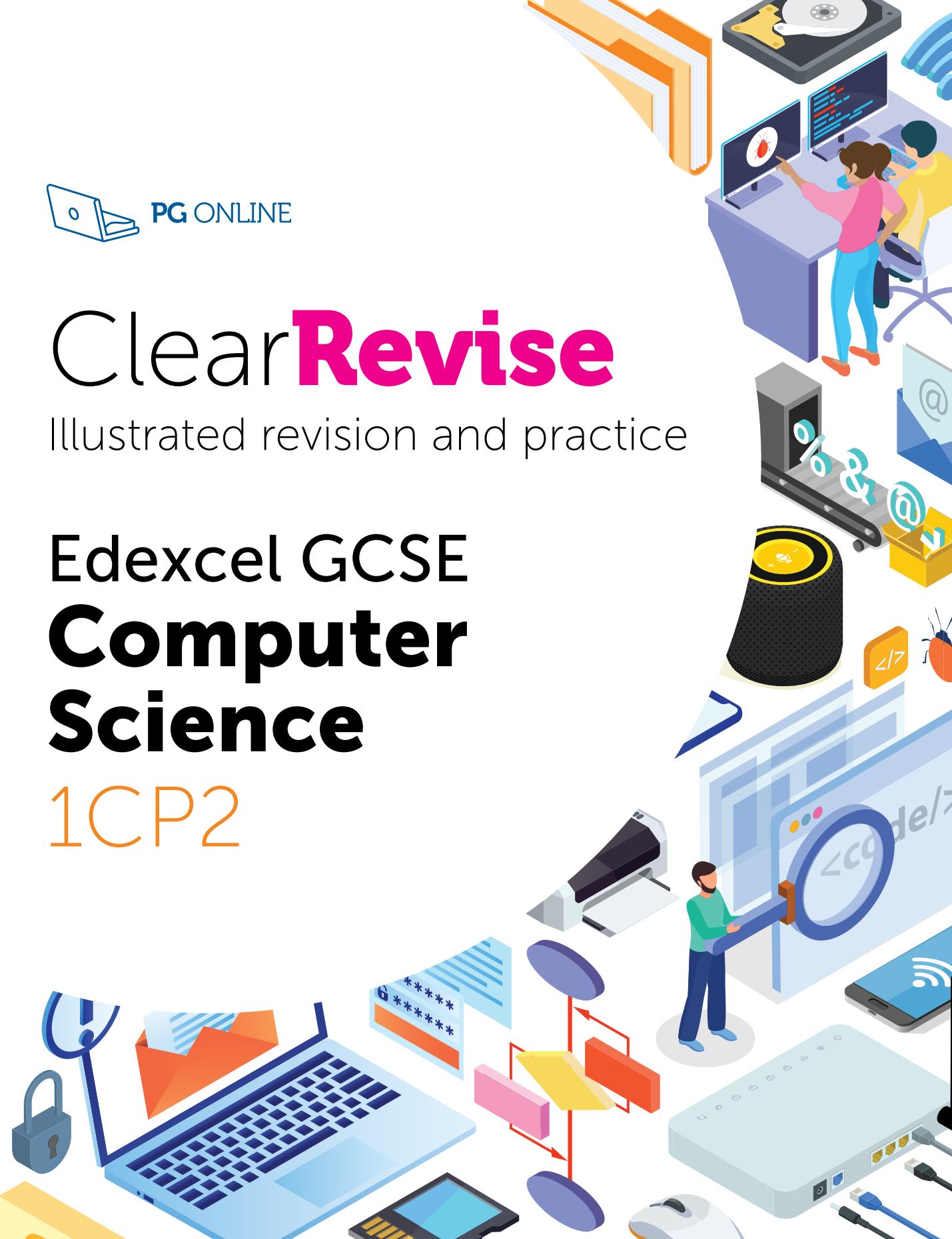
PG ONLINE

ClearRevise

Illustrated revision and practice

Edexcel GCSE
Computer Science
1CP2



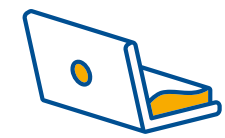
ClearRevise™

Edexcel GCSE
Computer Science 1CP2

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
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.

ACKNOWLEDGEMENTS

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 or material in this textbook.

Examination structure: © Pearson Education Ltd

Design and artwork: Jessica Webb / PG Online Ltd
Graphics / images: © Shutterstock

First edition 2020. 10 9 8 7 6 5 4 3 2 1
A catalogue entry for this book is available from the British Library
ISBN: 978-1-910523-28-5

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.[1] 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.[2] 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.[3] 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.[4]

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.[5] 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.[6]

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.[5] If you choose to study with friends, choose carefully – effort is contagious.[7]

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

Paper 1 Principles of Computer Science

Topic 1 Computational thinking

☑

Specification

Topic 2 Data

☑

Topic 3 Computers

Topic 4 Networks

Topic 5 Issues and impact

Paper 2 Application of Computational Thinking

Topic 6 Problem solving with programming

MARK ALLOCATIONS

Green mark allocations[1] on answers to in-text questions through this guide help to indicate where marks are gained within the answers. A bracketed '1' e.g. [1] = 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
PRINCIPLES OF COMPUTER SCIENCE
(1CP2/01)

Information about Paper 1

Written exam: 1 hour and 30 minutes
75 marks
50% of GCSE

Specification coverage

Computational thinking, data, computers, networks, and issues and impact.

