Endorsed for
Pearson Edexcel Qualifications

PEARSON EDEXCEL INTERNATIONAL GCSE (9–1)

COMPUTER SCIENCE

Student Book

David Waller, Chris Charles, Pete Dring, Alex Hadwen-Bennett, Jason Welch, Shaun Whorton
Series Editor: Ann Weidmann

eBook included

PEARSON EDEXCEL INTERNATIONAL GCSE (9–1)

COMPUTER SCIENCE

Student Book

David Waller
Chris Charles
Pete Dring
Alex Hadwen-Bennett
Jason Welch
Shaun Whorton

SERIES EDITOR
Ann Weidmann

Published by Pearson Education Limited, 80 Strand, London, WC2R 0RL.

https://www.pearson.com/international-schools

Copies of official specifications for all Pearson Edexcel qualifications may be found on the website: https://qualifications.pearson.com

First published 2020

22 21 20
10 9 8 7 6 5 4 3 2 1

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN 978 1 292 31022 0

Printed in Slovakia by Neografia

Acknowledgements

Text Credits
Loon LLC: Extracts from Loon LLC, © Loon LLC, 256; **McKinsey & Company:** Stefan Heck, Sri Kasa, Dickon Pinner, Creating Value in the Semiconductor industry, © 2011, McKinsey & Company, 264.

Image Credits
(key: b-bottom; c-center; l-left; r-right; t-top)

Cover Image: Shutterstock/PopTika

123RF: Anton Starikov/123RF 160bc, Jan Mikš/123RF 54, Kbuntu/123RF 214, Lajo_2/123RF 71, Scyther5/123RF 123, Scanrail/123RF 238, Stockyimages/123RF 69; **Alamy Stock Photo:** Jonathan Ball/Alamy Stock Photo 168r, Martin Moxter/imageBROKER/Alamy Stock Photo 248, Paul Broadbent/Alamy Stock Photo 168l, Richard Levine/Alamy Stock Photo 52t, Stuart Kinlough/Alamy Stock Photo 2, 32, 108, 158, 200, 242; **Python Software Foundation:** Copyright ©2001-2019. Python Software Foundation 51, 94, 96, 101; **Shutterstock:** Adisa/Shutterstock 160tl, Asharkyu/Shutterstock 245, Anton Gvozdikov/Shutterstock 265, Burlingham/Shutterstock 34, Baloncici/Shutterstock 106, vii, Bloomicon/Shutterstock 160br, Brian A Jackson/Shutterstock 175, 226, Brainpencil/Shutterstock 181, Cristi180884/Shutterstock 125, Chrisdorney/Shutterstock 166, Crisp/Shutterstock 185, Cybrain/Shutterstock 267t, Cobalt88/Shutterstock 234, DavidTB/Shutterstock 24, Dja65/Shutterstock 160tc, D-VISIONS/Shutterstock 163, Dny3d/Shutterstock 234, ESB Professional/Shutterstock 85, Evgeny Karandaev/Shutterstock 211, Gaby Kooijman/Shutterstock 52b, Gorodenkoff/Shutterstock 172, GraphicINmotion/Shutterstock 268t, Hxdbzxy/Shutterstock 258, Jerry Zitterman/Shutterstock 37, Joshua Davenport/Shutterstock 7, Kavione/Shutterstock 160tr, Luisa Leal Photography/Shutterstock 160bl, Maksym Kapliuk/Shutterstock 25, Michaeljung/Shutterstock 65, Mmaxer/Shutterstock 219, Moomchak V. Design/Shutterstock 268b, Nito/Shutterstock 17, Nikita G. Bernadsky/Shutterstock 266b, Nobeastsofierce/Shutterstock 267b, OlegDoroshin/Shutterstock 5, vii, Platslee/Shutterstock 129, 142, Pakhnyushchy/Shutterstock 160cl, Phonlamai Photo/Shutterstock 224, Paul Fleet/Shutterstock 266t, Silverkblackstock/Shutterstock 184, Shutterstock 93, Thinglass/Shutterstock 248, Trong Nguyen/Shutterstock 238, TY Lim/Shutterstock 4, vii, Vlad Kochelaevskiy/Shutterstock 177, 187, Volodymyr Krasyuk/Shutterstock 235, Wright Studio/Shutterstock 144, Xfilephotos/Shutterstock 237.

Endorsement statement
In order to ensure that this resource offers high-quality support for the associated Pearson qualification, it has been through a review process by the awarding body. This process confirmed that this resource fully covers the teaching and learning content of the specification at which it is aimed. It also confirms that it demonstrates an appropriate balance between the development of subject skills, knowledge and understanding, in addition to preparation for assessment.

Endorsement does not cover any guidance on assessment activities or processes (e.g. practice questions or advice on how to answer assessment questions) included in the resource, nor does it prescribe any particular approach to the teaching or delivery of a related course.

While the publishers have made every attempt to ensure that advice on the qualification and its assessment is accurate, the official specification and associated assessment guidance materials are the only authoritative source of information and should always be referred to for definitive guidance.

