all the six marks available.

A small number of candidates lost a mark for wrongly converting both characters being compared to the same case. The question made no mention of case conversion either explicitly or implicitly.

A few candidates did not make use of the sting handling functions from the insert but treated the string as an array when attempting to extract a character.

Question 5

(a) (i) Just under 20 per cent gaining both marks for this question.

Most answers showed a lack of understating of the different program development life cycles that this question was focusing on and were unable to give correct reasons why the waterfall model was not the most appropriate model to use in the context given.

(a) (ii) Just under a quarter of candidates correctly answered this question.

(b) A question covering a testing method from section 12.3 of the syllabus. Candidates should understand when each of these testing methods is appropriate and be able to describe them.

Many candidates gained at least one mark on this question by correctly identifying the testing method. Just under a third of the candidates gained all three marks available for this question.

Some candidates who correctly identified the testing method went on to describe a different method so gained no further marks.

Question 6

The second question on the paper that required a pseudocode algorithm to be written. This was a more challenging question compared to question 4. Although the underlying concept of calculating an average was straightforward the need to extract data from a 2D array and store the results in a 1D array added a layer of complexity that many students found too challenging.

A quarter of the candidates did not attempt this question with only 15 per cent obtaining all the six marks available. Most students did not recognise the need for a nested loop when summing each column of the 2D array although the diagram in the question indicated this would be required.

An understanding of creating algorithms using the fundamental programming constructs of loops and selection were being tested here along with accessing the elements of an array. Candidates need to ensure they are very familiar with how to write pseudocode programs which make use of these constructs.

Question 7

(a) This question focused on decomposition which is part of section 9.1 of the syllabus. The question involves candidates reading and understanding a scenario and applying decomposition to break down a problem into modules.

Just under half the candidates obtaining all three marks available for this question with most candidates being awarded at least one mark.

At times marks were lost for the justification being too vague although the module name given seemed appropriate.

(b) Both parts of this question carry on the theme of decomposition and focus on how structure charts are used in this process.

(b) (i) This question required candidates to show an understanding of a specific structure chart and the relationship between the modules depicted in the chart provided.

17 per cent of candidates obtained both the marks available for this question. Many of the answers seen were too vague such as “A loop will have Module A call the other Modules”.

To gain both marks the answer must make it clear that Module-A calls the other three modules and that this process is repeated.

(b) (ii) This question required candidates to draw a structure chart from the information provided. Most candidates missed the need to include a selection shape as indicated by the word ‘either’ in the phrase ‘Module-D() will call either Module-X() or Module-Y()’.

A third of candidates gained all three marks with many candidates gaining at least one mark for correctly drawing, labelling and showing the connecting line for the three modules.

Question 8

(a) Just over 10 per cent of candidates gained all seven marks for this question. Around 30 per cent of candidates did not attempt this question. Of those attempting the question many gained some marks.

The solution to this question required a series of selection statements, a simple loop to check each character in a string and the use of an existing function. The use of the latter was clearly indicated in the question.

The three-mark points most commonly awarded were:

- MP 1 – Checking the parameter is 5 characters long.
- MP 3 – Loop for length of parameter.
- MP 7 – Return of a Boolean value.

MP 6 was often not gained as not both upper and lower cases characters were checked.

MP 2 was often not awarded as candidates inserted CALL before the function call although they often tried to assign the return value to a variable.

(b) This was a challenging question requiring a candidate to write pseudocode to solve a problem that required the use of file handling, string manipulation along with a loop that ended when one of a number of conditions was met.

Just under 10 per cent gained all seven marks available for this question with around 30 per cent of the candidates not attempting to answer the question.

A number of candidates who attempted this question lost a mark for not putting quotes around the file name when writing the instruction to open and/or close the file.

The most commonly awarded mark points were:

- MP 2 – extracting a substring from the parameter.
- MP 3 – looping to the end of the file.

Very few candidates gained MP 4 (ending the loop when the new item is found) and MP 5 (ending loop when item being searched for was not in the file).

(c) (i) 20 per cent of candidates did not attempt this question. This area of the syllabus (section 12.3) on program testing covers a range of testing methods that candidates are required to understand.

Just over 15 per cent of candidates correctly answered this question.

(c) (ii) Another question on the same area as **8(c)(i)**. Only 72 per cent of candidates attempted this question with only 10 per cent achieving both marks.

'Stub testing' was correctly identified by a number of candidates. Other suggestions covered the range of possible testing methods.

Many answers described the replacement of the faulty module with dummy module, but fewer mentioned that these would generate a known or expected result.

Common mistakes included:

- suggesting that the dummy module would 'check' the program
- referring to a dummy program rather than a module
- describing integration testing rather than stub testing

(d) This was a one-mark question on the appropriate mode for opening a file. A quarter of the candidates gained this mark.

(e) This was the last question on the paper and was a challenging question requiring candidates to describe the benefits of a specific file organisation. Less than 5 per cent of candidates gained the three marks available for this question. However, many candidates gained one of the available marks.

COMPUTER SCIENCE

Paper 9618/22

Fundamental Problem-Solving and Programming Skills

Key messages

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

This subject makes use of a technical vocabulary. These specific words and phrases need to be used correctly.

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 ensure they understand what is being asked when preparing their answer.

Candidates should also be aware that marks are unlikely to be gained from answers based on re-arranging words and phrases from the question without the application of computational thinking.

Centres are recommended to concentrate more on the following areas:

- The ability to describe an algorithm as a sequence of clearly defined steps.
- The ability to write meaningful pseudocode.

The use of the insert and particularly the section relating to date functions should be emphasisied.

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

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

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

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

The following comments relate to the use of pseudocode

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

Functions and Operators

The functions and operators that are available for use in pseudocode answers are described in the insert which accompanies the paper and candidates are strongly encouraged to refer to this information when writing their answers. 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 concept of a function returning a value is not well understood.

As an illustration, given a function with header as shown:

```
FUNCTION Valid(InString : STRING) RETURNS BOOLEAN
```

Then a valid expression using this function would be:

```
IF Valid(MyString) = TRUE THEN
```

An example of invalid use seen frequently is:

```
CALL Valid(Mystring)
IF RESULT = TRUE THEN
```

There is also an increasing tendency to use the complete module header when using a function or procedure, for example:

```
IF Valid(MyString : STRING) : RETURNS BOOLEAN = TRUE THEN
```

Comments on specific questions

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) (i) Correctly answered by only a small number of candidates.

The use of a literal value together with the statement that the calculation was performed at various points in the program proved insufficient signposting for the majority of candidates, who failed to recognise the requirement for the use of a constant.

A common incorrect answer was to suggest the use of a `CASE` statement in place of the `IF ... ENDIF`, which suggests that candidates did not read the command sentence carefully enough. Others suggested the use of an integer or a string.

A small number of candidates did not attempt this question.

(ii) Candidates who identified the use of constants in **Question 1 (a) (i)** gained at least one mark, usually with a statement that mapped to MP2.

Very few three-mark answers were seen, suggesting that the use of constants is not widely appreciated.

Where the answer to **Question 1 (a) (i)** had identified the use of a string to hold the postal cost, the advantage suggested here was often that if a currency symbol was added as the first character it would be easier for the user to understand, ignoring the impact on any calculations.

(b) Very well-answered by the majority of candidates.

An occasional mistake was to cite programming features, such as `IF` statement, iteration and assignment.

(c) Well-answered by the majority of candidates.

An occasional mistake was to transpose the first and last answers.

Question 2

(a) Fewer correct answers than might be expected for this pseudocode assignment statement, as the insert included an example of what was required.

Centres could usefully emphasise the importance of the insert when answering pseudocode questions.

Statements often incorrectly included data types and/or parts of the function header as in the following example:

```
MyDob ← SETDATE(17, 11, 2007) RETURNS DATE
```

(b) Many correct answers, but a significant number included similar mistakes to that mentioned in **Question 2 (a)**.

A small number of answers suggested the use of string functions rather than date functions to extract the month from `MyDob`.

(c) A small number of very good answers were seen, giving steps that could have been easily mapped to a program.

