

OCR GCSE **Computer Science** J277

# Clear**Revise**<sup>TM</sup>

## Illustrated revision and practice:

- Over 500 marks of examination style questions
- Answers provided for all questions within the book
- Illustrated topics to improve memory and recall
- Specification references for each topic
- Examination tips and techniques

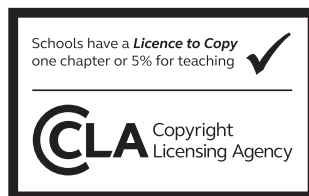
## Experience + science + beautiful design = better results

Absolute clarity is the aim with a new generation of revision guide for the 2020s. This guide has been expertly compiled and edited by successful former teachers of Computer Science, highly experienced examiners and a good measure of scientific research into what makes revision most effective.

PG Online have a record of significantly raising and sustaining examination results at GCSE in schools using their award-winning teaching resources. This book aims to improve things a step further.

Past examinations questions are essential to good preparation, improving understanding and confidence. This guide has combined revision with tips and more practice questions than you could shake a stick at. All the essential ingredients for getting a grade you can be really proud of.

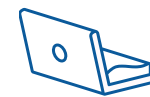
Each specification topic has been referenced and distilled into the key points to make in an examination for top marks. Knowledge, application and analysis are all specifically and carefully assessed in exam questions and throughout this book.



PG ONLINE

Clear**Revise**

OCR GCSE Computer Science J277

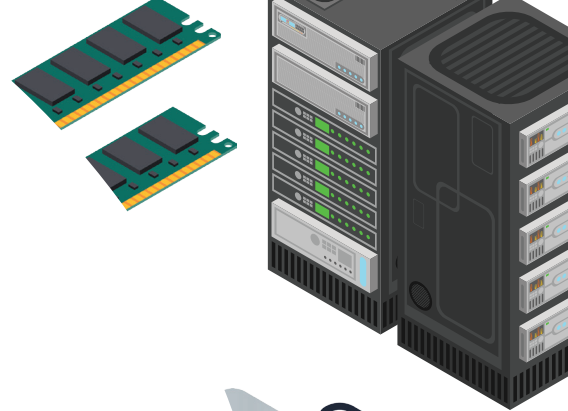
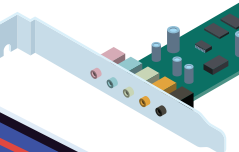
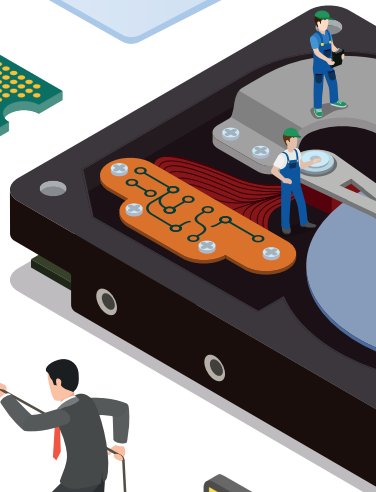
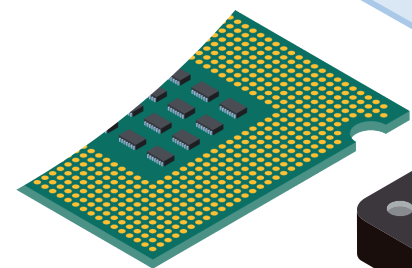
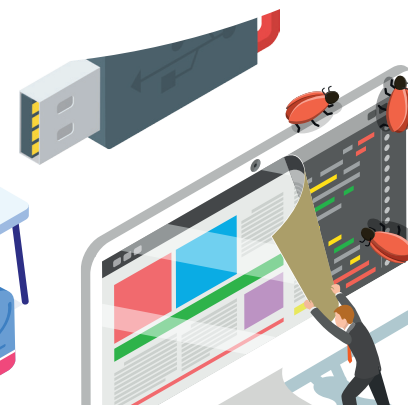


PG ONLINE

# Clear**Revise**

Illustrated revision and practice

## OCR GCSE **Computer Science** J277



# Clear**Revise**<sup>TM</sup>

## OCR GCSE

## **Computer Science** J277

Illustrated revision and practice

Published by  
PG Online Limited  
The Old Coach House  
35 Main Road  
Tolpuddle  
Dorset  
DT2 7EW  
United Kingdom

[sales@pgonline.co.uk](mailto:sales@pgonline.co.uk)  
[www.pgonline.co.uk](http://www.pgonline.co.uk)  
**2020**



**PG ONLINE**

# PREFACE

Absolute clarity! That’s the aim.

This is everything you need to ace your exam and beam with pride. Each topic is laid out in a beautifully illustrated format that is clear, approachable and as concise and simple as possible.

Each section of the specification is clearly indicated to help you cross-reference your revision. The checklist on the contents pages will help you keep track of what you have already worked through and what’s left before the big day.

We have included worked examination-style questions with answers for almost every topic. This helps you understand where marks are coming from and to see the theory at work for yourself in an examination situation. There is also a set of exam-style questions at the end of each section for you to practise writing answers for. You can check your answers against those given at the end of the book.

# LEVELS OF LEARNING

Based on the degree to which you are able to truly understand a new topic, we recommend that you work in stages. Start by reading a short explanation of something, then try and recall what you’ve just read. This has limited effect if you stop there but it aids the next stage. Question everything. Write down your own summary and then complete and mark a related exam-style question. Cover up the answers if necessary, but learn from them once you’ve seen them. Lastly, teach someone else. Explain the topic in a way that they can understand. Have a go at the different practice questions – they offer an insight into how and where marks are awarded.

# ACKNOWLEDGMENTS

**The questions in the ClearRevise textbook are the sole responsibility of the authors and have neither been provided nor approved by the examination board.**

Every effort has been made to trace and acknowledge ownership of copyright. The publishers will be happy to make any future amendments with copyright owners that it has not been possible to contact. The publisher would like to thank the following companies and individuals who granted permission for the use of their images in this textbook.

**All Sections**  
Photographic images: © Shutterstock  
Icons by Icons8.com

Design and artwork: Jessica Webb / PG Online Ltd  
First edition 2020  
A catalogue entry for this book is available from the British Library  
ISBN: 978-1-910523-23-0  
Copyright © PG Online 2020  
All rights reserved

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without the prior written permission of the copyright owner.  
Printed on FSC certified paper by Bell and Bain Ltd, Glasgow, UK.



# THE SCIENCE OF REVISION

## Illustrations and words

Research has shown that revising with words and pictures doubles the quality of responses by students.<sup>1</sup> This is known as ‘dual-coding’ because it provides two ways of fetching the information from our brain. The improvement in responses is particularly apparent in students when asked to apply their knowledge to different problems. Recall, application and judgement are all specifically and carefully assessed in public examination questions.

## Retrieval of information

Retrieval practice encourages students to come up with answers to questions.<sup>2</sup> The closer the question is to one you might see in a real examination, the better. Also, the closer the environment in which a student revises is to the ‘examination environment’, the better. Students who had a test 2–7 days away did 30% better using retrieval practice than students who simply read, or repeatedly reread material. Students who were expected to teach the content to someone else after their revision period did better still.<sup>3</sup> What was found to be most interesting in other studies is that students using retrieval methods and testing for revision were also more resilient to the introduction of stress.<sup>4</sup>

## Ebbinghaus’ forgetting curve and spaced learning

Ebbinghaus’ 140-year-old study examined the rate in which we forget things over time. The findings still hold power. However, the act of forgetting things and relearning them is what cements things into the brain.<sup>5</sup> Spacing out revision is more effective than cramming – we know that, but students should also know that the space between revisiting material should vary depending on how far away the examination is. A cyclical approach is required. An examination 12 months away necessitates revisiting covered material about once a month. A test in 30 days should have topics revisited every 3 days – intervals of roughly a tenth of the time available.<sup>6</sup>

## Summary

Students: the more tests and past questions you do, in an environment as close to examination conditions as possible, the better you are likely to perform on the day. If you prefer to listen to music while you revise, tunes without lyrics will be far less detrimental to your memory and retention. Silence is most effective.<sup>5</sup> If you choose to study with friends, choose carefully – effort is contagious.<sup>7</sup>

1. Mayer, R. E., & Anderson, R. B. (1991). Animations need narrations: An experimental test of dual-coding hypothesis. *Journal of Education Psychology*, (83)4, 484-490.

2. Roediger III, H. L., & Karpicke, J.D. (2006). Test-enhanced learning: Taking memory tests improves long-term retention. *Psychological Science*, 17(3), 249-255.