Pearson examiners have not contributed to any sections in this resource relevant to examination papers for which they have responsibility.

Examiners will not use endorsed resources as a source of material for any assessment set by Pearson. Endorsement of a resource does not mean that the resource is required to achieve this Pearson qualification, nor does it mean that it is the only suitable material available to support the qualification, and any resource lists produced by the awarding body shall include this and other appropriate resources.

ABOUT THIS BOOK

This book is written for students following the Pearson Edexcel International GCSE (9–1) Computer Science specification and covers both years of the course.

The course has been structured so that teaching and learning can take place in any order, both in the classroom and in any independent learning. The book contains six units that match the six areas of content in the specification: Problem Solving, Programming, Data, Computers, Communication and the Internet, The Bigger Picture.

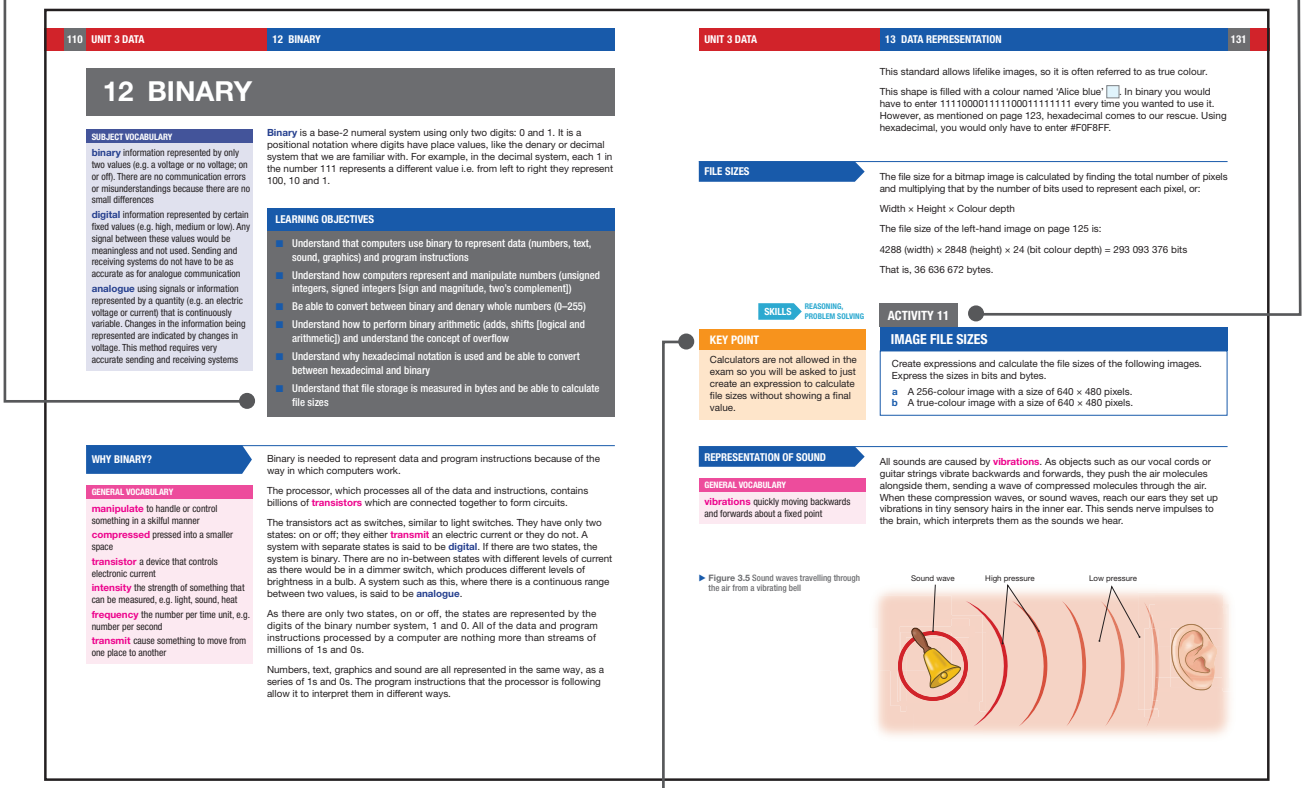
Each unit is split into multiple sections to break down content into manageable chunks and to ensure full coverage of the specification.

Each unit features a mix of learning and activities. Summary questions at the end of each chapter help you to put learning into practice and prepare for the exam.

Paper 1 is Principles of Computer Science and Paper 2 is Application of Computational Thinking. Knowing how to apply your learning to both of these will be critical for your success in the exam. There is a real applied focus to the book. You will be encouraged to put the theory you are learning into context and apply what you have learned to your own practical activities.

Learning objectives
Each section starts with a list of what you will learn from it. They are carefully tailored to address key assessment objectives central to the course.

Activity
Each chapter includes activities to embed understanding through practical tasks and questions.