Many answers failed to give a viable description of the required steps for this algorithm. Steps were sometimes vague and frequently repeated phrases from the question.

Common mistakes included:

- giving a textbook definition of an array rather than identifying the required attributes (size and datatype)
- suggesting that an array could be 'called' or that it would 'return a value'
- omitting the array initialisation, or referring to it only vaguely
- using imprecise phrases such as 'search for', 'find' or 'look-up'
- assigning values to the array (in Step3) rather than extracting a particular value
- the use of `RETURN` rather than `OUTPUT` for the identified value.

Question 3

(a) (i) A queue manipulation question, this was well-answered by many candidates and the operation of the circular queue was usually explained correctly.

Only a small number of answers addressed MP1.

References to push and pop operations were commonly seen, suggesting confusion between different ADTs.

Many candidates unnecessarily provided a textbook definition of FIFO.

A common mistake was to suggest that existing queue contents would have to be moved or removed before the new data items could be added.

(ii) Incorrect answers given were usually 10 or 5.

(b) As for **Question 2 (c)** above, a small number of very good answers were seen, giving steps that could have been easily mapped to a program.

A number of answers failed to give a viable description of the required steps in describing an algorithm. Steps were often vague and simply repeated phrases from the question.

Despite the final command sentence, a small number of answers consisted entirely of pseudocode.

Common mistakes included:

- not specifying a file open mode
- omitting any clear reference to a loop
- using vague phrases when referring to a file `READ` operation
- failing to make use of the given `AddToQueue()` function and instead often explaining the mechanics of the operation in unnecessary detail.

Question 4

Many full-mark pseudocode solution were seen for this algorithm.

MP2, MP5 and MP6 were commonly given.

Most candidates identified the need for a loop (MP3), and this was usually count-controlled.
Where a conditional loop was used a number of solutions only included the loop-counter increment within an `IF` statement so MP3 was not given. Occasionally a `FOR` loop incorrectly included an additional statement to increment the loop counter and several loops made an incorrect number of iterations.

Other mistakes included:

- giving the function return data type as `Count` rather than `INTEGER`
- using a keyword (e.g. `STRING`) as an identifier (MP1, MP3)
- including `INPUT` statements for the two parameters values (whether or not these had been included in the function header)
- use of case conversion prior to comparison, contrary to the requirement (MP4)
- use of `LEFT()` rather than `MID()` (MP4)
- attempting to extract a single character by treating the string as an array (MP4)
- use of `OUTPUT` rather than `RETURN` (MP6).

Question 5

(a) A small number of candidates gave a clear description of why passing the parameter by reference would lead to unexpected results.

Candidates needed to recognise that as this was a three-mark question, so simply referring to 'ByRef' without explanation would not gain full marks.

A number of answers suggested that the error was the use of different identifiers: `Num` (line 41) but `ThisNum` (line 50) suggesting that the parameter passing process is not properly understood.

Other mistaken causes of unexpected results included:

- comparing a variable of type `REAL` with an integer value (line 43)
- referring to the procedure `DisplaySqrt()` before it is declared.

(b) The answer 'there are no syntax errors' was seen on relatively few occasions.

Many answers were vague or inaccurate.

(c) A small number of good answers were seen, correctly describing three features.

A number of answers were suggesting that an IDE could automatically identify and correct the error or referring to features such as Pretty Print or AutoComplete which did not apply to the given scenario.

(d) A significant number of answers correctly suggested turning the statement into a comment.

Incorrect answers usually focused on stopping execution prior to the statement.

Question 6

(a) A small number of very good solutions were seen, often gaining all 6 marks.

There were a number of answers with candidates making little or no viable attempt at a solution.

The majority of candidates gained two marks or less. Several solutions included output of the 'hello ginger cat' strings used in the question to illustrate multi-line output.

MP1, MP2 and MP4 were often the only marks given.

A procedure header was usually included, but often the parameter or the corresponding `ENDPROCEDURE` statement was missing (MP1).

Most candidates recognised the requirement for a loop (MP2) and many of these included an attempt to construct a line. Many loops did not use the parameter value as the loop-counter upper value. Data type errors were seen in many cases. For example, attempting to concatenate an integer with a string or using the `+` operator in place of `&` (MP3).

MP4 was seen in many solutions, either as a separate loop or within a single use via the use of a suitable selection statement.

A common error was the attempt to build a string using the multiplication operator. This was not given credit.

For example:

```
OUTPUT Num, '*' * Num, Num
```

Solutions that attempted to use an array to represent the number square often made the mistake of attempting to output the whole array (or a complete row) with a single `OUTPUT` statement.

A number of answers employed a very repetitive but simpler selection-based solution. Often these were incomplete, either in terms of the selection clause or the number of squares correctly output.

Question 7

(a) (i) Many correct answers gained full marks for this abstraction question.

Marks were awarded across the range of MPs.

Where marks were lost it was usually from 'justifications'. There was a tendency to describe a piece of information rather than to justify why it would be needed in the new module.

(ii) Few correct answers.

As for **Question 7 (a) (i)**, marks were often lost due to poor 'justifications'. Simply stating that a particular item was not needed was not sufficient.

Marks were not given for reference to information that was not given in the question.

(b) This was attempted by virtually all candidates with full marks being gained by most.

Where marks were lost this was usually due to:

- failing to add the initial transition arrow
- omitting either the input or the output from the transition label
- placing the label far away from the transition arrow so that it was not clear what it referred to.

Although not directed by the question, some candidates chose to add an output to the first row of the table for the S1 to S2 transition. Where this output was added correctly on the diagram MP1 was given.

Question 8

(a) An exercise in string manipulation.

Many answers did not contain any elements of a viable solution.

This question assesses string manipulation techniques and a number of answers included irrelevant file operations.

Many candidates did not answer this question.

Common mistakes included:

- referring to identifiers that had not been assigned a value
- failing to extract fields correctly before testing them
- attempting to compare strings with integers
- treating `OnlyAlpha()` as a procedure by use of the `CALL` keyword
- failing to correctly combine the logical results of the individual tests.

Many solutions attempted to address the first mark point (MP1). Rather than simply check the length of the complete string, a common technique was to attempt to extract the `Description` field using the `MID()` function. Often the third parameter was given as the total length of the parameter string rather than the length minus 7.

MP2 and MP5 were the most accessible marks and were given in many solutions.

MP3 was often attempted by applying `IS_NUM()` to the first four characters. This was not given credit as the function would return `TRUE` if the number was decimal or if it was negative.

MP4 was omitted in many solutions. Where present, data type mismatches were common as was the tendency to omit the identifier in the second part of the comparison as shown in the following example:

```
IF Num >= 1 AND <= 5999 THEN
```

The function `OnlyAlpha()` was rarely used correctly (MP6).

Solutions that correctly addressed MP7 usually used either separate Boolean variables for each test or a single Boolean variable initialised to `TRUE` and then assigned the value `FALSE` for each test that failed. Solutions that attempted a nested `IF` often contained mismatched `ENDIF` statements.

In a number of cases the returned value depended only on the last test rather than on a combination of all tests.

A number of answers included the use of a comma separator in place of a logical `AND` as shown in the following example. This was not accepted for pseudocode.

```
IF A, B, C = TRUE THEN
```

(b) Very few viable answers seen. Many candidates did not answer this question.

The inclusion of many unnecessary substring operations from **Question 8 (a)** and parts of bubble-sort algorithms were often seen.

Common mistakes included:

- opening either file in the wrong mode and/or failing to close both files
- failing to enclose file names in quotes to indicate they are strings
- attempting a conditional loop until `EOF()` for a file that is open in `WRITE` mode

- use of the function `EOF()` without a parameter.

MP1 was rarely addressed correctly (see first two mistakes above).

MP2 was often addressed as many candidates recognised the need for a loop to read all the lines from `Stock.txt`.

Loop construct errors that were seen occasionally are illustrated as follows:

```
FOR n ← 1 to EOF("Stock.txt")
FOR n ← 1 to LENGTH("Stock.txt")
```

MP3 and MP4 were addressed correctly only in better solutions. When writing the new item string to file `NewStock.txt` it was common to omit also writing the line just read from `Stock.txt`.