The content for this assessment will be drawn from Topics 1 to 5 of the specification.

Questions

This paper consists of five compulsory questions, each one focused on one of the major topic areas. The questions will use multiple-choice, short answer and extended response styles. Tables and diagrams will also be used.

PSEUDOCODE AND ALGORITHMS

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 on page 3 could be expressed like this:

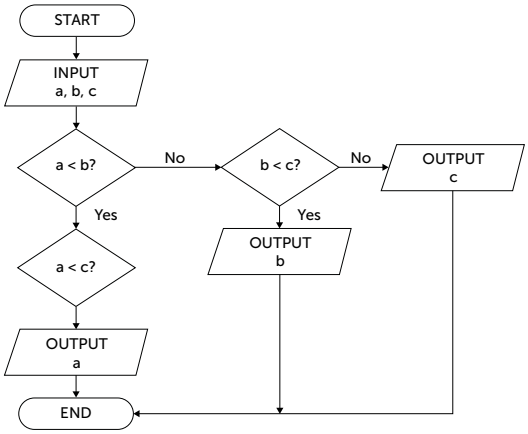
```
input startNumber, endNumber, step
set number to startNumber
while number <= endNumber
   output(number)
   add step to number
endwhile
```

Follow and write algorithms

An **algorithm** is a sequence of steps that can be followed in order to complete a task. Examples include recipes, assembly instructions and directions.

An algorithm is not the same as a computer program. A computer program is one way of implementing an algorithm in a particular language, but it is the series of instructions and the order of those instructions that are the basis of any algorithm.

In this paper, you may be given a flowchart and asked to determine the purpose of the algorithm.



Determine the purpose of the algorithm shown in the flowchart. [1]

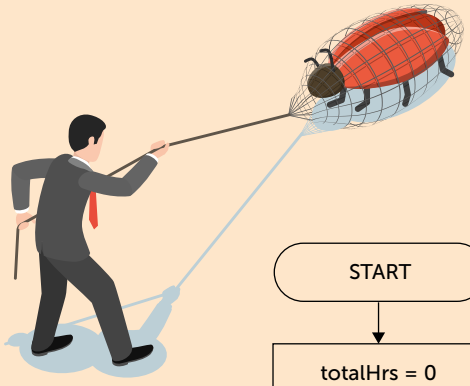
The purpose is to output the smallest of three values.[1]

This algorithm uses a **nested selection** structure.

The IF statement (or decision) has an IF statement nested inside it. In this example, the ELSE statement also has a nested IF statement.

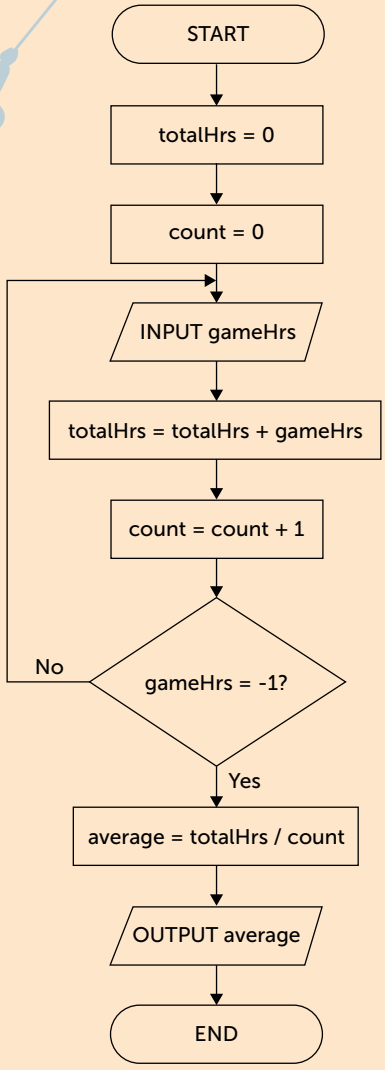
TRACE TABLES

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 below. 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.



A trace table, shown below, has identified an error with the flowchart.

Describe how the algorithm could be corrected. [3]

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

Download the Python program **Sec 1.2 Trace table game hrs corrected.py** from **www.clearrevise.com**

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.