110 UNIT 3 DATA 12 BINARY

12 BINARY

SUBJECT VOCABULARY

binary information represented by only two values (e.g. a voltage or no voltage; on or off). There are no communication errors or misunderstandings because there are no small differences

digital information represented by certain fixed values (e.g. high, medium or low). Any signal between these values would be meaningless and not used. Sending and receiving systems do not have to be as accurate as for analogue communication

analogue using signals or information represented by a quantity (e.g. an electric voltage or current) that is continuously variable. Changes in the information being represented are indicated by changes in voltage. This method requires very accurate sending and receiving systems

Binary is a base-2 numeral system using only two digits: 0 and 1. It is a positional notation where digits have place values, like the denary or decimal system that we are familiar with. For example, in the decimal system, each 1 in the number 111 represents a different value i.e. from left to right they represent 100, 10 and 1.

LEARNING OBJECTIVES

- Understand that computers use binary to represent data (numbers, text, sound, graphics) and program instructions
- Understand how computers represent and manipulate numbers (unsigned integers, signed integers [sign and magnitude, two's complement])
- Be able to convert between binary and denary whole numbers (0–255)
- Understand how to perform binary arithmetic (adds, shifts [logical and arithmetic]) and understand the concept of overflow
- Understand why hexadecimal notation is used and be able to convert between hexadecimal and binary
- Understand that file storage is measured in bytes and be able to calculate file sizes

WHY BINARY?

Binary is needed to represent data and program instructions because of the way in which computers work.

GENERAL VOCABULARY

manipulate to handle or control something in a skilful manner

compressed pressed into a smaller space

transistor a device that controls electronic current

intensity the strength of something that can be measured, e.g. light, sound, heat

frequency the number per time unit, e.g. number per second

transmit cause something to move from one place to another

The processor, which processes all of the data and instructions, contains billions of **transistors** which are connected together to form circuits.

The transistors act as switches, similar to light switches. They have only two states: on or off; they either **transmit** an electric current or they do not. A system with separate states is said to be **digital**. If there are two states, the system is binary. There are no in-between states with different levels of current as there would be in a dimmer switch, which produces different levels of brightness in a bulb. A system such as this, where there is a continuous range between two values, is said to be **analogue**.

As there are only two states, on or off, the states are represented by the digits of the binary number system, 1 and 0. All of the data and program instructions processed by a computer are nothing more than streams of millions of 1s and 0s.

Numbers, text, graphics and sound are all represented in the same way, as a series of 1s and 0s. The program instructions that the processor is following allow it to interpret them in different ways.

UNIT 3 DATA 13 DATA REPRESENTATION 131

This standard allows lifelike images, so it is often referred to as true colour.

This shape is filled with a colour named 'Alice blue'. In binary you would have to enter 11110000111100011111111 every time you wanted to use it. However, as mentioned on page 123, hexadecimal comes to our rescue. Using hexadecimal, you would only have to enter #F0F8FF.

FILE SIZES

The file size for a bitmap image is calculated by finding the total number of pixels and multiplying that by the number of bits used to represent each pixel, or:

Width × Height × Colour depth

The file size of the left-hand image on page 125 is:

4288 (width) × 2848 (height) × 24 (bit colour depth) = 293 093 376 bits

That is, 36 636 672 bytes.

SKILLS REASONING, PROBLEM SOLVING

ACTIVITY 11

IMAGE FILE SIZES

Create expressions and calculate the file sizes of the following images. Express the sizes in bits and bytes.

a A 256-colour image with a size of 640 × 480 pixels.
b A true-colour image with a size of 640 × 480 pixels.

KEY POINT

Calculators are not allowed in the exam so you will be asked to just create an expression to calculate file sizes without showing a final value.

REPRESENTATION OF SOUND

GENERAL VOCABULARY

vibrations quickly moving backwards and forwards about a fixed point

All sounds are caused by **vibrations**. As objects such as our vocal cords or guitar strings vibrate backwards and forwards, they push the air molecules alongside them, sending a wave of compressed molecules through the air. When these compression waves, or sound waves, reach our ears they set up vibrations in tiny sensory hairs in the inner ear. This sends nerve impulses to the brain, which interprets them as the sounds we hear.