MP5 was seen infrequently, and often resulted in no remaining file lines being read from `Stock.txt` after the new item string had been written.

MP6 was often given for viable solutions.

MP7 was rarely seen.

(c) 'Stub testing' was correctly identified by a reasonable number of candidates. Other suggestions covered the range of possible testing methods.

Many answers described the replacement of the faulty modules with dummy modules, but fewer mentioned that these would generate a known or expected result.

Common mistakes included:

- suggesting that the dummy module would 'check' the program
- referring to a dummy program rather than a module
- describing integration testing rather than stub testing.

COMPUTER SCIENCE

Paper 9618/23

Fundamental Problem-Solving and Programming Skills

Key messages

The programming language for this component is pseudocode. 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 may well be unacceptable for this component.

In some programming languages a simple change of case is sufficient to distinguish it from language keywords – candidates must be aware this is not allowed in the pseudocode used for this component.

Candidates need to have sight of the insert for the paper prior to the examination and be familiar with the various pseudocode functions. Answers would suggest the string handling functions are well understood – the 'date functions' very much less so.

General

The precise title of the paper is a good guide to what is required in candidate answers. For the longer questions – on this paper **Questions 6, 8(a)** and **8(b)** – the question requires both 'problem solving' skills – that is 'design skills' – and also the ability to express the design using the programming syntax in pseudocode which is specific to this component. A simple example would be as follows: Having established that for algorithm a loop is required, which of the three possible loop structures is most appropriate? Then having to decide, is it (say) a pre-condition (WHILE – ENDWHILE) loop or post-condition loop (REPEAT-UNTIL)? The candidate must finally be able to implement this with accurate pseudocode syntax. Often in candidate answers the pseudocode is lacking in some fundamental detail.

Question 1

(a) Generally, well answered. The question was assessing the candidate's understanding of the terminology associated with arrays.

(b) Most candidates stated that the issue was the difference in the bounds of the array as shown in the declaration statement and the assignment statement on line 101. As the question was ambiguous as to whether the error was on line 10 or line 101, a range of answers were acceptable for the second mark.

(c) Well answered. Candidates appreciated that the Sheet 4 array was storing integers.

(d) This was poorly understood. The concept of 'input-processing-output' is fundamental to most computer applications. Candidates need to appreciate that the terms input and output are used as descriptors for the reading and writing of data from and to a data file (rows three and four in the given table).

Question 2

(a) There were a number of answers which displayed a lack of understanding. Candidates need to be confident with the use of the date functions.

(b) (i) Many candidates correctly identified that it would be the day and month component of the date of birth which could be verified independently. Precise answers were required such as *'the month is*

an integer value in the range 1 to 12, Explanations for the year value either correctly stated it must be prior to 2003 or, expressed a calculation subtracting 18 from 2020.

(ii) This was less well answered. The answer often correctly stated that the two dependant components would be the day and the month, but then failed to express that given a particular month, the day value should be checked or vice-versa for the second mark. Many answers extensively described the various months having a differing number of days without an explanation that a check must take place between the day and month and gained no credit.

Question 3

(a) (i) Most candidates correctly stated that six pop operations would take place before an error is generated.

(ii) Generally, well answered but there were some common errors which included omission of the two pointers. The most common error was to not display the AAA and BBB values remaining in locations 504 and 505 respectively. This is a subtle point for candidates to understand that following a pop operation, the values remain in memory but are effectively 'bypassed' by the pointers.

(b) (i) Candidates often described the power being switch off which was not relevant here. The answer sought was the understanding that the next time the program is run the values comprising the final state of the stack from the previous execution of the program must be recovered or restored.

(ii) There were two distinct correct approaches here. The candidate who described a 'pop' operation would have had less to write as a 'pop' operation assumes that the value is read from the top of stack position and top of stack is then decremented.

A fundamental point to score marks was that the values on the stack are being read one at a time. Answers often stated that *'the values on the stack are written to the file'* so lacking any detail. Both approaches required the fundamental points that:

- the algorithm should first check for an empty file (this was rarely seen)
- the file is opened in write mode (well answered)
- a loop is required to process successive values; expressed either using the word 'loop' or a description (say) *back to Step 2*.

Question 4

Note the points made in the opening General section. A very common error here was to use the identifier name Char for one of the parameters. Char is one of the pseudocode data types and therefore cannot be used in the pseudocode as an identifier.

This question was generally well answered with only minor errors in the syntax the cause of lost mark(s). Examples included the incorrect operator used for string concatenation and the incorrect use of OUTPUT instead of RETURN.

Question 5

(a) Very few candidates stated that the key requirement would be the original program source code. Better answers stated that the key requirement was to test every possible path through the code. Many peripheral points were also able to score such as the drawing up of a trace table where various inputs would determine the resulting output. This is a testing fundamental which candidates should be encouraged to adopt if they are exposed to practical programming to support the fundamental design structures contained in this component.

(b) Well answered with perfective maintenance the correct (and only) answer.

Question 6

(a) Generally, well answered. Most answers correctly had a correct design with the use of a FOR–END loop with the correct boundaries. Also, the syntax for the correct function header with its two

parameters was usually in place. Some candidates it would appear have been taught to immediately write the ENDPROCEURE following the header lest they should forget. The correct design then required using various string handing functions to isolate the two required numbers and then convert them to a numeric value in order to perform the addition and the conditional statement. Answers which incorrectly tried to isolate the numbers using the MOD and DIV functions were still able to score five of the seven available marks.

(b) Despite a good understanding of a function header in **part (a)** this did not carry through to **part (b)**. A re-designed function header or a clear description would have secured 2 of the available 4 marks. The pseudocode syntax or a description often omitted to state the two parameters would be integers. The question then required two advantages which would result from the redesigned function and answer responses here were generally weak.

Question 7

(a) (i) Generally well answered with candidates stating the concept required is to focus on the essential data required for the new module.

(ii) Well answered with the most popular correct answers the email address of the candidate and for the not required item, their subject choices. Various others were also acceptable. A justification was required to secure each mark. For example, *'In order to send an email, the candidate's email address must be known'.*

(iii) This final part was less well answered as candidates were sometimes not clear what the word 'operation' in the question stem was indicating. Clear answers were typically:

'The status of the candidate must be changed in order that they are now able to continue making new loans'. Or, *'The status of the book must be changed to indicate the book is now available for a new loan'.* Answers which indicated there would be a change to a single data item did not gain credit.

(b) Well answered. Candidates were able to deduce from the various procedure headers the ordering of the modules in the structure chart. The different arrow notation (clear circle, shaded circle or double arrow) was well understood.

Question 8

(a) A number of candidates did not gain credit due to inaccuracies and omissions in the syntax of the pseudocode. Examples included:

- the omission of quotation marks for the literal filenames
- missing brackets when using the EOF function.

To avoid the first error some candidates first declared a variable of type STRING which was then assigned the 'Stock.txt' filename, so that later when the file was referenced in the file handling commands, the variable name could be used in place of the literal string. In general, the string handling functions were correctly used in the four parts of the design where they were needed; to isolate the supplier code, the product code and the description. The functions were finally needed to construct the single string which needed to be written to the NewStock.txt file. A common error was to write this string to the file only for the case where the required product code (the function parameter) was matched and failed to write anything for a 'no-match'.

Most answers successfully initialised the count, incremented it when a matching code was found and returned this value outside the loop.

An alternative correct design was to store the matching item data in an arrays as the file was read. Then all the output is produced after the file is closed and required a second loop for the item data output. This approach would have served the candidate well for **parts (b)** and **(c)** which followed.

(b) Most candidates appreciated that the parameter value for the supplier code was needed for the initial output and the syntax for this outline line was usually correct. A common error was to confuse which output lines would be before any loop is formed, inside the loop, or after the loop terminates.

The same use of the string handling functions was needed as for **part (a)** and candidates generally mastered the syntax and the logic. Again, the use of the count was well designed and implemented. Alternative syntax was acceptable for the formation of the output lines; either using the comma separator for the various constituent parts, or the ampersand character to concatenate the various parts. However, if the second approach was used candidates needed to be aware that any numeric value (i.e. the count variable) would require the NUM_TO_STR function in the OUTPUT statement.