EXAMINATION PRACTICE

1. An algorithm is given below.

```
01  array 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 < len(aList):
06    if aList[n] == x:
07            found = True
08            print(x, "found at position ", n)
09    else:
10            n = n + 1
11  if not found:
12    print("Invalid number")
```

(a) Explain the purpose of this algorithm. [2]

(b) At line 05, what is the value of `len(aList)`? [1]

(c) The user enters 9 at line 04. How many times is the **while** loop performed? [1]

(d) Explain how the use of the variable named `found` makes the algorithm more efficient. [2]

2. An array `names` holds `n` items. An algorithm for a bubble sort is given below.

```
01  swapMade = True
02  while swapMade
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]

TWO'S COMPLEMENT SIGNED INTEGERS

An **unsigned** representation of a binary number can only represent positive numbers. A **signed** integer can represent both positive and negative numbers.

Two's complement

Using two's complement, the leftmost bit is the **sign bit**. If it is 1, the number is negative. If it is 0, the number is positive.

Two's complement works in a similar way to numbers on an analogue counter. When it shows 0000, moving the wheel forwards by one, would create a reading of 0001. Moving it backwards would create a reading of 9999, which is interpreted as –1.

1111 1101	=	–3
1111 1110	=	–2
1111 1111	=	–1
0000 0000	=	0
0000 0001	=	1
0000 0010	=	2
0000 0011	=	3

The maximum range that can be represented with 8 bits is –128 to 127 because the leftmost bit is used as the sign bit, with a value of –128, leaving only 7 bits to represent the positive part of the number. A 1 as the leftmost bit indicates the number is negative.

In an 8-bit byte, the leftmost bit represents = –128

Converting a negative denary number to binary

Work out the positive binary equivalent of the number, flip all of the bits and add 1.

For example:

Target: –21
Positive 21: 0001 0101
Flip the bits: 1110 1010
Add one: 1
Convert: 1110 1011

Converting a negative two's complement binary number to denary

The same method works the other way. Flip all of the bits and add 1. Then work out the denary equivalent as normal.

For example:

Target: 1110 1110
Flip the bits: 0001 0001
Add one: 1
Convert: -0001 0010 = –18

1. Convert the following denary numbers to signed binary integers:
   (a) –128 [1] (b) –57 [1] (c) –9 [1]
2. Convert the following signed binary integers to denary:
   (a) 1000 0011 [1] (b) 1110 0110 [1] (c) 1110 0101 [1]

1. (a) 1000 0000[1], (b) 1100 0111[1], (c) 1111 0111[1] 2. (a) –125[1], (b) –26[1], (c) –27[1]

2.1.6

USES OF HEXADECIMAL

Hexadecimal numbers are easier to read and remember than binary, so they are used in the following situations:

- Colour values in photo editing software and HTML
- MAC addresses
- Memory address locations in assembly language

Hexadecimal colour number: ED468C

2.2.1

CHARACTER ENCODING

Each character on a keyboard has a binary code which is transmitted to the computer each time a key is pressed.

Some of the characters and their codes, known as the **character set**, for the standard keyboard are given opposite. The **ASCII** character set consists of 128 characters, each using 7 bits to uniquely represent them. ASCII stands for American Standard Code for Information Interchange. Extended ASCII uses 8 bits (or 1 byte) per character.

Use the ASCII table opposite for this question.

(a) Show how the word CAGE is represented in 7-bit ASCII. Give your answer in binary. [1]