▶ Figure 3.5 Sound waves travelling through the air from a vibrating bell

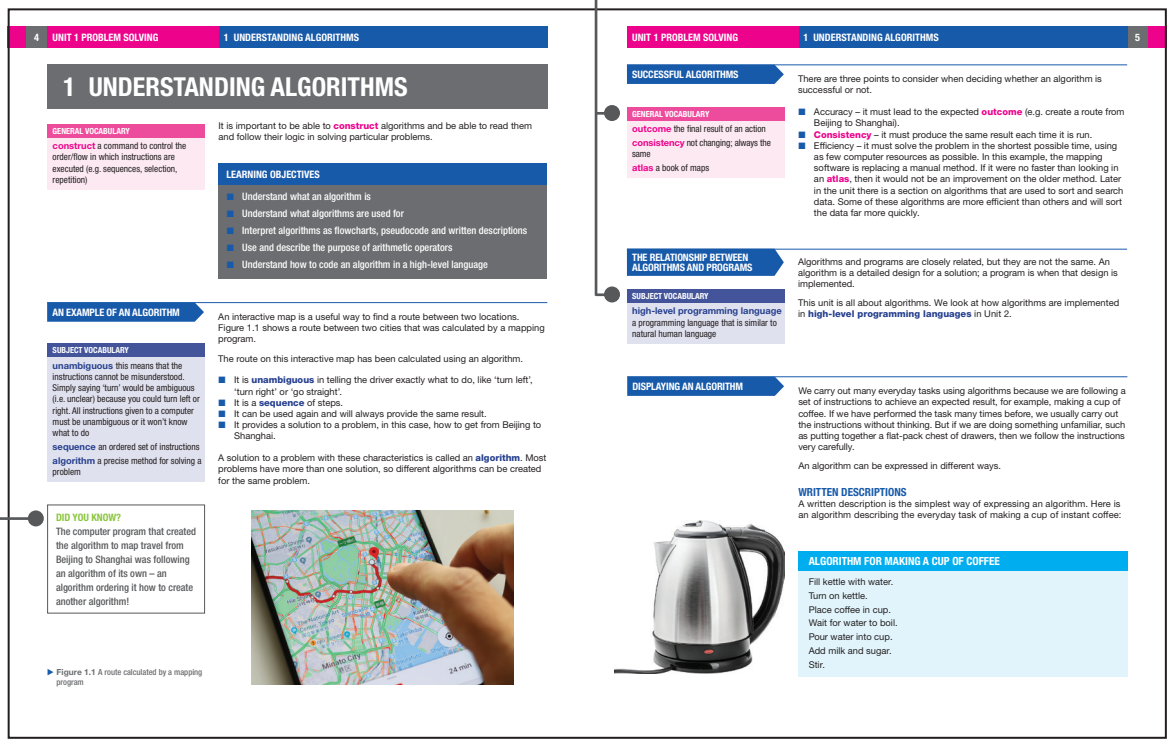
Extend your knowledge
Push yourself further by looking beyond the content of the course.

Key point
Easy to understand, core information to take away from sections.

Worked example
Key concepts can be demonstrated using step-by-step walkthroughs.

Did you know?
Interesting facts to encourage wider thought and stimulate discussion.

Subject vocabulary and General vocabulary
Useful words and phrases are colour coded within the main text and picked out in the margin with concise and simple definitions. These help understanding of key subject terms and support students whose first language is not English.



4 UNIT 1 PROBLEM SOLVING 1 UNDERSTANDING ALGORITHMS

1 UNDERSTANDING ALGORITHMS

GENERAL VOCABULARY
construct a command to control the order/flow in which instructions are executed (e.g. sequences, selection, repetition)

It is important to be able to **construct** algorithms and be able to read them and follow their logic in solving particular problems.

LEARNING OBJECTIVES
- Understand what an algorithm is
- Understand what algorithms are used for
- Interpret algorithms as flowcharts, pseudocode and written descriptions
- Use and describe the purpose of arithmetic operators
- Understand how to code an algorithm in a high-level language

AN EXAMPLE OF AN ALGORITHM

SUBJECT VOCABULARY
unambiguous this means that the instructions cannot be misunderstood. Simply saying 'turn' would be ambiguous (i.e. unclear) because you could turn left or right. All instructions given to a computer must be unambiguous or it won't know what to do
sequence an ordered set of instructions
algorithm a precise method for solving a problem

An interactive map is a useful way to find a route between two locations. Figure 1.1 shows a route between two cities that was calculated by a mapping program.

The route on this interactive map has been calculated using an algorithm.

- It is **unambiguous** in telling the driver exactly what to do, like 'turn left', 'turn right' or 'go straight'.
- It is a **sequence** of steps.
- It can be used again and will always provide the same result.
- It provides a solution to a problem, in this case, how to get from Beijing to Shanghai.

A solution to a problem with these characteristics is called an **algorithm**. Most problems have more than one solution, so different algorithms can be created for the same problem.

DID YOU KNOW?
The computer program that created the algorithm to map travel from Beijing to Shanghai was following an algorithm of its own – an algorithm ordering it how to create another algorithm!

▶ **Figure 1.1** A route calculated by a mapping program

UNIT 1 PROBLEM SOLVING 1 UNDERSTANDING ALGORITHMS 5

SUCCESSFUL ALGORITHMS

There are three points to consider when deciding whether an algorithm is successful or not.

GENERAL VOCABULARY
outcome the final result of an action
consistency not changing; always the same
atlas a book of maps

- Accuracy – it must lead to the expected **outcome** (e.g. create a route from Beijing to Shanghai).
- **Consistency** – it must produce the same result each time it is run.
- Efficiency – it must solve the problem in the shortest possible time, using as few computer resources as possible. In this example, the mapping software is replacing a manual method. If it were no faster than looking in an **atlas**, then it would not be an improvement on the older method. Later in the unit there is a section on algorithms that are used to sort and search data. Some of these algorithms are more efficient than others and will sort the data far more quickly.