(c) (i) Candidates needed to express the key issues in the design that the re-arrangement of the output would cause. The count can only be calculated when all the file items have been read. So, the solution required either that the file must be read a second time, or the items must in some way be stored.

(ii) Clear answers described the items stored in an array (1D or 2D) as each item from the file is read. Candidates then gained credit for describing the correct sequence to produce the output:

- each item is read from the file
- if the supplier code matches then
- increment the count
- store the item data in the array(s).

When the EOF is reached.

- output the header including the count
- set up a loop to output the stored items from the array(s).

Some candidates stored the items in a second file instead of array(s) and this gained credit.

COMPUTER SCIENCE

Paper 9618/31
Advanced Theory

Key messages

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

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

Candidates 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.

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 to maximise their mark.

Questions asking for benefits and drawbacks require statements about the item under consideration only. Comparisons with other similar concepts, or advantages and disadvantages compared with these, do not answer the question. For example, **Question 5(b)** required benefits and drawbacks of circuit switching, so only circuit switching needed to be mentioned. Some candidates chose to compare circuit switching and packet switching. Similarly, **Question 9(b)** asked for benefits and drawbacks of quantum cryptography, which did not require a comparison with other types of cryptography, such as asymmetric, nor did it require a list of features of quantum cryptography.

Questions asking about the purpose of something, such as **Question 9(a)**, which required a description of the purpose of asymmetric key cryptography cannot be adequately answered by simply describing the process of asymmetric encryption or by listing its features.

Comments on specific questions

Question 1

(a) The majority of candidates achieved at least one mark for this question, with many achieving full or nearly full marks.

(b) Most candidates explained the fact that the mantissa in system 2 only had 8 bits available, so it was too small to represent the whole binary number, which required 10 bits. Those who went on to correctly state that the number would lose precision, also achieved the second mark.

Question 2

(a) The full range of marks was seen for this question with many candidates achieving full marks.

(b) Most candidates were able to describe the purpose of both the A* algorithm and Dijkstra's algorithm as being used to find the optimum path between two nodes on a graph, or similar.

Question 3

(a) The majority of candidates stated one or both correct hash values.

(b) Candidates generally seemed to find this question difficult. However, some good responses were seen and the full range of marks was awarded. Although not strictly required to achieve the marks, many candidates chose to include the concepts of open and closed hash in their responses. Many of these were not correct, with closed hash, for example, being used to describe the process of searching an overflow area, when it should be searching the file linearly for the next available space. Open hash is applied to an overflow area.

Question 4

(a) Many candidates achieved one or both marks for displaying their ability to declare an enumerated data type that holds a set of primary numbers.

(b) Candidates generally found this question a little more difficult and often gave another enumerated data type as the answer. Candidates who recognised that a pointer data type was required and gave a complete answer, achieved the marks.

Question 5

(a) Most candidates were able to state, with a reason, where circuit switching would be used, and achieved one or both marks.

(b) Many candidates were able to give the correct benefits and drawbacks of circuit switching. Some answers erroneously compared circuit switching with packet switching, which was not required in this question.

Question 6

(a) Most candidates identified the fact that the password `DPAD99$` was valid and that the passwords `DAD$95WY` and `ADY123?` were invalid. Those candidates who also gave clear reasons for this achieved the marks.

(b) A high proportion of candidates whose responses accurately matched the syntax diagrams for `<symbol>` and `<letter>`.

(c) The full range of marks was seen for this question, with many interesting designs of syntax diagram for an identifier shown by candidates. Any diagram that allowed any number of letters, followed by any number of digits, including no digits, in the correct sequence and syntax, achieved the marks.

Question 7

(a) Candidates who correctly populated the whole Karnaugh map with 0s and 1s, leaving no gaps and without using other characters, achieved both marks.

(b) Candidates who correctly identified two overlapping groups of four 1s and drew loops around them achieved both marks.

(c) Candidates who correctly wrote the simplified sum-of-products to represent both correct loops identified on the Karnaugh map achieved both marks.

(d) Candidates who correctly simplified their sum-of-products from part **(c)** as a Boolean expression in its lowest terms achieved the mark.

Question 8

Most candidates achieved at least one mark for this question. Candidates who outlined the characteristics of massively parallel computers as a large number of computer processors working collaboratively and connected using a network infrastructure, or similar, achieved full marks.

Question 9

(a) Candidates who correctly described the purpose of asymmetric cryptography with answers, such as, to provide better security by using a public and a private key, achieved the marks. Some incorrect responses gave features of asymmetric encryption, or described how it was used.

(b) Candidates who identified appropriate benefits and drawbacks of quantum cryptography achieved between one and four marks. However, some answers erroneously compared quantum cryptography with other types of encryption, or simply listed features, without clearly stating the benefit or drawback.

Question 10

Most candidates achieved at least one mark for this question. Many candidates performed very well, achieving full or nearly full marks.

Question 11

(a) Most candidates achieved at last one mark for correctly defining the three integer identifiers. Some candidates also correctly defined either the array or the constant, but candidates who correctly defined all three parts were rare.

(b) A decision was made to award all candidates four marks for this question, because the question had not made it clear that `FrontPointer` was assumed to be updated before the function was called, and therefore did not need to be updated in the pseudocode provided. This may have confused some candidates. The published version of the paper was corrected by adding an extra response line for candidates to update the `FrontPointer` in the code provided.

(c) Candidates were asked how a new element could be added to a queue if the queue was implemented as two stacks, to demonstrate their understanding of the management of these Abstract Data Types. Several interesting responses were seen with some candidates achieving high marks. Candidates generally understood the principal that a queue required data elements to be removed from the queue in the order they were added, which required some moving of the data between the two stacks for this to be achieved correctly.

Question 12

(a) The majority of candidates understood that recursion involved a function or procedure defined in terms of itself and achieved at least one mark for this. Many of these candidates were also able to achieve the second mark by expanding on this, for example by describing the use of a base case.

(b) Candidates generally appeared to find the trace of the given recursive function difficult. However, a significant number of candidates achieved four or five 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 practised 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 in 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 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.

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 to maximise their mark. For example, questions asking for differences, require comments to be made on both items that are being compared. **Question 5(b)** required comments about both symmetric encryption and asymmetric encryption, to explain the difference between them. This could not be done by simply describing features of one of the given types of encryption. Similarly, **Question 7(b)** required comments about both circuit switching and packet switching to show differences between them. In addition, questions asking about the purpose of something should be answered precisely. For example, **Question 5(a)** which required a description of the purpose of quantum cryptography could not be adequately answered by simply describing what the term 'quantum cryptography' means or listing features of it.

Comments on specific questions

Question 1

(a) The majority of candidates achieved at least one mark for this question. Candidates who also correctly stated the exponent and gave evidence of working achieved both marks.

(b) Most candidates achieved some marks for this question. High marks were achieved by candidates who were specific in the number of bits available for the storage of the mantissa, in addition to including the consequence of the available mantissa size being too small.

Question 2

(a) The majority of candidates achieved some marks for this question, with candidates who described the organisation and access of sequential files in terms of the records achieving the highest marks.

(b) The majority of candidates stated one or both correct hash values.

Question 3

(a) Most candidates identified the fact that the variable `9SW` was invalid and `UWY` was valid, with candidates who also gave clear reasons for why this was the case achieving the marks.

(b) This was generally a well answered question, with candidates whose responses accurately matched the syntax diagram for a variable achieving the highest marks.

(c)(i) Candidates whose responses matched the requirements of the question; a vehicle registration with two letters followed by one, two or three digits, along with the available letters and digits from the syntax diagrams, achieved the mark.

(ii) The full range of marks was seen for this question, with many interesting designs of syntax diagram shown by candidates. Any diagram that only allowed two letters, with at least one digit, but no more than three digits, in the correct sequence and syntax, achieved the marks.

Question 4

Nearly all candidates achieved at least one mark for this question, with the full range of possible marks seen.