3. Nestojko, J., Bui, D., Kornell, N. & Bjork, E. (2014). Expecting to teach enhances learning and organisation of knowledge in free recall of text passages. *Memory and Cognition*, 42(7), 1038-1048.

4. Smith, A. M., Floerke, V. A., & Thomas, A. K. (2016) Retrieval practice protects memory against acute stress. *Science*, 354(6315), 1046-1048.

5. Perham, N., & Currie, H. (2014). Does listening to preferred music improve comprehension performance? *Applied Cognitive Psychology*, 28(2), 279-284.

6. Cepeda, N. J., Vul, E., Rohrer, D., Wixted, J. T. & Pashler, H. (2008). Spacing effects in learning a temporal ridgeline of optimal retention. *Psychological Science*, 19(11), 1095-1102.

7. Busch, B. & Watson, E. (2019), *The Science of Learning*, 1st ed. Routledge.

# CONTENTS AND CHECKLIST

## Section 1

<b>Spec</b>	<b>Computer systems (Paper 1 - J277/01)</b>	<b>1</b>	<input checked="" type="checkbox"/>
1.1.1	Architecture of the CPU	2	<input type="checkbox"/>
1.1.1	Common CPU components and their function	3	<input type="checkbox"/>
1.1.2	CPU performance	4	<input type="checkbox"/>
1.1.3	Embedded systems	4	<input type="checkbox"/>
1.2.1	Primary storage (memory)	6	<input type="checkbox"/>
1.2.1	Virtual memory	7	<input type="checkbox"/>
1.2.2	Secondary storage	8	<input type="checkbox"/>
	<b>Section 1 Examination practice</b>	<b>10</b>	<input type="checkbox"/>

## Section 2

1.2.3	Units of data storage	11	<input type="checkbox"/>
1.2.4	Binary ⇌ denary conversion	12	<input type="checkbox"/>
1.2.4	Adding binary integers	13	<input type="checkbox"/>
1.2.4	Hexadecimal ⇌ binary conversion	14	<input type="checkbox"/>
1.2.4	Hexadecimal ⇌ denary conversion	15	<input type="checkbox"/>
1.2.4	Binary shifts	16	<input type="checkbox"/>
1.2.4	Characters	17	<input type="checkbox"/>
1.2.4	Images	18	<input type="checkbox"/>
1.2.4	Sound	20	<input type="checkbox"/>
1.2.5	Compression	21	<input type="checkbox"/>
	<b>Section 2 Examination practice</b>	<b>22</b>	<input type="checkbox"/>

## Section 3

1.3.1	Networks	23	<input type="checkbox"/>
1.3.1	Network hardware	24	<input type="checkbox"/>
1.3.1	Topologies	25	<input type="checkbox"/>
1.3.1	Client-server networks	26	<input type="checkbox"/>
1.3.1	Peer-to-peer (P2P) networks	27	<input type="checkbox"/>
1.3.1	The Internet	28	<input type="checkbox"/>
1.3.2	Connecting wired and wireless networks	29	<input type="checkbox"/>
1.3.2	Encryption	30	<input type="checkbox"/>
1.3.2	IP and MAC addressing	31	<input type="checkbox"/>
1.3.2	TCP/IP layers	31	<input type="checkbox"/>
1.3.2	Standards and protocols	32	<input type="checkbox"/>
	<b>Section 3 Examination practice</b>	<b>33</b>	<input type="checkbox"/>

## Section 4

1.4.1	Threats to computer systems and networks	34	<input type="checkbox"/>
1.4.2	Identifying and preventing vulnerabilities	35	<input type="checkbox"/>
	<b>Section 4 Examination practice</b>	<b>36</b>	<input type="checkbox"/>

## Section 5

1.5.1	Operating systems	37	<input type="checkbox"/>
1.5.2	Utility software	38	<input type="checkbox"/>
	<b>Section 5 Examination practice</b>	<b>39</b>	<input type="checkbox"/>

## Section 6

1.6.1	Ethical, legal, cultural and environmental impact	40	<input type="checkbox"/>
1.6.1	Legislation	43	<input type="checkbox"/>
1.6.1	Software licensing	44	<input type="checkbox"/>
	<b>Section 6 Examination practice</b>	<b>45</b>	<input type="checkbox"/>

## Section 7

<b>Spec</b>	<b>Computational thinking, algorithms and programming (Paper 2 - J277/02)</b>	<b>46</b>	<input checked="" type="checkbox"/>
2.1.1	Computational thinking	47	<input type="checkbox"/>
2.1.2	Identifying the inputs, processes and outputs for a problem	48	<input type="checkbox"/>
2.1.2	Structure diagrams	49	<input type="checkbox"/>
2.1.2	Using flowcharts	50	<input type="checkbox"/>
2.1.2	Using pseudocode	51	<input type="checkbox"/>
2.1.2	Trace tables	52	<input type="checkbox"/>
2.1.3	Searching algorithms	54	<input type="checkbox"/>
2.1.3	Bubble sort	55	<input type="checkbox"/>
2.1.3	Merge sort	56	<input type="checkbox"/>
2.1.3	Insertion sort	57	<input type="checkbox"/>
2.1.3	Identifying algorithms	58	<input type="checkbox"/>
	<b>Section 7 Examination practice</b>	<b>59</b>	<input type="checkbox"/>

## Section 8

2.2.1	Variables, constants, assignments	61	<input type="checkbox"/>
2.2.1	Inputs, outputs and operators	62	<input type="checkbox"/>
2.2.1	Sequence and selection	63	<input type="checkbox"/>
2.2.1	Iteration	64	<input type="checkbox"/>
2.2.2	Data types and casting	66	<input type="checkbox"/>
2.2.3	String manipulation	67	<input type="checkbox"/>

2.2.3	Arrays .....	68	<input type="checkbox"/>
2.2.3	Two-dimensional arrays .....	69	<input type="checkbox"/>
2.2.3	Structured records .....	70	<input type="checkbox"/>
2.2.3	Using SQL to search for data .....	71	<input type="checkbox"/>
2.2.3	File handling operations .....	72	<input type="checkbox"/>
2.2.3	Subprograms .....	73	<input type="checkbox"/>
2.2.3	The use of procedures .....	74	<input type="checkbox"/>
	<b>Section 8 Examination practice .....</b>	<b>75</b>	<input type="checkbox"/>

<b>Section 9</b>			<input checked="" type="checkbox"/>
2.3.1	Defensive design .....	78	<input type="checkbox"/>
2.3.2	Testing .....	80	<input type="checkbox"/>
	<b>Section 9 Examination practice .....</b>	<b>81</b>	<input type="checkbox"/>

<b>Section 10</b>			<input checked="" type="checkbox"/>
2.4.1	Boolean logic .....	82	<input type="checkbox"/>
2.5.1	Languages .....	84	<input type="checkbox"/>
2.5.2	The Integrated Development Environment (IDE) .....	85	<input type="checkbox"/>
	<b>Section 10 Examination practice .....</b>	<b>86</b>	<input type="checkbox"/>

Examination practice answers .....	87
Band descriptions and levels of response for extended response questions .....	94
Index .....	95
<b>Examination tips .....</b>	<b>98</b>

# MARK ALLOCATIONS

**Green mark allocations<sup>[1]</sup>** on answers to in-text questions through this guide help to indicate where marks are gained within the answers. A bracketed '1' e.g. <sup>[1]</sup> = one valid point worthy of a mark. There are often many more points to make than there are marks available so you have more opportunity to max out your answers than you may think.

# TOPICS FOR PAPER 1

## Computer systems (J277/01)

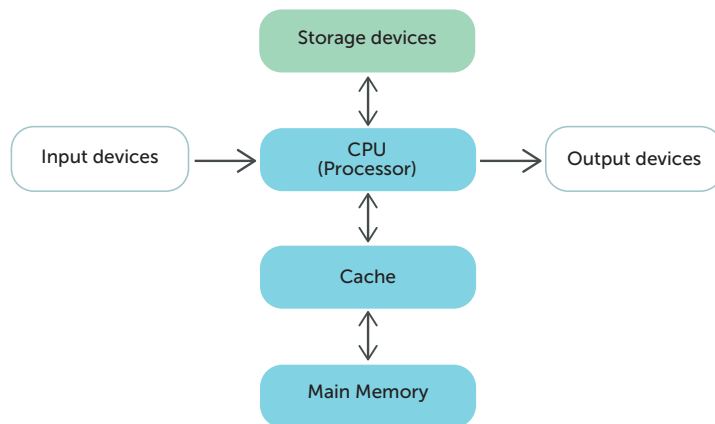
### Information about Paper 1

**Written paper: 1 hour and 30 minutes**  
**50% of total GCSE**  
**80 marks**

It is a non-calculator paper.  
All questions are mandatory.  
It consists of multiple-choice questions, short response questions and extended response questions.