THE RELATIONSHIP BETWEEN ALGORITHMS AND PROGRAMS

Algorithms and programs are closely related, but they are not the same. An algorithm is a detailed design for a solution; a program is when that design is implemented.

SUBJECT VOCABULARY
high-level programming language a programming language that is similar to natural human language

This unit is all about algorithms. We look at how algorithms are implemented in **high-level programming languages** in Unit 2.

DISPLAYING AN ALGORITHM

We carry out many everyday tasks using algorithms because we are following a set of instructions to achieve an expected result, for example, making a cup of coffee. If we have performed the task many times before, we usually carry out the instructions without thinking. But if we are doing something unfamiliar, such as putting together a flat-pack chest of drawers, then we follow the instructions very carefully.

An algorithm can be expressed in different ways.

WRITTEN DESCRIPTIONS

A written description is the simplest way of expressing an algorithm. Here is an algorithm describing the everyday task of making a cup of instant coffee:

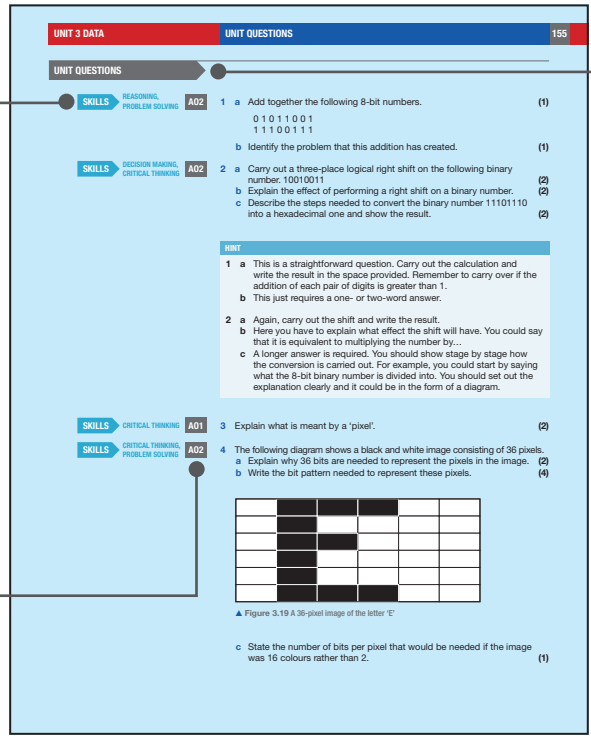
ALGORITHM FOR MAKING A CUP OF COFFEE

Fill kettle with water.
Turn on kettle.
Place coffee in cup.
Wait for water to boil.
Pour water into cup.
Add milk and sugar.
Stir.

Summary
Quickly recap the core content of each section.

Skills
Relevant exam questions have been assigned the key skills that you will gain from undertaking them, allowing for a strong focus on particular academic qualities. These transferable skills are highly valued in further study and the workplace.

Assessment objectives
Questions are tagged with the relevant assessment objectives that are being examined.



UNIT 3 DATA UNIT QUESTIONS 155

UNIT QUESTIONS

SKILLS REASONING, PROBLEM SOLVING AO2
1 a Add together the following 8-bit numbers. (1)

01011001
11100111

b Identify the problem that this addition has created. (1)

SKILLS DECISION MAKING, CRITICAL THINKING AO2
2 a Carry out a three-place logical right shift on the following binary number. 10010011 (2)
b Explain the effect of performing a right shift on a binary number. (2)
c Describe the steps needed to convert the binary number 11101110 into a hexadecimal one and show the result. (2)

HINT
1 a This is a straightforward question. Carry out the calculation and write the result in the space provided. Remember to carry over if the addition of each pair of digits is greater than 1.
b This just requires a one- or two-word answer.
2 a Again, carry out the shift and write the result.
b Here you have to explain what effect the shift will have. You could say that it is equivalent to multiplying the number by...
c A longer answer is required. You should show stage by stage how the conversion is carried out. For example, you could start by saying what the 8-bit binary number is divided into. You should set out the explanation clearly and it could be in the form of a diagram.

SKILLS CRITICAL THINKING AO1
3 Explain what is meant by a 'pixel'. (2)

SKILLS CRITICAL THINKING, PROBLEM SOLVING AO2
4 The following diagram shows a black and white image consisting of 36 pixels.
a Explain why 36 bits are needed to represent the pixels in the image. (2)
b Write the bit pattern needed to represent these pixels. (4)

▲ **Figure 3.19** A 36-pixel image of the letter 'E'

c State the number of bits per pixel that would be needed if the image was 16 colours rather than 2. (1)

Checkpoint
Checkpoints help you to check and reflect on your learning at the end of each section. Strengthen questions help you to consolidate basic knowledge and understanding. Challenge questions are more demanding and ask you to apply your learning.

Unit questions
These exam-style questions are found at the end of each unit. They are tailored to the Pearson Edexcel specification to allow for the practice and development of exam writing technique. They also allow for practice responding to the command words used in the exams.

ASSESSMENT OVERVIEW

The following tables give an overview of the assessment for this course. You should study this information closely to help ensure that you are fully prepared and know exactly what to expect in each part of the assessment.