Question 5

(a) Candidates who described the purpose of quantum cryptography, such as 'to produce a virtually unbreakable encryption system', rather than simply describing quantum cryptography or how it works, achieved the marks.

(b) Candidates who gave differences between symmetric and asymmetric encryption by including both encryption types in their response achieved marks. However, many responses described only one difference in detail, rather than giving a comprehensive answer with several differences.

Question 6

(a) Many good responses were seen for this question with a wide range of marks awarded. Candidates who included all the required data with correct data types, such as the use of `STRING` for the telephone number, as well as correct key words such as `TYPE`, `ENDTYPE` and `DECLARE`, all formatted correctly, achieved the highest marks.

(b) Most candidates achieved at least one mark for this question, with many going on to achieve higher marks. Candidates whose responses more closely followed the examples given in the published pseudocode guide achieved the highest marks.

Question 7

(a) This question was answered well, with many appropriate examples of where packet switching would be used seen.

(b) Candidates were asked to give four differences between circuit switching and packet switching, which meant that answers showing how the difference applied to both switching types were required. Many correct answers were possible and seen, and candidates stating clear differences that were also not alternative phrasings of the same difference achieved the best marks.

Question 8

(a) Many candidates achieved at least one mark for this question, which asked candidates to describe the use of pipelining in Reduced Instruction Set Computers. This required a response and expansion based on the idea that pipelining allows multiple instructions to be processed simultaneously.

(b) This was usually answered well with many different techniques seen to show how six instructions that had each been divided into five stages would be completed in the minimum number of clock cycles. Responses needed to clearly identify each of the six instructions as they passed through the five stages.

Question 9

(a) Candidates who correctly identified and wrote the six Boolean terms where the output from the truth table would be 1, as a sum-of-products, achieved the marks.

(b) Candidates who correctly populated the whole Karnaugh map with 0s and 1s, leaving no gaps and without using other characters, achieved both marks.

(c) Candidates who correctly identified two overlapping groups of four 1s and drew loops around them achieved both marks.

(d) Candidates who correctly wrote the simplified sum-of-products to represent both correct loops identified on the Karnaugh map achieved both marks.

(e) Candidates who correctly simplified their sum-of-products from **(d)** as a Boolean expression in its lowest terms achieved the mark.

Question 10

(a) Many correct answers were possible and were seen for this question. The most common correct answers were supervised learning, unsupervised learning and deep learning.

(b) Many candidates made good attempts at this question with the vast majority achieving some marks and many candidates achieving high marks. Most candidates gave complete or partially complete examples of their working and achieved credit for this. Some candidates also managed to give the final path without showing a fully complete set of working. However, the strongest responses included both the correct final path along with all the working required to show how this was achieved.

Question 11

(a)(i) Very few errors were seen in this question. Where there was an error, it was most likely due to not using the given identifiers from the pseudocode within the response.

(ii) This question was usually answered well with many candidates achieving high marks. Common errors seen included the incorrect initialisation of `RearPointer` to 0, or an incorrect `IF` statement, such as checking if `Length < MaxLength -1` instead of `Length < MaxLength`.

(b) Most candidates achieved at least one mark here. Candidates who achieved the best marks correctly identified the fact that print queues in general are expected to output the print jobs in the order they are received. They then went on to explain how print jobs would be handled if the print queue worked as a queue ADT, compared with how the print queue would operate if it was a stack ADT, to illustrate why the queue ADT would be the better choice.

COMPUTER SCIENCE

Paper 9618/33
Advanced Theory

Key messages

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

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

Candidates 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.

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 to maximise their mark.

Questions asking for benefits and drawbacks require statements about the item under consideration only. Comparisons with other similar concepts, or advantages and disadvantages compared with these, do not answer the question. For example, **Question 5(b)** required benefits and drawbacks of circuit switching, so only circuit switching needed to be mentioned. Some candidates chose to compare circuit switching and packet switching. Similarly, **Question 9(b)** asked for benefits and drawbacks of quantum cryptography, which did not require a comparison with other types of cryptography, such as asymmetric, nor did it require a list of features of quantum cryptography.

Questions asking about the purpose of something, such as **Question 9(a)**, which required a description of the purpose of asymmetric key cryptography cannot be adequately answered by simply describing the process of asymmetric encryption or by listing its features.

Comments on specific questions

Question 1

(a) The majority of candidates achieved at least one mark for this question, with many achieving full or nearly full marks.

(b) Most candidates explained the fact that the mantissa in system 2 only had 8 bits available, so it was too small to represent the whole binary number, which required 10 bits. Those who went on to correctly state that the number would lose precision, also achieved the second mark.

Question 2

(a) The full range of marks was seen for this question with many candidates achieving full marks.

(b) Most candidates were able to describe the purpose of both the A* algorithm and Dijkstra's algorithm as being used to find the optimum path between two nodes on a graph, or similar.

Question 3

(a) The majority of candidates stated one or both correct hash values.

(b) Candidates generally seemed to find this question difficult. However, some good responses were seen and the full range of marks was awarded. Although not strictly required to achieve the marks, many candidates chose to include the concepts of open and closed hash in their responses. Many of these were not correct, with closed hash, for example, being used to describe the process of searching an overflow area, when it should be searching the file linearly for the next available space. Open hash is applied to an overflow area.

Question 4

(a) Many candidates achieved one or both marks for displaying their ability to declare an enumerated data type that holds a set of primary numbers.

(b) Candidates generally found this question a little more difficult and often gave another enumerated data type as the answer. Candidates who recognised that a pointer data type was required and gave a complete answer, achieved the marks.

Question 5

(a) Most candidates were able to state, with a reason, where circuit switching would be used, and achieved one or both marks.

(b) Many candidates were able to give the correct benefits and drawbacks of circuit switching. Some answers erroneously compared circuit switching with packet switching, which was not required in this question.

Question 6

(a) Most candidates identified the fact that the password `DPAD99$` was valid and that the passwords `DAD$95WY` and `ADY123?` were invalid. Those candidates who also gave clear reasons for this achieved the marks.

(b) A high proportion of candidates whose responses accurately matched the syntax diagrams for `<symbol>` and `<letter>`.

(c) The full range of marks was seen for this question, with many interesting designs of syntax diagram for an identifier shown by candidates. Any diagram that allowed any number of letters, followed by any number of digits, including no digits, in the correct sequence and syntax, achieved the marks.

Question 7

(a) Candidates who correctly populated the whole Karnaugh map with 0s and 1s, leaving no gaps and without using other characters, achieved both marks.

(b) Candidates who correctly identified two overlapping groups of four 1s and drew loops around them achieved both marks.

(c) Candidates who correctly wrote the simplified sum-of-products to represent both correct loops identified on the Karnaugh map achieved both marks.

(d) Candidates who correctly simplified their sum-of-products from part **(c)** as a Boolean expression in its lowest terms achieved the mark.

Question 8

Most candidates achieved at least one mark for this question. Candidates who outlined the characteristics of massively parallel computers as a large number of computer processors working collaboratively and connected using a network infrastructure, or similar, achieved full marks.

Question 9

(a) Candidates who correctly described the purpose of asymmetric cryptography with answers, such as, to provide better security by using a public and a private key, achieved the marks. Some incorrect responses gave features of asymmetric encryption, or described how it was used.

(b) Candidates who identified appropriate benefits and drawbacks of quantum cryptography achieved between one and four marks. However, some answers erroneously compared quantum cryptography with other types of encryption, or simply listed features, without clearly stating the benefit or drawback.

Question 10

Most candidates achieved at least one mark for this question. Many candidates performed very well, achieving full or nearly full marks.

Question 11

(a) Most candidates achieved at last one mark for correctly defining the three integer identifiers. Some candidates also correctly defined either the array or the constant, but candidates who correctly defined all three parts were rare.

(b) A decision was made to award all candidates four marks for this question, because the question had not made it clear that `FrontPointer` was assumed to be updated before the function was called, and therefore did not need to be updated in the pseudocode provided. This may have confused some candidates. The published version of the paper was corrected by adding an extra response line for candidates to update the `FrontPointer` in the code provided.