# ARCHITECTURE OF THE CPU

All computers have a CPU, memory, one or more input devices and output devices.



Identify **two** input devices and **one** output device that may be connected to the CPU. [3]

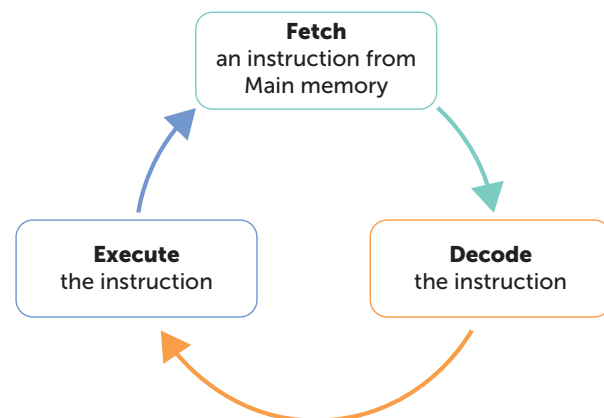
Input devices include a keyboard<sup>[1]</sup>, mouse<sup>[1]</sup>, scanner<sup>[1]</sup>, digital camera<sup>[1]</sup>, microphone<sup>[1]</sup> and web cam<sup>[1]</sup>. Output devices include a monitor<sup>[1]</sup>, printer<sup>[1]</sup> and speaker<sup>[1]</sup>.

## The purpose of the CPU

The purpose of the **Central Processing Unit (CPU)** is to execute instructions stored in memory by repeatedly carrying out the **fetch-execute cycle**. The CPU contains the **Arithmetic Logic Unit (ALU)**, the **Control Unit** and several general-purpose and special-purpose registers.

## The fetch-execute cycle

Every CPU instruction is **fetch**ed from memory. Once fetched, it is **dec**oded by the Control Unit to find out what to do with it. Then the instruction is executed. Every operation carried out within the fetch-execute cycle is regulated by a 'tick' or cycle of the CPU clock.



A single core 4.5GHz processor has 4,500,000,000 clock cycles or 'ticks' a second. This is known as the clock speed.

# COMMON CPU COMPONENTS AND THEIR FUNCTION

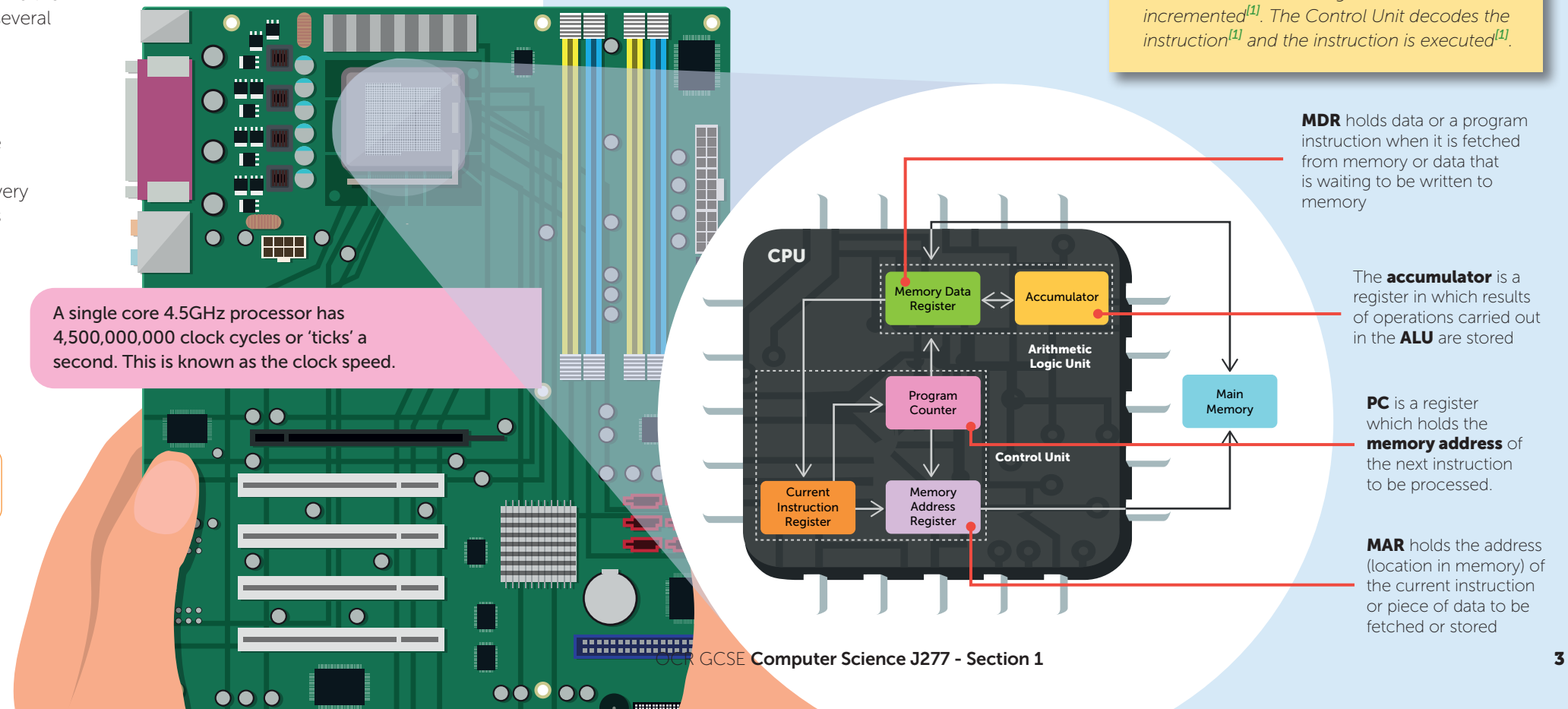
CPU Component	Typical size	Function
<b>ALU</b> (Arithmetic Logic Unit)		Carries out mathematical and logical operations including AND, OR and NOT, and binary shifts
<b>CU</b> (Control Unit)		Coordinates all of the CPU's actions in the fetch-decode-execute cycle
<b>Cache</b>	Up to 32 MB	Sends and receives control signals to and from other devices within the computer
<b>Registers</b>	32 bits or 64 bits	Even smaller and faster than cache memory, registers are memory locations within the CPU to temporarily store memory addresses, instructions or data

## Von Neumann architecture

**John von Neumann** developed the **stored program computer**. In a von Neumann computer, both programs and the data they use are stored in memory.

Identify **two** events that happen during the fetch-decode-execute cycle. [2]

The address of the next instruction to be executed is held in the PC.<sup>[1]</sup> The CPU fetches the instruction and data from memory<sup>[1]</sup> and stores them in its registers<sup>[1]</sup>. The PC is incremented<sup>[1]</sup>. The Control Unit decodes the instruction<sup>[1]</sup> and the instruction is executed<sup>[1]</sup>.

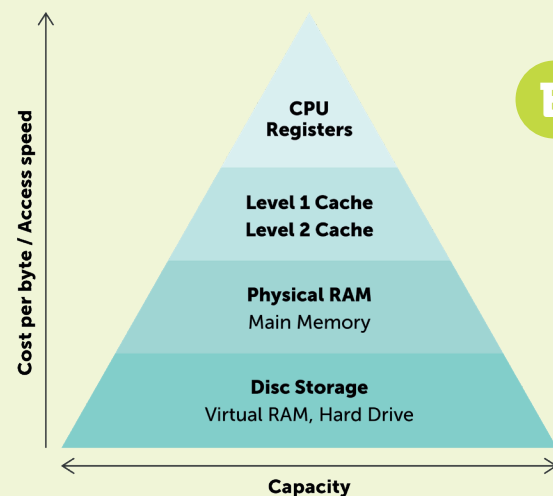
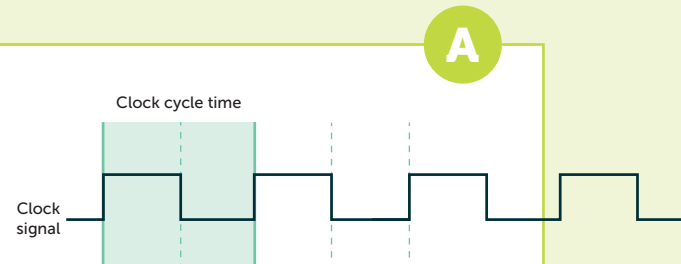




# CPU PERFORMANCE

## Clock speed

The **clock speed** determines the number of **fetch-execute cycles** per second. Every action taking place in the CPU takes place on a tick of the clock, or clock cycle. Each cycle is one **hertz** so a 3.7 GHz processor will cycle at 3.7 billion times per second.