(b) One byte is used to store one character. The leftmost bit of each byte is set to 0. State how many bytes would be used to store the phrase "BIRD CAGE". [1]

(c) The uppercase character 'T' in ASCII is represented by the denary value 84. State the denary value for the character 'R'. [1]

(a) 1000011 1000001 1000111 1000101[1]

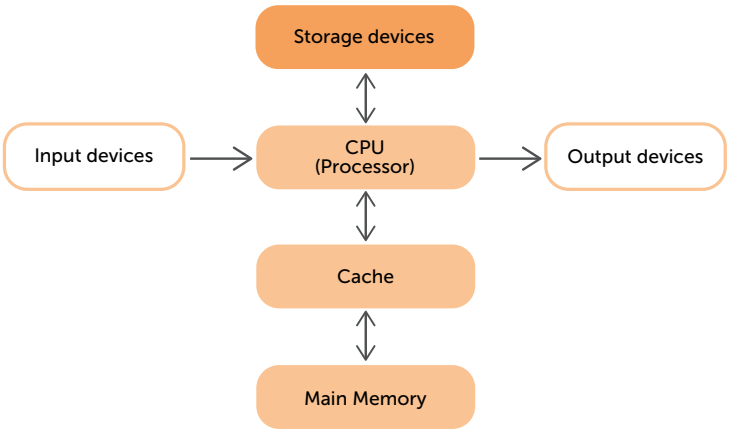
(b) 9 bytes.[1] *(The Space character has the denary code 32 and occupies one byte.)*

(c) 'R' is 82.[1]

SYSTEMS ARCHITECTURE

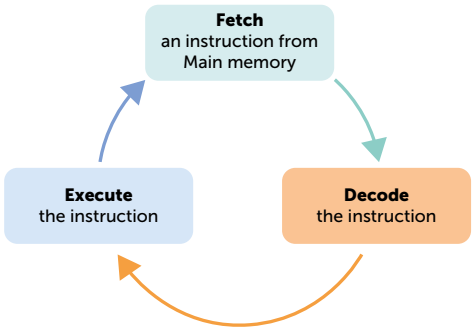
The purpose of the CPU

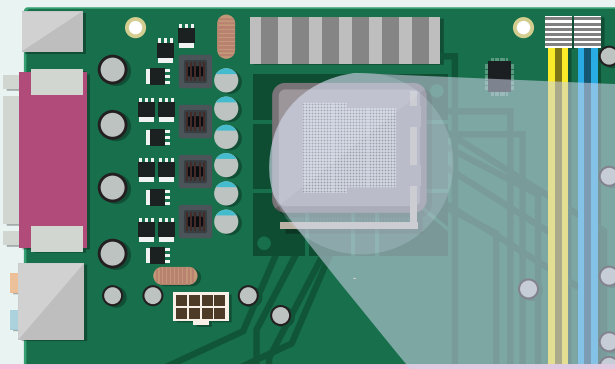
The purpose of the **Central Processing Unit** (**CPU**) is to continuously fetch, read and execute instructions stored in memory by repeatedly carrying out the **fetch-execute cycle**. The CPU contains the **Arithmetic Logic Unit** and the **Control Unit**, in addition to several general-purpose and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



The fetch-execute cycle

Every CPU instruction is fetched from memory. Once fetched, it is decoded 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 4.5 GHz processor has 4,500,000,000 clock cycles or 'ticks' a second. This is known as the clock speed.

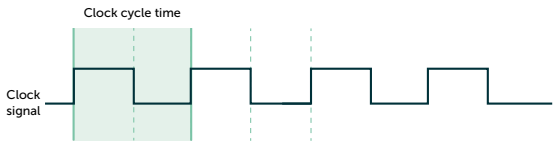
Von Neumann architecture

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

CPU component	Function
ALU (Arithmetic Logic Unit)	Carries out mathematical and logical operations including AND, OR and NOT, and binary shifts. It compares values held in registers.
CU (Control Unit)	Coordinates all of the CPU's actions in the fetch-decode-execute cycle and decodes instructions. Sends and receives control signals to fetch and write data.
Clock	The clock regulates the speed and timing of all signals and computer functions.
Registers	Very small, very fast memory locations. Registers are built into the CPU chip to temporarily store memory addresses, instructions or data. They are used in the fetch-execute cycle for specific purposes.
Address, **data** and **control buses**	Wires used to transfer data, instructions, memory addresses (of data and instructions), and control signals from one component to another.

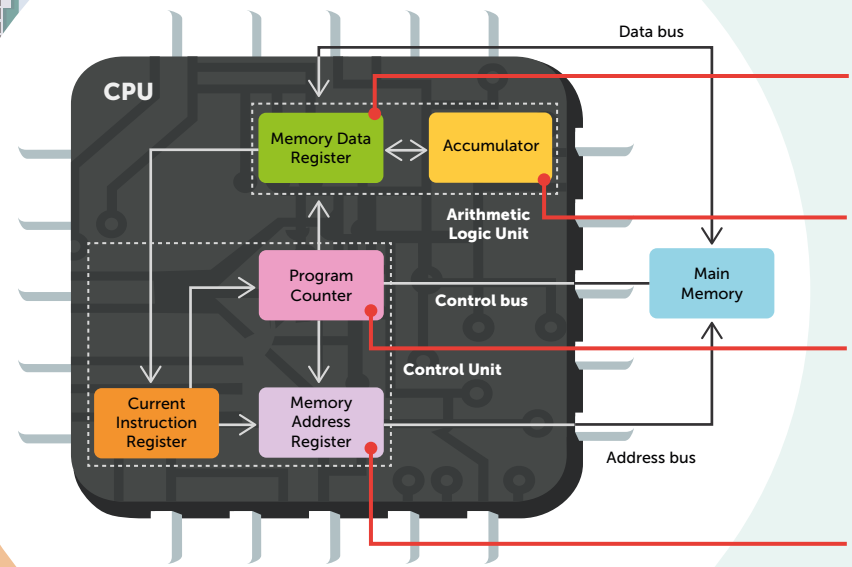
Clock speed

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.



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.[1] The CPU fetches the instruction and data from memory[1] and stores them in its registers[1]. The PC is incremented[1]. The Control Unit decodes the instruction[1] and the instruction is executed[1].



MDR holds data or a program instruction when it is fetched from memory or data that is waiting to be written to memory.

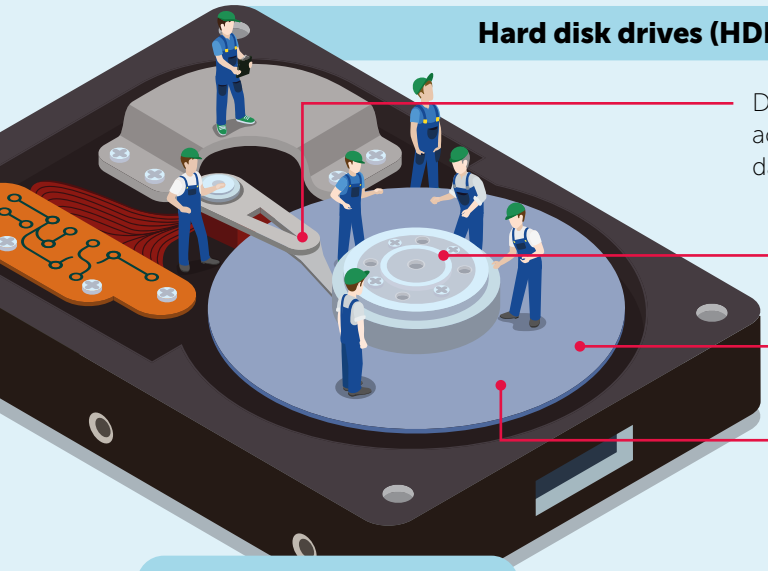
The **accumulator** is a register in which results of operations carried out in the **ALU** are stored.

PC is a register which holds the **memory address** of the next instruction to be processed.

MAR holds the address (location in memory) of the current instruction or piece of data to be fetched or stored.

DEVICE OPERATION

Hard disk drives (HDD)



Drive read/write head moves into position across concentric tracks which hold the data. This movement takes time.

Drive spindle rotates disk. Moving parts cause issues if dropped.

Magnetic platter stores data. Affected by heat and magnetic fields.

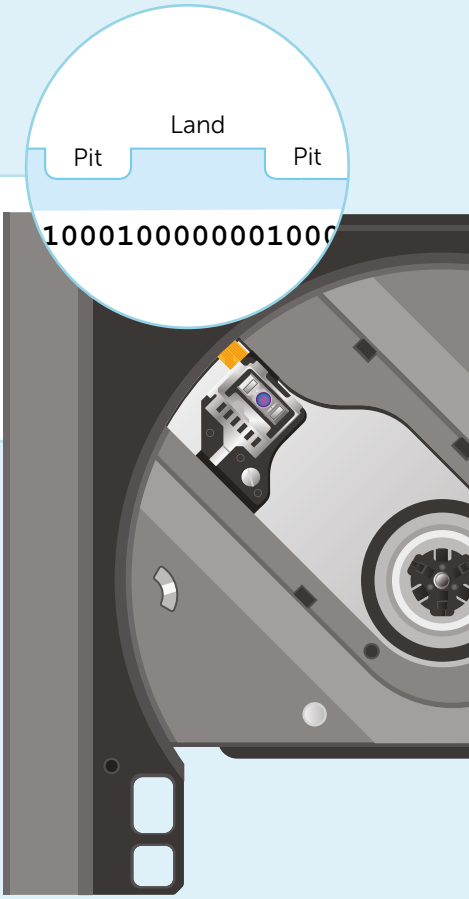
Iron particles on the disk are magnetised to be either north or south, representing 0 or 1.

Solid state disks (SSD)