(c) Candidates were asked how a new element could be added to a queue if the queue was implemented as two stacks, to demonstrate their understanding of the management of these Abstract Data Types. Several interesting responses were seen with some candidates achieving high marks. Candidates generally understood the principal that a queue required data elements to be removed from the queue in the order they were added, which required some moving of the data between the two stacks for this to be achieved correctly.

Question 12

(a) The majority of candidates understood that recursion involved a function or procedure defined in terms of itself and achieved at least one mark for this. Many of these candidates were also able to achieve the second mark by expanding on this, for example by describing the use of a base case.

(b) Candidates generally appeared to find the trace of the given recursive function difficult. However, a significant number of candidates achieved four or five marks.

COMPUTER SCIENCE

Paper 9618/41
Practical

Key messages

Centres need to make sure that the only document submitted is the evidence document, in a Word document or pdf form. Zip folders should not be uploaded, neither should images showing the candidates, or the program code produced. Additional documentation will not be marked and can cause a delay in the marking of the scripts.

Candidates need to copy their program code into the evidence document. This needs to be copied as text and not as a screenshot. Screenshots are not always clear and when there is coloured text on a black background the text may be unreadable. If the screenshot is not legible then the answer will not be awarded the marks.

Screenshots must be used for the outputs. Candidates need to make sure these are black text on a white background. The screenshots need to be legible and sufficiently large to clearly read the text. Candidates must make sure that all of their answer is visible, if any of it is off the page and cannot be viewed it will not be able to be marked.

Candidates need to adapt their solutions for their chosen language. If a record is being used and the language does not support a record, then an appropriate alternative needs to be selected.

General comments

Candidates must make sure they are answering the question given, for example producing the data structure that has been asked for, instead of creating one they have used before. For example, if a queue is being used the question will state whether the tail pointer points to the last item of the next empty space.

Comments on specific questions

Question 1

This question required candidates to read data from a text file into a 1D array and use a linear search on the contents.

(a) (i) Many candidates were able to accurately define an array. Some candidates only provided a comment to state their intention but did not create an actual array (or list where appropriate).

(ii) Candidates were often able to open the text file and read in the data. There was a mix of approaches to checking when to stop, including looping the number of times there were elements, looping until the end of file and reading all data into a list that was then iterated through. Some candidates also closed the file, but this was fewer than those who opened the file. Some responses included appropriate exception handling.

(b) (i) This question required candidates to create a procedure to output the array contents on single line. Many candidates were able to output the array, but some candidates did this in the main program instead of a procedure.

(ii) Many candidates were able to call the function they defined in **part (b)(i)**.

(iii) Many candidates produced a screenshot showing the correct result. Some candidates produced one screenshot, but it was not all visible in the evidence document. Outputs can be given using two screenshots if they would not be visible with one.

(c) This question required candidates to write a function that performed a linear search on an array to find the number of times a specific value was stored in the array. The stronger candidates were able to create a function that took appropriate parameters and then looped through each element in turn, keeping a count of the number of times the parameter occurred. Some weaker responses took the value to find as an input instead of a parameter, some responses also returned the index where the item was found instead of counting the number of times it occurred.

(d) (i) Many candidates were able to read the data value as input. The weaker responses checked if the value was a number, whilst the stronger responses either cast the value as an integer or checked whether it was a whole number, as well as checking the correct values.

Candidates were often able to call their method from **part 1(c)** but fewer output the return value in an appropriate message. Some candidates attempted to include the message within their function in **part (c)** and some candidates did not store the return value from the search.

(ii) Many candidates were able to produce the correct output. Some candidates did not input the number 12 as required by the question and hence did not produce the required output.

Question 2

This question required candidates to use object-oriented programming to create classes and objects.

(a) (i) Many candidates were able to accurately declare the class and constructor. The class design identifies the values to initialise attributes to in the constructor. The stronger candidates followed this and initialised the current speed to 0 and the horizontal position to 0.

(ii) Candidates were often able to create the required get methods. A common error was sending a parameter to the get method or outputting the value instead of returning it.

(iii) Many candidates could demonstrate an understanding of set methods, they were able to create the required methods, taking a parameter and stored the value correctly. Some of the weaker responses attempted to read in a value from the user within the function or returned the parameter instead of the attribute.

(iv) Candidates were often able to define the method accurately, taking the correct parameters. Due to this being a method, Python programmers need to include self with the parameters.

Many candidates correctly added the speed to the horizontal position, but fewer accurately limited the speed. A common error was not increasing the speed at all if it would go above the maximum, instead of limiting it to the maximum speed.

(b) (i) This question required an understanding of inheritance. Candidates were required to declare a class that inherited from the one they made in **part (a)**. The stronger responses used inheritance accurately and called the parent class constructor within the constructor. Some candidates created a new class and redefined the attributes instead of using the parent class.

(ii) In this question candidates needed to override the method from the parent class. Some candidates were able to do this accurately by overriding the parent method. The stronger responses also called the parent method to avoid the repetition of code. Many candidates were able to add the vertical change to the position, but fewer accurately stopped it from exceeding its maximum speed. Some responses did not attempt to update the horizontal position or speed if the vertical change was too high, instead of limiting it and then updating the speed and position.

(c) There were several different ways of approaching this Questions. 1: A method in each class could be used that outputs the data for the helicopter and then calls the parent method for the horizontal position and speed. 2: One method that tests what type of object is being output and then only outputs the vertical position if the object is of type Helicopter. 3: Using exception handling to attempt to output the vertical position but always outputting the horizontal position and speed.

Many candidates were able to identify one of these methods. Few candidates were able to do this all entirely accurately, with a common error being just outputting all 3 values no matter what the type of object.

(d) (i) Candidates were commonly able to create the required objects with the correct values. Some candidates did not call the required methods the number of times required for each object or called a procedure instead of a method for each object.

(ii) Some candidates were able to produce the correct output.

Question 3

This question required candidates to implement two stacks using an array. They had to push and pop data into and from the stack.

(a) Many candidates were able to declare the arrays and initialise the pointers to 0. Where candidates are provided with the initial values, they need to use these in the question, for example initialising the top pointers to 0. Some candidates initialised them to –1.

When declaring an array in Python candidates need to create the list, a comment is suitable to identify the number of elements and data type, but an empty (or populated) list is still required.

(b) (i) Candidates were provided with the pseudocode algorithm for the push function. Candidates needed to convert this into their chosen programming language. Many candidates were able to do this accurately.

(ii) As with **part (b)(i)** candidate were provided with the pop function. Candidates were required to convert this into their chosen programming language. Many candidates were able to do this accurately.

(iii) Candidates were provided with a text file to read data in for the Animal stack. The question required exception handling and some candidates were able to use this appropriately. When using exception handling with text files candidates must make sure the opening and closing of the file is within the try clause. Otherwise if the try fails and the program then attempts to close a file that is not open it could produce an error. Some candidates opened the file but did not close the file.

(iv) The stack for the colours works in the same way as for animals. Candidates needed to replicate the code and adapt it for the colour array. This meant that the pointer identifiers needed changing and the value of the top pointer needed amending to check whether the stack was full as it has a different number of elements.

(v) The data for the colour stack was provided in a text file and candidates need to repeat their code from reading the animal data file. Candidates needed to make sure they made adjustments, so it was appropriate for this stack. The adjustments included opening the ColourData.txt file, looping 10 times (if not looping until end of file) and using the correct push function. Some candidates did not adjust the number of records being read in or called PushAnimal instead of PushColour.

(c) This question required application of the stacks using the pop functions declared. If there was both an animal and colour, then they were both output. If there was only a colour (no animal) then the text ‘No animal’ was output and the colour was pushed back onto the stack. The reverse happened if there was only an animal.

There were a range of attempts at this question. Some candidates output the return value without checking them, so if it ran out of colours it would have produced an error. They often did then go on to make the appropriate checks.

Some candidates output the correct message but pushed the animal onto the incorrect stack or did not push it.

The stronger candidates had a clear structure in their response where they checked if each element was present in turn and then finally outputting if both were present.

Candidates did not have to consider what would happen if there were neither present, because it would have been caught by the no colour or no animal. The stronger responses did, however, consider this and output an appropriate message.