B

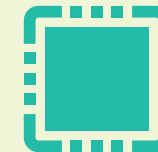
## Cache size

Since **cache memory** operates much faster than main memory, data is transferred in and out of cache memory more quickly, which makes the CPU more efficient as less time is spent waiting for data to be transferred. There are two or three levels of cache. The fastest cache with the smallest capacity is Level 1 cache. The CPU will optimise its use of the fastest cache before using the next level, or using **Random Access Memory (RAM)**, in order to improve performance speed.

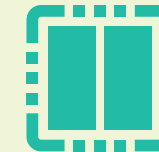
C

## Number of cores

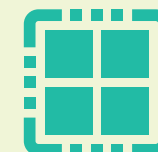
A processor may contain more than one **core**. Each core can process one operation per clock cycle. A dual- or quad-core processor will be able to perform 2 or 4 operations simultaneously (for example, run two programs simultaneously), but only if the software it is running is designed for multi-core processors.



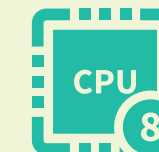
CPU



Dual Core



Quad Core



Octa Core

Amy's computer has a 4.5 GHz, dual core processor.

(a) How many cycles is a 4.5 GHz, dual core processor theoretically able to perform each second? [1]

(b) Explain why a computer with a dual core processor may not be twice as fast as a single core processor with the same clock speed. [2]

(a) 9 billion.<sup>[1]</sup>

(b) The software running on the computer may not be written to make the most efficient use of multiple cores.<sup>[1]</sup> Additional cores may be redundant if the software is only written for a single core<sup>[1]</sup> or if the output of one operation is required to perform the second operation<sup>[1]</sup> so they cannot be processed simultaneously.<sup>[1]</sup>

# EMBEDDED SYSTEMS

An **embedded system** is used to control the function of electronic devices such as those commonly found in the home. They often don't need a full operating system since they perform limited and very specific tasks with their input frequently controlled by a button press or switch.

Embedded systems must be reliable since they cannot be modified once manufactured. The program that controls them is held in **Read Only Memory (ROM)**.

Examples include air conditioning or heating systems, radio alarm clocks, washing machines, fridges, microwave ovens and digital cameras.

Jonny says that his car's satnav is an embedded system. State whether he is correct and explain your answer. [3]

Yes, he is correct.<sup>[1]</sup> It has one dedicated function<sup>[1]</sup> with simple controls. The user cannot change the software held in ROM within the embedded system.<sup>[1]</sup> The user cannot run other general software on it.<sup>[1]</sup>



Remember to give full answers to the questions – don't just list key words.



K9Track is a new device designed to be attached to a dog collar. It allows the owner to track the pet's location on a connected smartphone and to monitor the animal's health and levels of exercise each day.

Discuss the ethical, legal and privacy issues that should be considered when creating tracking and monitoring technology such as K9Track.

[6]

Since a dog is commonly with their owner, such a tracking device will also be effectively tracking and recording the movements of the (various) owners.<sup>[1]</sup> The use of this data will need to be securely stored by the app or cloud service provider.<sup>[1]</sup>

The costs of pet insurance or veterinary bills may reduce if its health and movement is monitored and regularly checked.<sup>[1]</sup> This may benefit those who can afford the high cost of new devices<sup>[1]</sup>, widening the digital divide between those who can and cannot afford technology<sup>[1]</sup>.

The algorithms used to determine what levels of exercise are healthy and what are not for each individual pet will need to be very accurate<sup>[1]</sup>, taking into prior account an animal's size, weight, breed and general health<sup>[1]</sup>. Inaccuracies could lead to blame if an animal's lack of exercise were to blame for ill health<sup>[1]</sup>.

Monitoring and tracking your pet's movements provides another reason to check your smartphone<sup>[1]</sup>, increasing the amount of (unnecessary) time spent using technology and looking at screens<sup>[1]</sup>. This may negatively impact or reduce the time spent with the pet or family.<sup>[1]</sup>

The security of the data<sup>[1]</sup> gathered, both accessible through the mobile app and in its raw form stored by the manufacturer will need to be carefully protected from unauthorised access<sup>[1]</sup>. A biometric access key<sup>[1]</sup> could be used to access the mobile phone app, and the stored data should be adequately protected<sup>[1]</sup> (under the Data Protection Act) from malware<sup>[1]</sup>, hackers<sup>[1]</sup> (unauthorised access), from social engineering<sup>[1]</sup> and from interception<sup>[1]</sup>.

Marks are indicative only. Refer to the band descriptions and levels of response guidance for extended response questions on page 94.



This essay style question is assessed against the levels of response guidelines on page 94. The quality of written communication, including spelling, punctuation and grammar may also be assessed through your response to similar questions.



## 1.6.1

# LEGISLATION

There are four main areas of **legislation** that need to be understood:

- The Data Protection Act 2018
- Computer Misuse Act 1990
- Copyright, Designs and Patents Act 1988
- Software licences (i.e. open source and proprietary)

## Data Protection Act 2018

The **Data Protection Act** was updated in 2018 to incorporate the General Data Protection Regulations (GDPR). It has six principles that govern how data should be stored and processed.

These state that data must be:

1. Fairly and lawfully processed
2. Used for specific purposes only
3. Adequate, relevant and not excessive
4. Accurate and up-to-date
5. Not be kept longer than necessary
6. Kept secure

In addition, the data must be kept in accordance with the rights of data subjects.

## Computer Misuse Act 1990

The **Computer Misuse Act** was introduced in 1990 to make unauthorised access to programs or data (hacking) and cybercrime illegal. The act recognises three offences:

1. Unauthorised access to computer material.
  2. Unauthorised access with intent to commit or facilitate a crime.
  3. Unauthorised modification of computer material.
- It is also illegal to make, supply or obtain anything which can be used in computer misuse offences, including the production and distribution of malware.

## Copyright, Designs and Patents Act 1988

This act is designed to protect the works of companies and individuals from being illegally used, copied or distributed. 'Works' include books, music, images, video and software.





# TOPICS FOR PAPER 2

## Computational thinking, algorithms and programming (J277/02)

### Information about Paper 2

Written paper: 1 hour and 30 minutes

50% of total GCSE

80 marks

This is a non-calculator paper.

This paper consists of two sections. Students must answer all questions from Section A and Section B.

In Section B, questions assessing students' ability to write or refine algorithms must be answered using either the OCR Exam Reference Language or the high-level programming language they are familiar with.

2.1.1

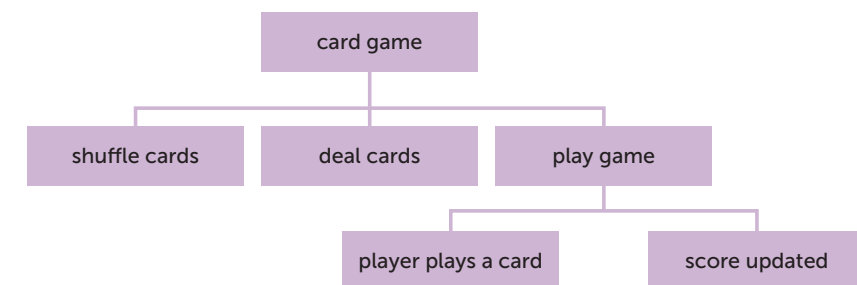
## COMPUTATIONAL THINKING

### Principles of computational thinking

**Computational thinking** is a process used to solve complex problems. It means formulating a problem and expressing its solution in such a way that a computer can carry it out.

There are three basic steps:

- **Abstraction** involves identifying the key parts of the problem and hiding those that aren't important so so that it becomes easier to solve. For example, if a program is to be written to simulate a card game, the first task to be accomplished may be 'Shuffle the cards'. This is an abstraction – implementing it will involve specifying a way to randomise 52 variables representing the cards. We can refer to 'shuffle' throughout the program without specifying how it will be done.
- **Decomposition** means breaking down a complex problem into smaller, manageable parts which are easier to solve. This comprises the following steps:
  - Identify the main problem
  - List the main components, functions or tasks
  - Break these down into smaller components or sub-tasks which can then be completed separately. For example:



- **Algorithmic thinking** is the consideration that goes into how to solve a problem using one or more algorithms. For instance, there may be two ways to shuffle the cards, but one may make them more random whilst another may be faster. An **algorithm** is the series of steps that a program needs to perform to solve the problem.

A self-driving car is being developed. The software has to be capable of distinguishing between an animal and a person crossing the road in front of it.