PAPER 1	PERCENTAGE	MARK	TIME	AVAILABILITY
PRINCIPLES OF COMPUTER SCIENCE Written exam paper Paper code 4CP0/01 Externally set and assessed by Pearson Edexcel Single tier of entry	50%	80	2 hours	June exam series First assessment June 2019

PAPER 2	PERCENTAGE	MARK	TIME	AVAILABILITY
APPLICATION OF COMPUTATIONAL THINKING Practical and written exam paper Paper code 4CP0/02 Externally set and assessed by Pearson Edexcel Single tier of entry	50%	80	3 hours	June exam series First assessment June 2019

ASSESSMENT OBJECTIVES AND WEIGHTINGS

ASSESSMENT OBJECTIVE	DESCRIPTION	% IN INTERNATIONAL GCSE
AO1	Demonstrate knowledge and understanding of the key principles of computer science	27.5
AO2	Apply knowledge and understanding of key concepts and principles of computer science	42.5
AO3	Analyse problems in computational terms: • to make reasoned judgements • to design, program, test, evaluate and refine solutions	30

RELATIONSHIP OF ASSESSMENT OBJECTIVES TO UNITS

UNIT NUMBER	ASSESSMENT OBJECTIVE		
	AO1	AO2	AO3
PAPER 1	21.5%	21%	7.5%
PAPER 2	6%	21.5%	22.5%
TOTAL FOR INTERNATIONAL GCSE	**27.5%**	**42.5%**	**30%**

ASSESSMENT SUMMARY

PAPER 1	DESCRIPTION	MARKS	ASSESSMENT OBJECTIVES
PRINCIPLES OF COMPUTER SCIENCE PAPER CODE 4CP0/01	**Structure** Paper 1 contributes 50% of the total marks for the Computer Science qualification. Students must answer all questions. The paper consists of multiple-choice, short open-response, open-response and extended open-response answer questions.	The total number of marks available is 80	Questions will test the following Assessment Objectives: AO1 – 21.5% AO2 – 21% AO3 – 7.5%
	Content summary This paper will primarily assess knowledge and understanding of the basic principles of computer science, including some coverage of how these principles are applied when solving problems that relate to a particular situation.		
	Assessment This is a single-tier exam paper and all questions cover the full ranges of grades from 9–1. The assessment duration is 2 hours.		
PAPER 2	**DESCRIPTION**	**MARKS**	**ASSESSMENT OBJECTIVES**
APPLICATION OF COMPUTATIONAL THINKING PAPER CODE 4CP0/02	**Structure** Paper 2 contributes 50% of the total marks for the Computer Science qualification. Students must answer all questions. The paper consists of multiple-choice, short open-response, open-response, extended open-response answer and task-based questions. The task-based questions will be carried out using a computer system under supervision. All other questions requiring a written response will be answered in the paper.	The total number of marks available is 80	Questions will test the following Assessment Objectives: AO1 – 6% AO2 – 21.5% AO3 – 22.5%
	Content summary This paper will primarily assess the practical application of computational thinking, whereby learners will create, use and adapt existing algorithms to solve problems in a particular situation. This paper will also test students' knowledge and understanding of the topics.		
	Assessment This is a single-tier exam paper and all questions cover the full ranges of grades from 9–1. The assessment duration is 3 hours. A choice of three programming languages will be available (Python, C# or Java). A pseudocode reference document will be available for learners to reference during the assessment.		

UNIT 1 PROBLEM SOLVING

Assessment Objective 1

Demonstrate knowledge and understanding of the key principles of computer science

Assessment Objective 2

Apply knowledge and understanding of key concepts and principles of computer science

Assessment Objective 3

Analyse problems in computational terms:
- to make reasoned judgements
- to design, program, test, evaluate and refine solutions

In this unit you will learn about algorithms, which are the basis of computer programming, and how they can be presented as flowcharts and pseudocode. You will learn about the basic constructs of an algorithm such as sequence, selection and iteration and how these are used to solve problems using computational thinking. You will also look at completing and correcting algorithms in addition to algorithms to sort and search data. In the next unit you will learn how to code these algorithms using pseudocode and high-level programming languages.

1 UNDERSTANDING ALGORITHMS

GENERAL VOCABULARY

construct a command to control the order/flow in which instructions are executed (e.g. sequences, selection, repetition)

It is important to be able to **construct** algorithms and be able to read them and follow their logic in solving particular problems.

LEARNING OBJECTIVES

- Understand what an algorithm is
- Understand what algorithms are used for
- Interpret algorithms as flowcharts, pseudocode and written descriptions
- Use and describe the purpose of arithmetic operators
- Understand how to code an algorithm in a high-level language

AN EXAMPLE OF AN ALGORITHM

SUBJECT VOCABULARY

unambiguous this means that the instructions cannot be misunderstood. Simply saying 'turn' would be ambiguous (i.e. unclear) because you could turn left or right. All instructions given to a computer must be unambiguous or it won't know what to do

sequence an ordered set of instructions

algorithm a precise method for solving a problem

An interactive map is a useful way to find a route between two locations. Figure 1.1 shows a route between two cities that was calculated by a mapping program.