SSDs look like a standard circuit board.
They use electrical circuits to persistently store data. These use microscopic transistors to control the flow of current. One that allows current to flow is a 1. Where current is blocked, a 0 is represented.

Optical drives (CD / Blu-ray)

An optical drive uses a laser to reflect light off the surface of the disk. One long spiral track contains pits and lands. When the laser beam hits the curved start or end of a pit, the light is refracted and a 1 is recorded. Where light is reflected back directly from the flat bottom of a **pit**, or from an area of the track with no pit (a **land**) a 0 is recorded.



1. Explain why hard disk drives have been largely replaced by solid state drives in portable devices. [4]

Hard disk drives have lots of moving parts[1] which can cause problems if dropped or shaken[1]. The read/write head moves across the disk and can scratch the disk irreparably if accidentally moved too violently whilst in operation.[1] Moving the head across the disk to read or write data reduces the access speed[1] that can be achieved with solid state devices that have no moving parts. The cost and capacity of solid-state storage is improving.[1]

METHODS OF PROTECTING NETWORKS

Networks require security measures to prevent unauthorised access. This ensures the privacy of data that is transferred within the network. Using a combination of methods provides greater protection against threats.

Using the correct settings and levels of permitted access

Access rights can be set up to allow different people different levels of use. Basic access rights can be configured as Read only, Read and write (or Edit), Execute (which allows a program to be run) and Full access which enables such a user to edit the rights of others.

Restrictions can be placed on access to drives, folders and files on an organisation's system. Ensuring these access levels are correct and not more than absolutely necessary for the job any individual is doing helps to narrow any risk.

Physical security

Locks on doors to server rooms and data centres provide a good level of physical defence. Other measures include **keypads**, card or **fob entry systems**, **fencing**, security guards and **barriers**.

Biometrics such as iris scanners, facial recognition and fingerprint readers authenticate your body measurements as your own. They may be used in combination with locks.

Give **two** security methods an organisation may use to protect data. [2]

Door entry locks[1], biometrics[1], security staff[1], CCTV[1], fire protection systems[1], passwords[1], secondary backup generators[1], off-site backups[1] two-factor authentication[1].

Firewall

A firewall is a software or hardware device that monitors all incoming and outgoing network traffic. Using a set of rules, it decides whether to block or allow specific data packets. By opening and closing ports, it can block traffic from disallowed connections from accessing the network, as well as blocking outgoing communications from the network, to make sure that only authorised traffic is permitted.

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.

For each of the following offences, state which section of the Computer Misuse Act 1990 would apply.

(a) Theft and resale of customer data from an organisation's computer. [1]

(b) Correctly guessing the password of a manager's office account. [1]

(c) Rewriting software programs to remove activation keys. [1]

(a) Section 2 [1]. (b) Section 1 [1]. (c) Section 3 [1].