(a) Define what is meant by **abstraction**. [2]

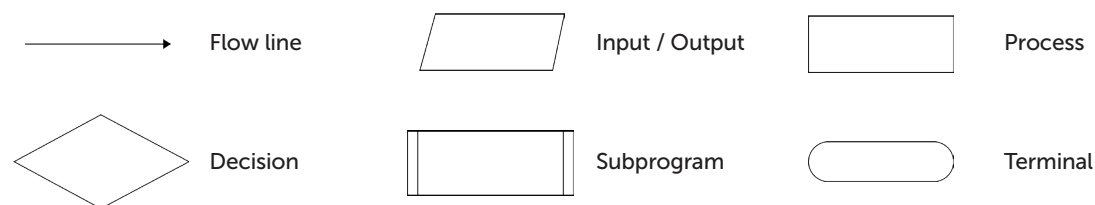
(b) Give **one** example of how abstraction could be used in developing this software. [1]

(a) Filtering out/removing/hiding details of a problem<sup>[1]</sup> that are not relevant to a solution<sup>[1]</sup>.

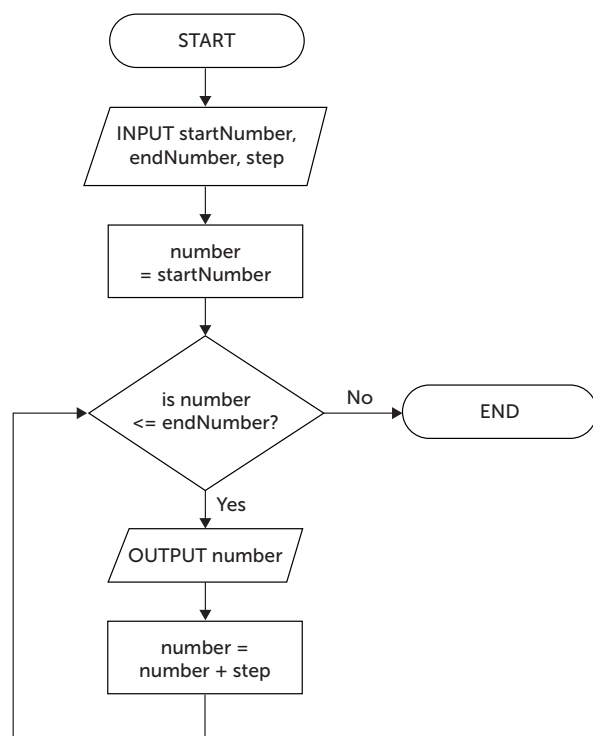
(b) Any example of something that can be removed or hidden, e.g. speed of movement<sup>[1]</sup>, location at which something is crossing<sup>[1]</sup> / whether it is on a pedestrian crossing<sup>[1]</sup> / aerodynamic design of the vehicle<sup>[1]</sup>.

# USING FLOWCHARTS

Flowcharts are a useful tool that can be used to develop solutions to a problem. Standard flowchart symbols are shown below:



Problem: Draw a flowchart for a program which will count in steps from a start number up to an end number. The user enters the start and end numbers and the step to count in. For example, if the user enters 2, 21, 3, the program will output the numbers 2, 5, 8, 11, 14, 17, 20.



- Look at the flowchart above.
- (a) What will be output if the user enters 7, 50, 10 for the three values? [1]
  - (b) What will be output if the user enters an end number which is less than the start number? [1]
- (a) 7, 17, 27, 37, 47<sup>[1]</sup>
- (b) Nothing will be output.<sup>[1]</sup>

# USING PSEUDOCODE

The problem with using a flowchart to develop an algorithm is that it does not usually translate very easily into program code.

**Pseudocode** is useful for developing an algorithm using programming-style constructs, but it is not an actual programming language. This means that a programmer can concentrate on figuring out how to solve the problem without worrying about the details of how to write each statement in the programming language that will be used.

Using pseudocode, the algorithm shown in the flowchart above could be expressed like this:

```

input startNumber, endNumber, step
set number to startNumber
while number <= endNumber
    output(number)
    add step to number
endwhile
    
```

## Using OCR Exam Reference Language

OCR has published a formally defined language called **OCR Exam Reference Language (ERL)**. This is defined in the specification for GCSE Computer Science J277, downloadable from the OCR website.

Some questions in the exam specify that you must use **either** OCR Exam Reference Language **or** a high-level programming language that you have studied to write or complete a program. **Marks are awarded for correctly using syntax to represent programming constructs, whichever language you use.**

The code for the above problem written using OCR Exam Reference Language looks like this:

```

startNumber = input("Enter start number: ")
endNumber = input("Enter end number: ")
step = input("Enter step: ")
number = startNumber
while number <= endNumber
    print(number)
    number = number + step
endwhile
    
```

Note that if there are three values to be input, you **MUST** use three input statements if you are asked to use OCR ERL or a programming language rather than pseudocode. Each input statement is used to input a single value and assign it to a variable.

The syntax used in OCR Exam Reference Language (ERL) is explained in more detail in Section 8.

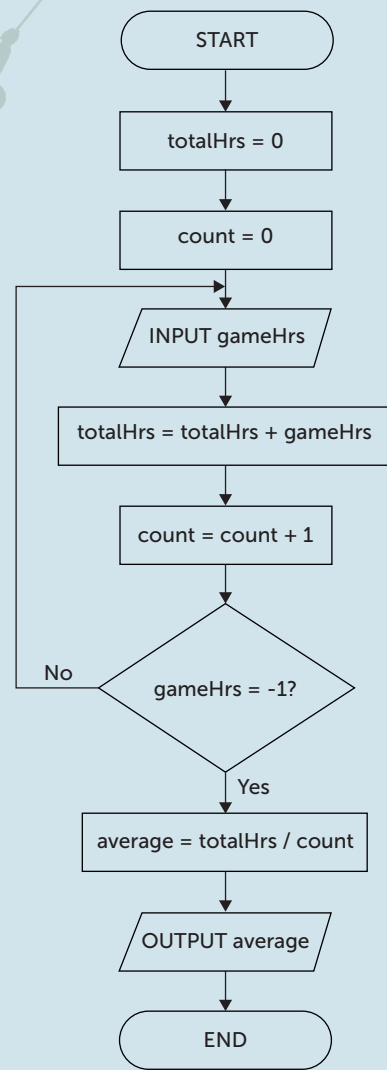
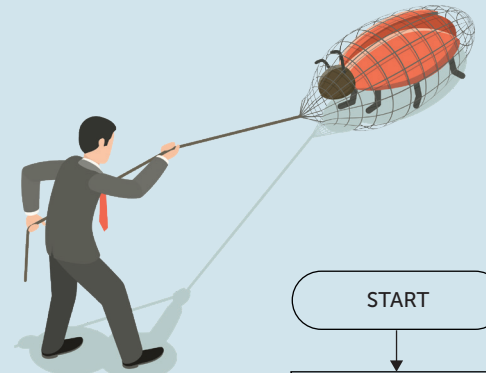


# TRACE TABLES

A **trace table** is used to show how the values of variables change during execution of a program.

As each line of code is executed, the current value of any variable or logical expression that is changed is written in the appropriate column of the table. It is not necessary to fill in a cell if the value has not changed from the line above.

Example: Ben designs a flowchart for an algorithm to calculate the average number of hours students spend per week playing computer games. He uses test data for 3 students spending respectively 8, 10 and 12 hours playing games. This should result in an average of 10 hours.



1. Describe how the algorithm could be corrected. [3]

The input, gameHrs, should be tested right after it has been input.<sup>[1]</sup> However, a program cannot jump out of a loop before completing it.<sup>[1]</sup> Therefore, the input statement and the test for gameHrs = -1 should be placed at the end of the loop.<sup>[1]</sup> An initial input statement is required before entering the loop.<sup>[1]</sup>

gameHrs	totalHrs	count	gameHrs = -1?	average
	0	0		
8	8	1	No	
10	18	2	No	
12	30	3	No	
-1	29	4	Yes	7.25

Oops! The algorithm must be incorrect, since it produces the wrong answer.

## Identifying common errors

There are two types of error that can occur in program code. A **syntax error** occurs when a statement is written which does not obey the rules of the programming language. Common syntax errors include:

- writing a single statement to input two variables:  
`input("Please enter x and y", x, y)` - is incorrect. It should be:  
`x = input("Please input x ")`  
`y = input("Please enter y ")`
- writing output statements incorrectly:  
`print(Total sum = , totalSum)` - should be written:  
`print("Total sum = ", totalSum)` - don't forget the quote marks!
- writing Boolean conditions incorrectly:  
`if x < 1 OR > 100` - should be written:  
`if x < 1 OR x > 100`

A **logic error** occurs when the program does not do what the user intended or gives an answer that isn't what the programmer intended.