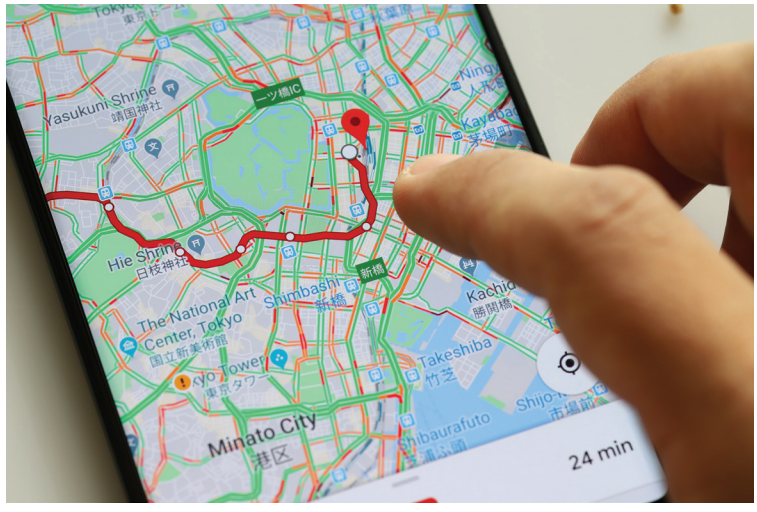
The route on this interactive map has been calculated using an algorithm.

- It is **unambiguous** in telling the driver exactly what to do, like 'turn left', 'turn right' or 'go straight'.
- It is a **sequence** of steps.
- It can be used again and will always provide the same result.
- It provides a solution to a problem, in this case, how to get from Beijing to Shanghai.

A solution to a problem with these characteristics is called an **algorithm**. Most problems have more than one solution, so different algorithms can be created for the same problem.

DID YOU KNOW?

The computer program that created the algorithm to map travel from Beijing to Shanghai was following an algorithm of its own – an algorithm ordering it how to create another algorithm!



▶ **Figure 1.1** A route calculated by a mapping program

SUCCESSFUL ALGORITHMS

There are three points to consider when deciding whether an algorithm is successful or not.

GENERAL VOCABULARY

outcome the final result of an action

consistency not changing; always the same

atlas a book of maps

- Accuracy – it must lead to the expected **outcome** (e.g. create a route from Beijing to Shanghai).
- **Consistency** – it must produce the same result each time it is run.
- Efficiency – it must solve the problem in the shortest possible time, using as few computer resources as possible. In this example, the mapping software is replacing a manual method. If it were no faster than looking in an **atlas**, then it would not be an improvement on the older method. Later in the unit there is a section on algorithms that are used to sort and search data. Some of these algorithms are more efficient than others and will sort the data far more quickly.

THE RELATIONSHIP BETWEEN ALGORITHMS AND PROGRAMS

Algorithms and programs are closely related, but they are not the same. An algorithm is a detailed design for a solution; a program is when that design is implemented.

SUBJECT VOCABULARY

high-level programming language a programming language that is similar to natural human language

This unit is all about algorithms. We look at how algorithms are implemented in **high-level programming languages** in Unit 2.

DISPLAYING AN ALGORITHM

We carry out many everyday tasks using algorithms because we are following a set of instructions to achieve an expected result, for example, making a cup of coffee. If we have performed the task many times before, we usually carry out the instructions without thinking. But if we are doing something unfamiliar, such as putting together a flat-pack chest of drawers, then we follow the instructions very carefully.

An algorithm can be expressed in different ways.

WRITTEN DESCRIPTIONS

A written description is the simplest way of expressing an algorithm. Here is an algorithm describing the everyday task of making a cup of instant coffee:

ALGORITHM FOR MAKING A CUP OF COFFEE

Fill kettle with water.
Turn on kettle.
Place coffee in cup.
Wait for water to boil.
Pour water into cup.
Add milk and sugar.
Stir.

SKILLS REASONING, PROBLEM SOLVING

ACTIVITY 1

GETTING TO SCHOOL

Produce a written description of an algorithm for getting to school. It should start with leaving home and end with arriving at school. For example, the algorithm could start with 'Walk to bus stop'.

Check your algorithm with other members of the group. Would your algorithm work for others? Are there any general statements that are common to all algorithms?

SUBJECT VOCABULARY

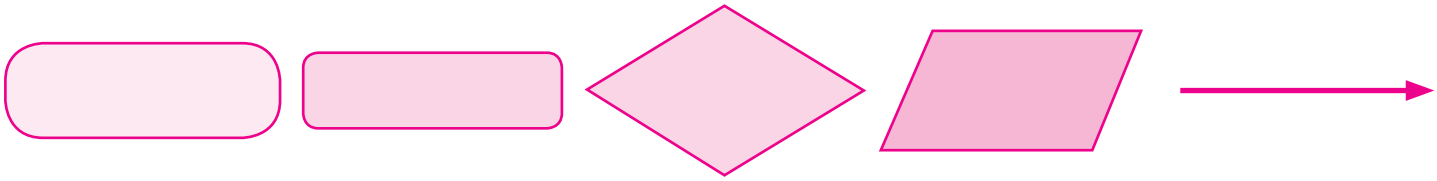
flowchart shows an algorithm as a diagram. Each step in the algorithm is represented by a symbol. Symbols are linked together with arrows showing the order in which steps are completed

FLOWCHARTS

Flowcharts can be used to show an algorithm as a diagram. They provide a more visual display.