(d) (i) This question was often answered well by candidates who attempted it. They called the ReadData once and then called OutputItem four times (sometimes manually calling it 4 times and sometimes in a loop).

(ii) Many candidates who gave an answer to this question produced the correct output.

COMPUTER SCIENCE

Paper 9618/42
Practical

Key messages

Centres need to make sure that the only document submitted is the evidence document, in a word document or pdf form. Zip folders should not be uploaded, neither should images showing the candidates, or the program code produced. Additional documentation will not be marked and can cause a delay in the marking of the scripts.

Candidates need to copy their program code into the evidence document. This needs to be copied as text and not as a screenshot. Screenshots are not always clear and when there is coloured text on a black background the text may be unreadable. If the screenshot is not legible then the answer will not be awarded the marks.

Screenshots must be used for the outputs. Candidates need to make sure these are black text on a white background. The screenshots need to be legible and sufficiently large to clearly read the text. Candidates must make sure that all of their answer is visible, if any of it is off the page and cannot be viewed it will not be able to be marked.

Candidates need to adapt their solutions for their chosen language. If a record is being used and the language does not support a record, then an appropriate alternative needs to be selected.

General comments

Candidates must make sure they are answering the question given, for example producing the data structure that has been asked for, instead of creating one they have used before. For example, if a queue is being used the question will state whether the tail pointer points to the last item of the next empty space.

Comments on specific questions

Question 1

This question required candidates to create, populate and manipulate a 1D array.

(a) Many candidates were able to declare an array or a list where candidates were using Python. The stronger candidates clearly indicated how this array had 10 string elements; through declaration or initialisation. Weaker responses initialised the data with integers, or null values, instead of strings, or only gave a comment without actually creating an array structure.

(b) This question required candidates to store the given data in the 1D array created in part a. Many candidates were able to store the data in individual elements. Some weaker responses did not store the data as strings or stored all of the data in one string inside one element. When copying data candidates must make sure they are exact, checking the spelling and order of the data given.

(c) This question provided candidates with an incomplete sorting algorithm. Candidates were required to identify the missing code and then rewrite the algorithm in their chosen language. Some candidates also included the completed pseudocode which is not required, only the final solution is required. When candidates are provided with an algorithm, they need to use the same structure and format as the algorithm. Some candidates made significant changes to the algorithm and changed its functionality, for example taking a parameter and attempting to return a data item.

Some candidates found the string manipulation functions challenging. The pseudocode algorithm uses LENGTH and MID as these are in the pseudocode guide. Candidates needed to identify their language equivalent functions or features. Some candidates repeated the pseudocode functions, whilst some candidates created a function named MID to perform the same function; when they could have used the in-built language tools.

(d) (i) Many candidates were able to call their function from **part (c)**. Some candidates attempted to pass a parameter to the function when they did not have a function in the function they created. Many candidates attempted to output the data, some candidates output the whole array on one line or did not loop through all of the data items, for example missing off the first or last elements.

(ii) Many candidate solutions produced the correct output. Common errors included the output of all data on the same line, the data out of order, or a data item missing. Some responses had the correct output even with incorrect coded solutions.

Question 2

This question required candidate to set up a record structure and then store records in a circular queue.

(a) There were a range of methods used to create a record. The stronger candidates created a record or a class. Where a class is used the data structured required initialisation within a constructor to show that they existed within the class. Many responses attempted to create a 1D array or list which did not meet the criteria for a record structure. Some responses used a dictionary or a 2D list which were accurately defined.

(b) Many candidates were able to accurately declare and initialise the three variables. Where candidates are provided with values for initialisation, they need to make sure they use these values. Data structures differ each year to test if candidates can manipulate them when they are working in a slightly different way. Some candidates initialised one or more of the values to –1.

Where candidates had created a suitable record structure in **part (a)** they were often able to create the array of the same type and initialise 5 values. Some candidates declared a class in **part (a)** but then did not use the class in **part (b)**, for example creating 1D lists instead. If candidates change their solution it is beneficial to show this in past answers so that their solution can be followed.

(c) This question required candidates to create an Enqueue function for the circular queue. Weaker responses created a linear queue, they incremented the tail pointer but did not reset this to 0 to create the circular queue.

Many candidates were able to accurately create the function header and check the number of items to determine if the queue was full. Some candidates attempted to check each value in the array for a free space instead of using the number of items and stored the record in the first available space.

(d) This question required a Dequeue function that also made use of the circular queue. The question required an empty record to be returned if the queue was full. Many candidates accurately checked if the queue was full but did not return an empty record, a range of values were returned instead for example –1 and null.

Candidates were often able to return the correct record from the queue. Some candidates returned the value before they adjusted the head pointer and the number of items. This would mean that the code after the return statement would never run.

(e) This question required a procedure that read data from the user, created a record and called the Enqueue function.

Some responses took the new record as a parameter and therefore did not take the required input. A common error was calling Enqueue and not storing the return value or calling it twice; once without storing the return value and the second time to store the value. In this case the same element will have been enqueued twice.

(f) (i) Many candidates were able to call EnterRecord six times. When calling Dequeue some candidates did not store the return value or called Dequeue twice. Those candidates who did store the return

value and checked if it was valid did not output the required ID and Quantity, instead outputting the record as an object or only the SaleID. Similarly, in the final output some responses output the whole object which did not display the data, the stronger answers looped through each record and output both values independently.

(ii) Some candidates were able to get the required output. When producing a screenshot candidates must make sure all outputs are visible. This can be from using one screenshot or multiple provided it is all visible.

Question 3

This question required candidates to use object-oriented programming to create classes and objects as well as accessing data from external text files.

(a) (i) Candidates were required to create a class for Employee and to declare its constructor. Many candidates were able to define the class and the constructor accurately. Some of the weaker responses did not define the attributes, attributes, or initialise them appropriately to identify the correct data type. A common error, particularly in Python solutions, was to only initialise 51 elements in the array instead of the required 52 elements.

(ii) There was one get method for candidates to design. Many candidates were familiar with get methods and were able to define this accurately. Some candidates sent a parameter to the function and then attempted to return this parameter instead of the attribute.

(iii) Many candidates were able to accurate define the procedure SetPay and send the required parameters. Some candidates multiplied the weak number by the number of hours instead of accessing the hourly pay for the object.

(iv) Some candidates were able to accurate total the values in the array, some added these manually whilst others used a built-in function to total the values. This function required the total to be returned; some responses returned the total within the loop and therefore the total would never be calculated.

(b) (i) This question required candidates to understand how to use inheritance in object-oriented programming. Some candidates found this challenging and defined the class Manager without inheritance from Employee. The constructor required an understanding that all attributes for Employee were needed as well as the additional one for Manager. Some candidates only include the bonus payment parameter.

(b) (ii) Candidates created a SetPay method for the class Employee and needed to use override this method in this question. Many candidates defined the method but did not call the required parent method to store the value. The stronger responses were able to accurately calculate the amount of pay and send this to the parent method. Weaker responses attempted to assign the data in this function and did not accurately calculate the pay, a common error being multiplying the hours by the bonus value.

(c) The external text file stored data about employees. Candidates were required to read the data from the file, identify if each employee should be an Employee or Manager and then create an object with the appropriate data. There were a range of ways Employees were differentiated from Managers; some candidates manually read each set of data one at a time and hard-coded it to be an employee or a manager, some candidates checked if there was a bonus value – the stronger candidates used exception handling to identify if the bonus value read in was numeric or a string.

Some candidates opened the file but did not close the file or did this in an inappropriate place for example within the loop where each employee record was being read.

Exception handling was used by some candidates, when used for reading from a file both opening and closing the file should be within the try clause.

(d) The hours the employees had worked were stored in an external text file. Candidates needed to read in this data, find the matching employee and then use the SetPay method to set the pay for that employee (or manager).

Many candidates were able to read in the data from the file, fewer candidates closed the file appropriately. Candidates often looped the appropriate number of times (either 8 or until the end of file) and read in the required lines.

The stronger candidates were able to then search through the array of employees to find the correct index. Some candidates compared one employee from the array but did not loop through all of the values. Those that did perform the correct comparison were often able to call the correct method.