If you do any programming you will likely discover many ways of making logic errors.

2. The code below adds the even numbers between 1 and 50.

```
count = 0
while count < 50
    count = count + 2
    sum = sum + count
endwhile
print(Total, sum)
```

Find **two** errors in this code. State in each case whether they are logic errors or syntax errors. [4]

`sum` has not been initialised to 0.<sup>[1]</sup> Logic error.<sup>[1]</sup>  
`count < 50` will never include 50. It should be `count <= 50`<sup>[1]</sup> Logic error.<sup>[1]</sup>  
`print(Total, sum)` doesn't have the string in quote marks. It should be `print("Total", sum)`<sup>[1]</sup>. Syntax error.<sup>[1]</sup>



# IDENTIFYING ALGORITHMS

Bubble sort	Insertion sort
<pre>//Bubble sort aList = [17, 3, 7, 15, 13, 23, 20] //get number of items in the array numItems = aList.length pass = numItems - 1 swapMade = True while pass &gt; 0 AND swapMade     swapMade = False     for j = 0 to passNumber - 1         if aList[j] &gt; aList[j + 1] then             temp = aList[j]             aList[j] = aList[j + 1]             aList[j + 1] = temp             swapMade = True         endif     next j     pass = pass - 1 endwhile print("Sorted list: ", aList)</pre>	<pre>//Insertion sort aList = [17, 3, 7, 15, 13, 23, 20] listLength = aList.length for index = 1 to listLength -1     current = aList[index]     pos = index     while pos &gt; 0 AND aList[pos-1] &gt; current         aList[pos] = aList[pos - 1]         pos = pos - 1     endwhile     aList[pos] = current next index print("Sorted list: ", aList)</pre>

Linear search	Binary search
<pre>//Linear search aList = [14, 2, 3, 11, 1, 9, 5, 8, 10, 6] print("List to be searched: ", aList) found = False index = 0 searchItem = input("Number sought: ") while NOT found AND index &lt; aList.length     if aList[index] == searchItem then         found = True     else         index = index + 1     endif endwhile if found then     print(searchItem, " was found in position ", index, " of the list.") else     print("Item not found") endif</pre>	<pre>//Binary search aList = [2, 3, 11, 12, 15, 19, 23, 30, 36, 45] print("List to be searched: ", aList) found = False first = 0 last = aList.length - 1 searchItem = input("Number sought: ") while NOT found AND first &lt;= last     midpt = int((first + last) / 2)     if aList[midpt] == searchItem then         found = True         index = midpt     else         if aList[midpt] &lt; searchItem then             first = midpt + 1         else             last = midpt - 1         endif     endif endwhile if found     print("Found at position ", index, " in the list.") else     print("Item is not in the list.") endif</pre>



# EXAMINATION PRACTICE

1. An algorithm is given below.

```
01 aList = [3,6,7,9,13,15,16,19,20,24,26,29,36]
02 found = False
03 n = 0
04 x = input("Enter a number: ")
05 while found == False AND n < aList.length
06     print(aList[n])
07     if aList[n] == x then
08         found = True
09     else
10         n = n + 1
11     endif
12 endwhile
13 if found then
14     print(x, " found at position ", n)
15 else
16     print("invalid number")
17 endif
```

(a) At line 05, what is the value of `aList.length`? [1]

(b) The user enters 9 at line 04. What is printed at line 06 the first 3 times the `while...endwhile` loop is performed? [3]

(c) State what will be printed at line 14 if the user enters the number 9. [1]

(d) Explain the purpose of this algorithm. [2]
2. An array `names` holds `n` items. An algorithm for a bubble sort is given below.

```
01 swapMade = True
02 while swapMade = True
03     swapMade = False
04     for index = 0 to n - 2
05         if names[index] > names[index+1] then
06             swap the names
07             swapMade = True
08         endif
09     next index
10 endwhile
```

(a) Explain the purpose of the variable `swapMade` in the algorithm. [2]

(b) Write the code for "*swap the names*" in line 06. [3]

(c) The list `names` contains the following:  
**Edna Adam Victor Charlie Jack Ken Maria**  
Write the contents of the list after each of the first two times the `while...endwhile` loop is executed. [2]

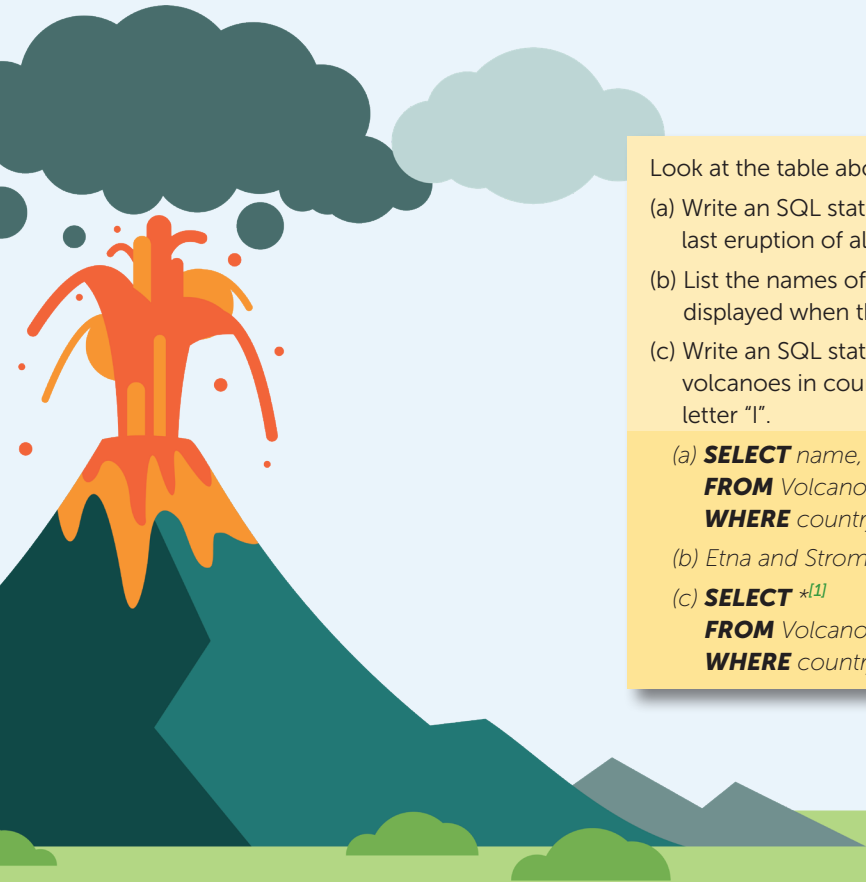
(d) How many times will the `while` loop be executed before the program terminates? Explain your answer. [2]

# STRUCTURED RECORDS

A **database** is a collection of records each having an identical record structure. Each field in a record has a defined field type such as integer, real, currency, Boolean or string.

VolcanoTable

name	country	lastErupted	explosivityIndex	elevationMetres
Taal	Philippines	2020	4	311
White Island	New Zealand	2019	2	321
Shiveluch	Russia	2019	4	3283
Anak Krakatoa	Indonesia	2018	3	813
Eyjafjallajökull	Iceland	2010	4	2119
Etna	Italy	2013	3	3350
Stromboli	Italy	2019	2	924
Puyehue-Cordón Caulle	Chile	2011	5	2236



Look at the table above.

(a) Write an SQL statement to display the name and last eruption of all volcanoes in Italy. [3]

(b) List the names of all the volcanoes which will be displayed when the query for part (a) is run. [2]

(c) Write an SQL statement to display all the fields for volcanoes in countries whose name begins with the letter "I". [3]

(a) **SELECT** name, lastErupted<sup>[1]</sup>  
**FROM** VolcanoTable<sup>[1]</sup>  
**WHERE** country = 'Italy'<sup>[1]</sup>

(b) Etna and Stromboli.<sup>[1]</sup>

(c) **SELECT** \*<sup>[1]</sup>  
**FROM** VolcanoTable<sup>[1]</sup>  
**WHERE** country LIKE 'I%'<sup>[1]</sup>

2.2.3

# USING SQL TO SEARCH FOR DATA

Records in this format can be searched using a Structured Query Language (SQL).