Privacy, ownership and consent

Data collection is subject to the Data Protection Act to protect the privacy of data subjects. Organisations can be required to obtain consent to collect and store personal data from the individuals concerned. A complete data set of personal information may belong to the organisation who gathered it, but individuals have a right to their own data. It is often unclear who owns data.

TOPICS FOR PAPER 2
APPLICATION OF COMPUTATIONAL THINKING
(1CP2/02)

Information about Paper 2

Written exam: 1 hour and 30 minutes
75 marks
50% of GCSE

Specification coverage

Problem solving with programming

The content for this assessment will be drawn from Topic 6 of the specification.

Questions

This practical paper requires students to design, write, test and refine programs in order to solve problems.

Students will complete this assessment onscreen using their Integrated Development Environment (IDE) of choice.

They will be provided with:

- Coding files
- A hard copy of the question paper
- The Programming Language Subset (PLS) – as an insert in the question paper and in electronic format.

Students should then answer the questions onscreen using Python 3.

This assessment consists of six compulsory questions.

Python question files are held in the folder Python Exam downloadable from www.clearrevise.com.

Python solution files are held in the folder Python Exam downloadable from www.clearrevise.com.

DEVELOP CODE

Paper 2 is a practical programming exam. In order to do well in this paper, you need to be proficient in all the following skills:

- Be able to use decomposition and abstraction to analyse, understand and solve problems.
- Be able to read, write, analyse and refine programs written in a high-level programming language (Python in this course).
- Be able to convert algorithms (flowcharts, pseudocode) into programs.
- Be able to use techniques (layout, indentation, comments, meaningful identifiers, white space) to make programs easier to read, understand and maintain.
- Be able to identify, locate and correct program errors (logic, syntax, runtime).
- Be able to use logical reasoning and test data to evaluate a program's fitness for purpose and efficiency (number of compares, number of passes through a loop, use of memory).

The examples and questions in this section will make sure you are up to scratch in Python syntax and the techniques that you will need.

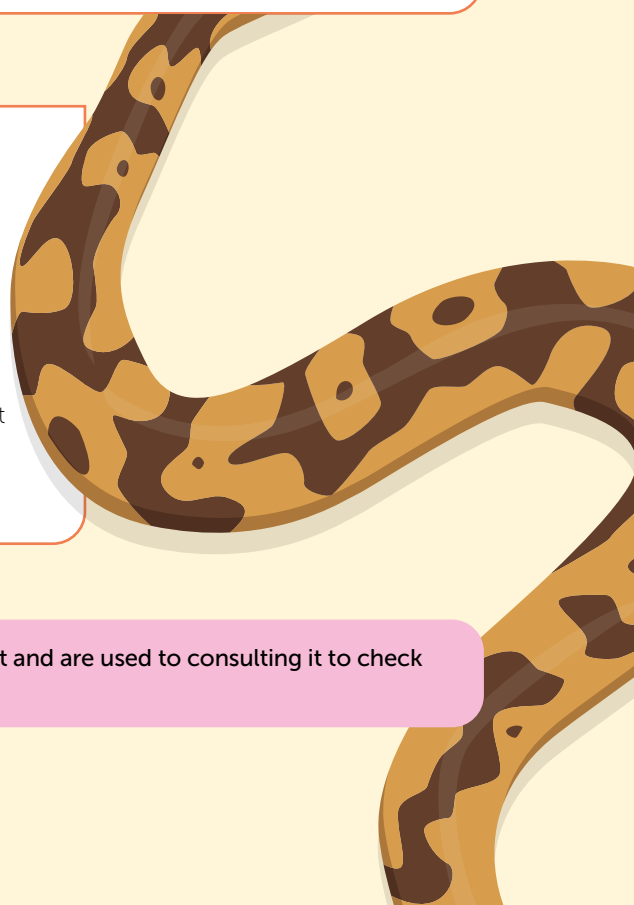
Python – A high-level language

A **high-level language** is one that is generally suited to the types of problem that you need to solve using computer programs.

The Python language subset

When you start learning to program in Python for this course, you will need a copy of the **Programming Language Subset** (**PLS**).
This document specifies which parts of Python 3 you may need in the exam, and how the different Python statements are written. It may be issued to you by your teacher, or you may need to download it from the Edexcel GCSE Computer Science website.

You will be given a copy of the Programming Language Subset with your exam paper. You should use this as a reference to remind you how to write a particular statement with the correct syntax.



Make sure you are familiar with every part of the PLS document and are used to consulting it to check how to write a particular statement correctly.

EXAMINATION PRACTICE

1. Load and execute the file **Exam Practice 6.1 Amend program error**, shown below:

```
total = 0
x = 0
while x != 100:
 total = total + x
 x = x + 3
 print("x =",x, "total =",total)
print("Total =",total)
```

Amend the program so that it works as intended. Save your program with a suitable name (e.g. **Answer 6.1 Qu1.py**) in a new folder. **You should save all the programs you amend.** [2]

2. To determine whether a year is a Leap Year, with 29 days in February, the following rules are applied:
   - The year must be evenly divisible by 4 but...
   - if the year can also be evenly divided by 100, it is not a Leap Year,

     unless...
   - the year is also evenly divisible by 400, in which case it is a Leap Year.

   According to these rules, the years 1000, 1600 and 2000 are Leap Years, while 1800, 1900 and 2100 are not Leap Years.

   Load and complete the Python program **Exam Practice 6.2 Qu 2 Leap Year.py** to determine whether a year input by the user is a Leap Year. Save it with a different name in your own folder.

```
# Accept a year entered by the user
year = int(input("Enter Year: "))

# Check if this is a Leap Year
if year % 4 == 0 and  ..........................
    print(year, "is a Leap Year")
elif year % 400 == 0:
    print(year, ..........................)
.........
    print(year, "is not a Leap Year")
```

[4]

3. The partially completed program **Exam Practice 6.2 Qu 3 Validate a user password.py** validates a user password.

   The user is permitted three attempts to enter the password correctly before being locked out.

   Complete the missing lines in the program **Exam Practice 6.2 Qu 3 Validate a user password.py** [4]

*You can find working programs in the folder **Python Exam Practice Finished Solutions**. There are sometimes alternative correct solutions which would score marks.*

TWO-DIMENSIONAL ARRAYS / LISTS

An array may have two or more dimensions. A two-dimensional array or list named sales could hold the number of properties sold each quarter (Jan–March, April–June, July–September, October–December) by three different branches of an estate agent. An index number is used to reference an array value.

	Index	0	1	2	3
Three branches	0	56	87	92	43
	1	167	206	387	54
	2	22	61	52	14

The index for both row and column of the array/list starts at 0. In Python the list may be defined as:

```
sales = [[56,87,92,43],[167,206,387,54],[22,61,52,14]]
```

The number of properties sold in Quarter 4 by Branch 1 is held in `sales[0][3]` and has the value 43.

1. The three branches of the estate agency are known as Branch A, Branch B and Branch C.
   (a) Write code to output the sales figure for Branch C for the period April–June. [1]
   (b) What will be output? [1]
2. Write a program to ask a user to enter the name and five race times in seconds for each of 3 competitors, and display the average time for each competitor. [8]

1. (a) `print(sales[2][1])`[1] (b) 61[1]
2.
```
name = ["","",""][1]
totalTime = [0,0,0][1]
averageTime = [0,0,0][1]
raceTime = [ [0,0,0,0,0],
             [0,0,0,0,0],
             [0,0,0,0,0] ][1]
for c in range(3):[1]
  name[c] = input("Enter competitor name: ")[1]
  for race in range(5):[1]
     raceTime[c][race] = float(input("Enter race time: "))[1]
     totalTime[c] = totalTime[c] + raceTime[c][race][1]
  averageTime[c] = round(totalTime[c] / 5, 2)[1]
  print("Average race Time for ",name[c], averageTime[c])[1]
```

LIST METHODS

Python has built-in list methods for creating an empty list, adding and deleting items. The ones you need to know about are listed in the Programming Language Subset.

The following program illustrates the use of each of these methods.

```
#Section 6.3 list methods
#create an empty list
mylist = []
#or, you could write:  myList = list()

#append an item to the end of a list
mylist.append(7)
mylist.append("George")
print(mylist)

#insert an item just before an existing one at index 1
#the first item is at index 0
mylist.insert(1, "Dodgson")
print(mylist)

#delete the first item (which has index 0) in the list
del mylist[0]
print(mylist)