(e) (i) Many candidates were able to call the procedure EnterHours, but fewer were able to loop through each object in the array. Some candidates attempted to output all data in the array or did not output the required employee number and the total pay.

(ii) Candidates who had accurately completed the task had appropriate outputs. Some candidates had the output of the total pay but had not included the employee ID so they could not be matched.

COMPUTER SCIENCE

Paper 9618/43
Practical

Key messages

Centres need to make sure that the only document submitted is the evidence document, in a Word document or pdf form. Zip folders should not be uploaded, neither should images showing the candidates, or the program code produced. Additional documentation will not be marked and can cause a delay in the marking of the scripts.

Candidates need to copy their program code into the evidence document. This needs to be copied as text and not as a screenshot. Screenshots are not always clear and when there is coloured text on a black background the text may be unreadable. If the screenshot is not legible then the answer will not be awarded the marks.

Screenshots must be used for the outputs. Candidates need to make sure these are black text on a white background. The screenshots need to be legible and sufficiently large to clearly read the text. Candidates must make sure that all of their answer is visible, if any of it is off the page and cannot be viewed it will not be able to be marked.

Candidates need to adapt their solutions for their chosen language. If a record is being used and the language does not support a record, then an appropriate alternative needs to be selected.

General comments

Candidates must make sure they are answering the question given, for example producing the data structure that has been asked for, instead of creating one they have used before. For example, if a queue is being used the question will state whether the tail pointer points to the last item of the next empty space.

Comments on specific questions

Question 1

This question required candidates to read data from a text file into a 1D array and use a linear search on the contents.

(a) (i) Many candidates were able to accurately define an array. Some candidates only provided a comment to state their intention but did not create an actual array (or list where appropriate).

(ii) Candidates were often able to open the text file and read in the data. There was a mix of approaches to checking when to stop, including looping the number of times there were elements, looping until the end of file and reading all data into a list that was then iterated through. Some candidates also closed the file, but this was fewer than those who opened the file. Some responses included appropriate exception handling.

(b) (i) This question required candidates to create a procedure to output the array contents on single line. Many candidates were able to output the array, but some candidates did this in the main program instead of a procedure.

(ii) Many candidates were able to call the function they defined in **part (b)(i)**.

(iii) Many candidates produced a screenshot showing the correct result. Some candidates produced one screenshot, but it was not all visible in the evidence document. Outputs can be given using two screenshots if they would not be visible with one.

(c) This question required candidates to write a function that performed a linear search on an array to find the number of times a specific value was stored in the array. The stronger candidates were able to create a function that took appropriate parameters and then looped through each element in turn, keeping a count of the number of times the parameter occurred. Some weaker responses took the value to find as an input instead of a parameter, some responses also returned the index where the item was found instead of counting the number of times it occurred.

(d) (i) Many candidates were able to read the data value as input. The weaker responses checked if the value was a number, whilst the stronger responses either cast the value as an integer or checked whether it was a whole number, as well as checking the correct values.

Candidates were often able to call their method from **part 1(c)** but fewer output the return value in an appropriate message. Some candidates attempted to include the message within their function in **part (c)** and some candidates did not store the return value from the search.

(ii) Many candidates were able to produce the correct output. Some candidates did not input the number 12 as required by the question and hence did not produce the required output.

Question 2

This question required candidates to use object-oriented programming to create classes and objects.

(a) (i) Many candidates were able to accurately declare the class and constructor. The class design identifies the values to initialise attributes to in the constructor. The stronger candidates followed this and initialised the current speed to 0 and the horizontal position to 0.

(ii) Candidates were often able to create the required get methods. A common error was sending a parameter to the get method or outputting the value instead of returning it.

(iii) Many candidates could demonstrate an understanding of set methods, they were able to create the required methods, taking a parameter and stored the value correctly. Some of the weaker responses attempted to read in a value from the user within the function or returned the parameter instead of the attribute.

(iv) Candidates were often able to define the method accurately, taking the correct parameters. Due to this being a method, Python programmers need to include self with the parameters.

Many candidates correctly added the speed to the horizontal position, but fewer accurately limited the speed. A common error was not increasing the speed at all if it would go above the maximum, instead of limiting it to the maximum speed.

(b) (i) This question required an understanding of inheritance. Candidates were required to declare a class that inherited from the one they made in **part (a)**. The stronger responses used inheritance accurately and called the parent class constructor within the constructor. Some candidates created a new class and redefined the attributes instead of using the parent class.

(ii) In this question candidates needed to override the method from the parent class. Some candidates were able to do this accurately by overriding the parent method. The stronger responses also called the parent method to avoid the repetition of code. Many candidates were able to add the vertical change to the position, but fewer accurately stopped it from exceeding its maximum speed. Some responses did not attempt to update the horizontal position or speed if the vertical change was too high, instead of limiting it and then updating the speed and position.

(c) There were several different ways of approaching this Questions. 1: A method in each class could be used that outputs the data for the helicopter and then calls the parent method for the horizontal position and speed. 2: One method that tests what type of object is being output and then only outputs the vertical position if the object is of type Helicopter. 3: Using exception handling to attempt to output the vertical position but always outputting the horizontal position and speed.

Many candidates were able to identify one of these methods. Few candidates were able to do this all entirely accurately, with a common error being just outputting all 3 values no matter what the type of object.

(d) (i) Candidates were commonly able to create the required objects with the correct values. Some candidates did not call the required methods the number of times required for each object or called a procedure instead of a method for each object.

(ii) Some candidates were able to produce the correct output.

Question 3

This question required candidates to implement two stacks using an array. They had to push and pop data into and from the stack.

(a) Many candidates were able to declare the arrays and initialise the pointers to 0. Where candidates are provided with the initial values, they need to use these in the question, for example initialising the top pointers to 0. Some candidates initialised them to –1.

When declaring an array in Python candidates need to create the list, a comment is suitable to identify the number of elements and data type, but an empty (or populated) list is still required.

(b) (i) Candidates were provided with the pseudocode algorithm for the push function. Candidates needed to convert this into their chosen programming language. Many candidates were able to do this accurately.

(ii) As with **part (b)(i)** candidate were provided with the pop function. Candidates were required to convert this into their chosen programming language. Many candidates were able to do this accurately.

(iii) Candidates were provided with a text file to read data in for the Animal stack. The question required exception handling and some candidates were able to use this appropriately. When using exception handling with text files candidates must make sure the opening and closing of the file is within the try clause. Otherwise if the try fails and the program then attempts to close a file that is not open it could produce an error. Some candidates opened the file but did not close the file.

(iv) The stack for the colours works in the same way as for animals. Candidates needed to replicate the code and adapt it for the colour array. This meant that the pointer identifiers needed changing and the value of the top pointer needed amending to check whether the stack was full as it has a different number of elements.

(v) The data for the colour stack was provided in a text file and candidates need to repeat their code from reading the animal data file. Candidates needed to make sure they made adjustments, so it was appropriate for this stack. The adjustments included opening the ColourData.txt file, looping 10 times (if not looping until end of file) and using the correct push function. Some candidates did not adjust the number of records being read in or called PushAnimal instead of PushColour.

(c) This question required application of the stacks using the pop functions declared. If there was both an animal and colour, then they were both output. If there was only a colour (no animal) then the text ‘No animal’ was output and the colour was pushed back onto the stack. The reverse happened if there was only an animal.

There were a range of attempts at this question. Some candidates output the return value without checking them, so if it ran out of colours it would have produced an error. They often did then go on to make the appropriate checks.

Some candidates output the correct message but pushed the animal onto the incorrect stack or did not push it.

The stronger candidates had a clear structure in their response where they checked if each element was present in turn and then finally outputting if both were present.

Candidates did not have to consider what would happen if there were neither present, because it would have been caught by the no colour or no animal. The stronger responses did, however, consider this and output an appropriate message.

(d) (i) This question was often answered well by candidates who attempted it. They called the ReadData once and then called OutputItem four times (sometimes manually calling it 4 times and sometimes in a loop).

(ii) Many candidates who gave an answer to this question produced the correct output.