The format of an SQL statement is:

```
SELECT... field1, field2, field3...  
FROM... table  
WHERE... criteria
```

Using the Volcanoes table above, the SQL statement below will return a Results table showing all eruptions since 2019:

```
SELECT name, country, lastErupted, explosivityIndex  
FROM VolcanoTable  
WHERE lastErupted >= '2019'
```

Results table

name	country	lastErupted	explosivityIndex
White Island	New Zealand	2019	2
Shiveluch	Russia	2019	4
Stromboli	Italy	2019	2

You can also use Boolean operators to search for data. To find the lowest or most significant volcanoes:

```
SELECT name, country, lastErupted, elevationMetres  
FROM VolcanoTable  
WHERE explosivityIndex = 5 OR elevationMetres < 500
```

Results table

name	country	lastErupted	elevationMetres
Taal	Philippines	2020	311
White Island	New Zealand	2019	321
Puyehue-Cordón Caulle	Chile	2011	2236

## Wildcards

The wildcard **\*** is a substitute for ALL fields, e.g. **SELECT \***  
The Boolean condition **LIKE** is used with the wildcard **%**, which is a substitute for zero or more characters, e.g.  
**WHERE** name **LIKE** 'S%'  
finds all records with names beginning with S.

70 ClearRevise

OCR GCSE Computer Science J277 - Section 8

71



EXAMINATION PRACTICE

1. Complete the truth table for  $P = A \text{ OR } \text{NOT } B$  [4]

A	B	NOT B	P
0			
0			

2. A motorbike has two tyres, front (F) and rear (R). If the ignition (I) is on, and either of the tyres is below the minimum air pressure, a warning light is displayed.  
(a) Draw a logic circuit diagram for this scenario. [2]



(b) Complete the truth table for this scenario. [5]

Front tyre pressure low (F)	Rear tyre pressure low (R)	Ignition on (I)	Working space (F or R)	Warning light on (W)
0	0	0		
0	0	1		

3. Shona is using a high-level language to learn programming.  
(a) Describe what is meant by a ‘high-level language’. [3]  
(b) Explain **two** features of an IDE (Integrated Development Environment) that can help Shona to find or prevent errors in her programming code. [4]

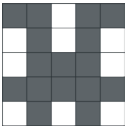
4. (a) Explain the difference between a compiler and an interpreter. [2]  
(b) PoundSoft is a software company selling accountancy software. The software is compiled rather than interpreted. Explain why they sell the software in this form. [4]

EXAMINATION PRACTICE ANSWERS

Section 1

- 1. D – Touch screen. [1]
- 2. B – A register in which the results of calculations are temporarily stored [1]
- 3. A – An area of internal HDD or SSD storage used when RAM is full. [1]
- 4. (a) Cache acts as a buffer between RAM and the CPU. Frequently requested data and instructions are transferred from RAM to cache. Cache is much faster than RAM so instructions and data can be accessed more quickly than from RAM. [3]  
(b) Cache is more expensive than RAM. [1]
- 5. For a computer’s boot up instructions // to hold the program in an embedded device // to store BIOS. [1]
- 6. (a) Control unit controls the input and output of data and the flow of data within the computer. It coordinates all the operations of the CPU, using clock timing signals to synchronise the stages of the Fetch-Decode-Execute cycle. [2]  
(b) The ALU carries out arithmetic operations such as adding two numbers, and logical operations such as AND, OR and NOT. [2]
- 7. One CPU may have more cache memory than the other which will speed up program execution as data and instructions can be retrieved much faster from cache. One CPU may have multiple cores, e.g. a dual-core computer will theoretically process twice as many instructions as a single core computer as the two cores can frequently process two instructions simultaneously. Disk access speed will affect the speed at which data is accessed and read from or written to storage e.g. SSD or HDD will affect this. A computer with more RAM will make less use of virtual memory, which slows down the execution of programs. Different architectures, for example PCs and Mac cannot be compared by clock speed alone as they perform differently at same clock speed. [4]  
(Tip: There are five points here – you only need two, but be sure to expand on each point you make. Just saying “One computer may have more cache” is not enough for two marks.)
- 8. SSD (or HDD). SSD is lightweight and unaffected by knocks or bumps as it has no moving parts. It runs with less power so increases the battery life of the tablet. It produces less heat when running so a fan is not required, saving space and weight inside the tablet. If HDD is chosen, they are fast, reliable and high capacity. An HDD may be less expensive to install making the overall tablet better value. [3]  
(Tip: Justification must match answer the answer you give. SSD is easier to justify here to gain 3 marks!)

Section 2

- 1. (a) 4,500 bytes [1] (b) 2,000 MB [1] (c) (i) 00011100 [1] (ii) Shifting one place right would divide the number by 2. [1]
- 2. (a) The set of characters or symbols that a computer can display using a particular representation, e.g. ASCII or Unicode. [1]  
(b) 160 bytes. [1]  
(c) (i) 1110, 1000. [2]  
(ii) Unicode uses more bytes (2 per character), which enables more characters/scripts to be represented. [2]
- 3. (a)   
(b) Lossy compression (JPG) would provide the smallest file size whilst maintaining a good quality image. Whilst some data is removed during the compression process, the image would still be recognisable. The smaller file size would mean it was able to download and display on a browser more quickly. Alternative compression methods such as PNG or GIF are acceptable with an explanation. [4]  
(c) Metadata is stored with the image data, to identify further information about the data, including its dimensions, bit depth, location data and file type. [2]
- 4. (a) Bit depth (or sample resolution) means the number of bits allocated to each recorded sample. [1]  
(b) The greater the number of bits, the more accurately the wave height of each sample can be recorded. This increases the overall quality of the recording as it will create a closer representation of the original sound. [2]  
(c) Sample rate is the number of samples taken each second. As the sample rate is increased, the file size will increase as each sample is saved at the given bit depth/resolution. [2]

# BAND DESCRIPTIONS AND LEVELS OF RESPONSE GUIDANCE FOR EXTENDED RESPONSE QUESTIONS

Questions that require extended writing use mark bands. The whole answer will be marked together to determine which mark band it fits into and which mark should be awarded within the mark band.

### Mark Band 3 – High Level (6–8 marks)

- Technical terms have been used precisely
- The answer is logical and shows an extensive understanding of Computer Science concepts, and principles
- The answer is almost always detailed and accurate
- All parts of the answer are consistent with each other
- Knowledge and ideas are applied to the context in the question
- Where examples are used, they help with understanding the answer
- Arguments and points are developed throughout the answer with a range of different perspectives. Different sides of a discussion are considered against each other

### Mark Band 2 – Mid Level (3–5 marks)

- The meaning of technical terms in the question has been understood
- The answer shows an understanding of Computer Science concepts
- Arguments and points are developed in the answer, but sometimes useful examples or related knowledge to the context have not been included
- Some structure has been given to the answer with at least one line of reasoning
- Sound knowledge has been effectively shown

### Mark Band 1 – Low Level (1–2 marks)

- The answer shows that technical terms used in the question have not been understood
- Key Computer science concepts have not been understood and have not been related to the context of the question
- The answer is only loosely related to the question and some inaccuracies are present
- Gaps are shown in Computer Science knowledge
- The answer only considers a narrow viewpoint or one angle
- The answer is unstructured
- Examples used are mostly irrelevant to the question or have no evidence to support them

### 0 marks

- No answer has been given or the answer given is not worth any marks

The above descriptors have been written in simple language to give an indication of the expectations of each mark band. See the OCR website at [www.ocr.org.uk](http://www.ocr.org.uk) for the official mark schemes used.

# INDEX

## Symbols

2D array 69

## A

abstraction 47, 49  
accumulator 3  
algorithmic thinking 47  
algorithms  
    compression 21  
    encryption 30  
    identifying 58  
    searching 54  
    sorting 55  
ALU 3  
amplitude 20  
analogue sound 20  
anticipating misuse 78  
anti-malware software 35  
architecture of the CPU 2  
Arithmetic Logic Unit 2  
arithmetic operators 62  
arrays 68  
ASCII 17  
    binary representation of 17  
assignment 61  
authentication 78

## B

bandwidth 23  
binary 11  
    addition 13  
    counting 12  
    representation of images 18  
    search 54  
    shifts 16  
    to denary 12  
    to hexadecimal 14  
bit 11  
bit depth 18, 20  
bitmap 18, 21  
blagging 34  
Bluetooth 29  
Boolean  
    conditions 62  
    data type 66  
    logic 82  
boundary data 80

breakpoint 85  
brute-force attack 34  
bubble sort 51, 55  
byte 11

## C

cabling 29  
cache 3, 4  
Caesar cipher 30  
camelCase 61  
capacity 9, 11  
case statement 63  
casting 66  
CD 8  
characters 17  
character set 17  
ciphertext 30  
client computer 26  
client-server network 26  
clock speed 2, 4  
Cloud 28  
coaxial cable 29  
colour depth 18  
commenting 79  
common errors 53  
comparison operators 62  
compiler 84  
compression  
    lossless 21  
    lossy 21  
    software 38  
computational thinking 47  
Computer Misuse Act 1990 43  
concatenation 67  
condition-controlled loop 65  
constants 61  
Control Unit 2  
copper coaxial cable 23  
Copyright Designs and Patents Act 43  
cores 2, 5  
cost 9  
count-controlled loop 64  
CPU  
    clock 2  
    instruction 2  
    performance 4  
cultural issues 40