#create a list containing 10 zeros
listB = [0]*10
print("listB:",listB)
```

3. The above program prints five lines. The first line printed is:

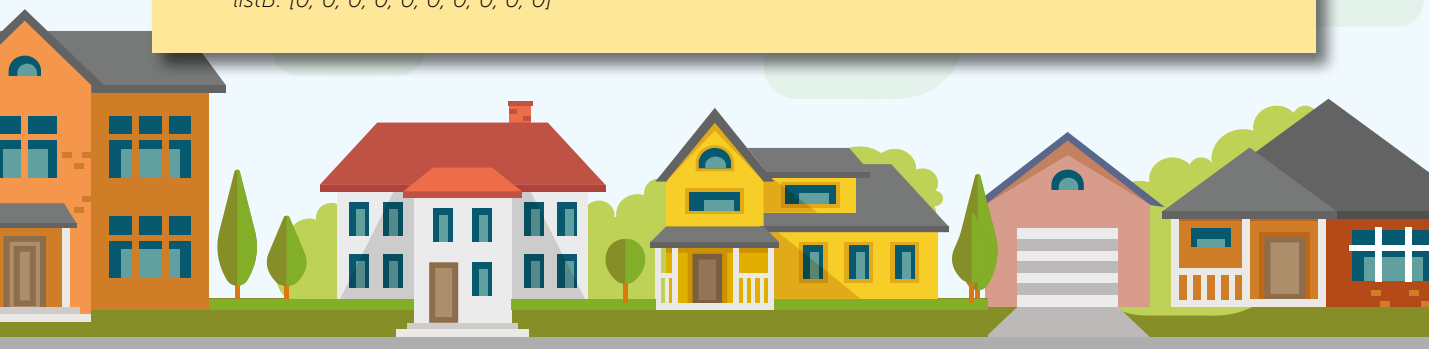
   [7, 'George']

   Write the next three lines printed. [3]

3. [7, 'Dodgson', 'George'][1]

['Dodgson', 'George'][1]

listB: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0][1]



EXAMINATION PRACTICE ANSWERS

Topic 1

1. (a) It performs a linear search on the list for an item entered by the user. If the item is not found, it prints "invalid number". [2]
   (b) 13 (the number of items in the list). [1]
   (c) 4 times. [1]
   (d) As soon as the item is found, the print statement at line 08 is executed and the while loop is exited. If there was no flag, the linear search would continue searching the whole list even after the item was found. [2]

2. (a) It acts as a 'flag' which is set to False when a pass through the list is made and no items are swapped, meaning that the list is now sorted. [2]
   (b) temp = names[index]
   names[index] = names[index+1]
   names[index+1] = temp [3]
   (c) Adam Edna Charlie Jack Ken Maria Victor
   Adam Charlie Edna Jack Ken Maria Victor [2]
   (d) 3 passes. Swaps are made on the first two passes. The list will be sorted after the second pass, and on the third pass, no swaps are made, so swapMade is set to False and the while loop terminates. [2]

3. (a) [4]

num	a	b	ans
	0	0	
3	3	1	
8	11	2	
2	13	3	
5	18	4	
-1			4.5

   (b) It calculates the average of the numbers input by the user. [1]

4. [4]

A	B	C	D = A OR B	E = NOT (A OR B)	F = B AND C	X = E OR F
0	0	0	0	1	0	1
0	0	1	0	1	0	1
0	1	0	1	0	0	0
0	1	1	1	0	1	1
0	0	0	0	1	0	1
1	0	1	1	0	0	0
1	1	0	1	0	0	0
1	1	1	1	0	1	1

Topic 2

1. 4.5 × 1024 = 4,608 bytes. [1]

2. 'C' is three characters before 'F', so deduct 3 from the value for 'F'. 100 0011. [2]

3. (a) (i) A signed integer would be most appropriate as this accommodates for negative values. [1]
   (ii) An 8-bit signed integer has minimum and maximum values of -128 to 127 which would be enough to store the highest and lowest recorded temperatures and more. [2]
   (b) D. 1111 1010 [1]

BAND DESCRIPTIONS AND LEVELS OF RESPONSE GUIDANCE FOR EXTENDED RESPONSE QUESTIONS

Level	Description	Mark range
3	• Thoughts, explanations, descriptions and ideas are consistent throughout the response • Clear explanations or descriptions • Evidence and examples are given to support explanations and descriptions • Logically structured response • Both advantages and disadvantages considered if required • Points and examples included that are relevant to the question • Points discussed / explained • At least three points typically required	**6–8 marks**
2	• Logically structured response • Clear and accurate explanations or descriptions • Both advantages and disadvantages considered if required • At least two points typically required	**3–5 marks**
1	• A description of some points has been given • Advantages or disadvantages briefly considered if required • At least one point typically required	**1–2 marks**
0	No answer has been given or the answer given is not worth any marks	**0 marks**

The above descriptors have been written in simple language to give an indication of the expectations of each mark band. See the Pearson Edexcel website at **https://qualifications.pearson.com** for the official mark schemes used.

INDEX

EXAMINATION TIPS

With your examination practice, apply a boundary approximation using the following table. This table is calculated using a rounded average of past years' boundaries for the 1CP1 GCSE course.

Be aware that boundaries can vary annually.

Grade	9	8	7	6	5	4	3	2	1
Boundary	70%	60%	52%	45%	38%	31%	23%	15%	8%

1. Read each question carefully. Some students give answers to questions they think are appearing rather than the actual question. Avoid simply rewriting a question in your answers or repeating examples that are already given in the question.
2. Understand the requirements 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 connective 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! 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.
4. Make your answers match the context of the question.
5. Avoid repetition of responses where more than one response is required.
6. Use technical language and keywords rather than informal language such as 'things' or 'stuff'.
7. Algorithm questions require an actual algorithm not a repetition of the question. 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.
8. If you have difficulties with algorithm questions, remember that you will gain marks (where appropriate) for input and output statements.
9. Generic answers are not sufficient. For example, 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. Arrays will always start at zero. Not one.
11. Remember that a nested loop completes fully for each iteration of the outer loop.
12. Be careful with quotes around strings. For example `choice = A` (which assigns a variable) is very different to `choice = "A"` (which assigns a value).
13. Remember when working out calculations that orders of precedence (BIDMAS) apply.
14. Write clearly and draw neatly. If the examiner can't read or understand what you have written you will get no marks

Good luck!

Edexcel GCSE **Computer Science** 1CP2

ClearRevise™

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
- Free Python solutions pack at clearrevise.com

Experience + science + beautiful design = better results

Absolute clarity is the aim with a new generation of revision guide. 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.

Past examination 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. Questions on all topics assessing knowledge, application and analysis are all specifically and carefully devised throughout this book.

www.clearrevise.com

ISBN 978-1-910523-28-5

PG ONLINE