## D

database 70  
data compression software 38  
Data Protection Act 2018 43  
data storage 11  
data types 66  
debugging 79, 85  
decomposition 47, 49  
defensive design 78  
defragmentation software 38  
denary  
    to binary 12  
    to hexadecimal 15  
denial of service attack 34  
identifying inputs 48  
DIV 62  
Domain Name Server 28  
drivers 37  
dual-coding iii  
durability 9

## E

Ebbinghaus iii  
editors 85  
embedded system 4  
encryption 30, 35, 38  
environmental issues 40  
erroneous data 80  
error  
    detection 85  
    diagnostics 85  
    overflow 13  
Ethernet 29  
ethical issues 40

## F

fetch-execute cycle 2, 4  
fibre optic cable 23, 29  
field 70  
file handling operations 72  
file management 37  
file server 26  
file size 19  
firewall 35  
float 66  
flowcharts 50  
format check 78

forms of attack 34  
for .. next 64  
frequency 20  
FTP 32  
Functions 73

**G**

gigabyte 11  
global variable 73, 74

**H**

hacking 34  
hard disk 8  
hardware 24  
Hertz 4, 20  
hexadecimal 14  
    to binary 14  
    to denary 15  
high-level language 84  
hosting 28  
HTTP / HTTPS 32

**I**

identifying algorithms 58  
identifying common errors 53  
IDEs 85  
if .. then .. else 63  
images 18  
IMAP 32  
indentation 79  
indexing  
    arrays 68  
    strings 67  
inputs 62  
input statement 62  
insertion sort 57  
instruction 2  
integer 66  
interception of data 34  
interface 37  
interference 29  
Internet 28  
interpreter 84  
invalid data 80  
IP address 28, 31  
IPv4 31  
IPv6 31  
ISP 28  
iteration 64

**K**

kilobyte 11

**L**

languages 84  
LAN (Local Area Network) 23  
latency 23  
layers 31  
least significant bit 12  
legal issues 40  
legislation 43  
length check 78  
levels of response guidance 94  
linear search 54  
local variables 73  
logical operations 3  
logic  
    diagrams 83  
    errors 53, 80  
    gates 82  
lossless compression 21  
lossy compression 21  
low-level language 84

**M**

MAC address 25, 31  
magnetic storage 9  
mail server 26  
maintainability 79  
malware 34  
MAR 3  
mathematical operations 3  
MDR 3  
megabyte 11  
memory 6  
    virtual 7  
    management 37  
merge sort 56  
mesh network 25  
metadata 19  
misuse 78  
MOD 62  
most significant bit 12  
multitasking 37

**N**

naming conventions 79  
nested loops 65  
networks 23

cabling 29  
client-server 26  
hardware 24  
layers 31  
P2P 27  
topology 25  
Network Interface Card 24, 31  
network performance 23  
nibble 11  
non-volatile 6  
normal data 80

**O**

OCR Reference Language 51, 63  
open source software 44  
operating systems 4, 37  
operators 62  
optical storage 9  
outputs 48, 62  
overflow 13

**P**

packets 23  
parameters 74  
passwords 35  
PC 3  
peer-to-peer network 27  
penetration testing 34, 35  
performance  
    of networks 23  
peripheral management 37  
permissions 34  
petabyte 11  
phishing 34  
physical security 34, 35  
pixel 18  
plaintext 30  
POP 32  
portability 9  
PPI 19  
presence check 78  
prettyprint 85  
primary storage 6  
print server 26  
print statement 62  
privacy issues 40  
procedures 73, 74  
processes 48  
programming fundamentals 61

proprietary software 44  
protocols 32  
protocol layers 31  
pseudocode 51, 63

**R**

RAM 4, 6  
random number generation 73  
range check 78  
real 66  
records 70  
reference language 51  
refining algorithms 80  
registers 2  
reliability 9  
resolution 19  
ROM 4, 6  
router 24, 31  
run-time environment 85  
runtime error 85

**S**

sample rate 20  
sample resolution 20  
scope of a variable 73  
searching algorithms 54  
secondary storage 8  
security 29, 35  
selection 63  
sequence 63  
server 26  
shifts 16  
shouldering 34  
signal strength 23, 29  
slicing strings 67  
SMTP 32  
social engineering 34  
software licences 44  
solid state storage 8, 9  
sound 20  
speed 9  
SQL 71  
SQL injection 34  
standards 32  
star network 25  
stepping 85  
storage  
    magnetic 9  
    optical 9  
    solid-state 9

stored program computer 3  
string 66  
string manipulation 67  
structured code 79  
structure diagrams 49  
subprograms 73, 79  
substrings 67  
switch 24  
switches 11  
syntax completion 85  
syntax errors 53, 80

**T**

TCP/IP 31  
terabyte 11  
test data 80  
testing 80  
text files 72  
theft of data 34  
topologies 25  
trace table 52  
translators 84, 85  
transmission media 29  
transmission speed 29  
truth table 82  
twisted pair 29  
two-dimensional arrays 69  
type check 78

**U**

Unicode 17  
units 11  
URL 28  
USB flash drive 8  
user access levels 35  
user interface 37  
user management 37  
utility software 38

**V**

validation 34, 78  
variables 61  
    local and global 73  
    watch 85  
verification 79  
virtual memory 7  
virus 34  
volatile 6  
Von Neumann architecture 3

**W**

WAN (Wide Area Network) 23  
watch 85  
web server 26  
while .. endwhile 64  
Wi-Fi 29  
WIMP 37  
wireless access point 24  
wireless connectivity 29  
World Wide Web 28

# EXAMINATION TIPS

With your examination practice, use a boundary approximation using the following table. Be aware that boundaries are usually a few percentage points either side of this.

Grade	9	8	7	6	5	4	3	2	1
Boundary	90%	80%	70%	60%	50%	40%	30%	20%	10%

1. Be aware of command words at the back of the specification. If 'describe' or 'explain' questions are given you need to expand your answers. To help you justify your responses, aim to include words such as BECAUSE... or SO... in every answer because this forces you to justify your point, so you get additional marks. See how well it works!
2. Explain questions such as 'explain why this is the most appropriate...' do not require just a list of benefits. Instead you should identify the benefits and then expand each one, applying them to the scenario or context.
3. Full answers should be given to questions – not just key words. Make your answers match the context of the question.
4. Algorithm questions require an actual algorithm not a repetition of the question.
5. If a question explicitly asks for an algorithm to be written in pseudocode, then it will not gain marks if it is written as a flowchart. Equally, a question that asks for an algorithm to be written as a flowchart will not gain marks if answered with pseudocode.
6. If you have difficulties with algorithm questions, remember that you will gain marks (where appropriate) for input and output statements.
7. The statement **INPUT = variableName** will not gain marks in pseudocode as the variable name needs to be on the left of the assignment operator. E.g. **variableName = INPUT**.  
**INPUT variableName** is an acceptable alternative.
8. String concatenation is not enough for an output e.g. **print(hello + name)** – the string must be in quotes, e.g. **print("hello " + name)**
9. Generic answers are not sufficient. E.g. If a question asks for a description of the function of a router, an answer 'it connects devices together' is not sufficient. Instead answers should describe how routers are used to receive packets from computers, read the destination address of each and then forward each packet to its destination. Faster, bigger and cheaper are not very useful responses unless you justify your point.
10. The pseudocode you write does not need to match any precise syntax as long as it can "be reasonably inferred by a competent programmer".
11. Arrays will always start at zero, not 1.
12. Remember that a nested loop completes fully for each iteration of the outer loop.
13. In pseudocode, **input("enter name")** will not gain marks as the result needs to be assigned to a variable to store it – e.g. **name = input("enter name")**. Equally, two values cannot be input at the same time as a variable will only store one value. Instead, use **a = INPUT("Enter a")** then **b = INPUT("Enter b")**.  
**INPUT a, b** would be an acceptable alternative.
14. A common error in IF statements is **if name != "Sam" or "sam" then**. This should be: **if name != "Sam" or name != "sam" then**
15. Be careful with quotes around strings. E.g. **choice = A** (which assigns a variable) is very different to **choice = "A"** (which assigns a string).

**Good luck!**