There are special symbols that have to be used in a flowchart. You can't just make up your own, because nobody else would be able to follow your algorithm.

Figure 1.2 shows the flowchart symbols that should be used.



Indicates the start or end of an algorithm

Indicates a process to be carried out

Indicates a decision to be made

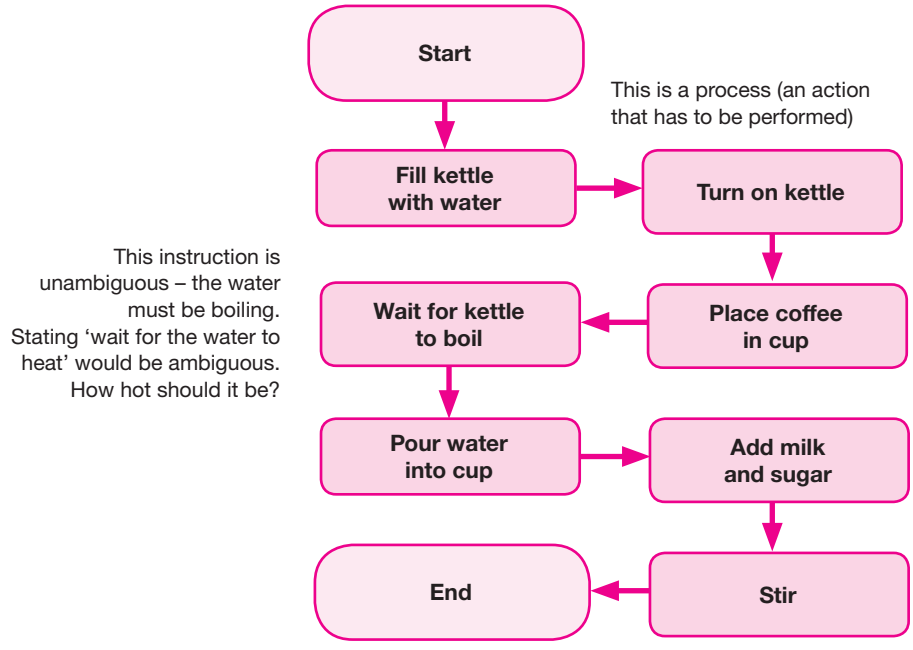
Indicates an input or output

Shows the logical flow of the algorithm

▲ **Figure 1.2** Flowchart symbols

The flowchart in Figure 1.3 is an alternative way of showing the algorithm for making a cup of coffee as a written description.

▶ **Figure 1.3** Flowchart of an algorithm to make a cup of coffee



SKILLS PROBLEM SOLVING

ACTIVITY 2

SCHOOL JOURNEY FLOWCHART

Display the 'journey to school' algorithm that you created in Activity 1 as a flowchart.

SKILLS PROBLEM SOLVING

ACTIVITY 3

BATH FLOWCHART

A student has created a written algorithm for preparing a bath. Working with a partner, display the following as a flowchart. You may need to change the order or add actions.

- Put in the plug.
- Fill the bath to the correct level.
- Check the temperature is OK.

GENERAL VOCABULARY

basis an important idea or fact that something is based on

SUBJECT VOCABULARY

pseudocode a structured, code-like language that can be used to describe an algorithm

developer a person whose job it is to create new software

logic the principles and reasoning underlying the constructs and elements to be applied in solving problems

The algorithms you have looked at so far are designed for humans to follow. Algorithms also form the **basis** of computer programs. Computers are mindless machines that simply do exactly what they are told. They follow a set of instructions, but they can carry out these instructions far more quickly than humans. That is why they are so useful.

PSEUDOCODE

In addition to flowcharts and written descriptions, algorithms can also be expressed in **pseudocode**. The pseudocode can be used to code the solution in an actual programming language.

It allows the **developer** to concentrate on the **logic** and efficiency of the algorithm without having to bother about the rules of any particular programming language. It is relatively straightforward to translate an algorithm written in pseudocode into any high-level programming language.

SKILLS RESEARCH

ACTIVITY 4

INVESTIGATING PSEUDOCODE

Different organisations or examination boards have their own unique versions of pseudocode. Investigate the Pearson Edexcel pseudocode that you will need for your International GCSE course and which will be used in this book.

EXAMPLE OF A SIMPLE ALGORITHM

To introduce the Pearson Edexcel pseudocode, here is a simple written algorithm that asks the user to input two numbers and then outputs the result of adding them together.

WRITTEN DESCRIPTION

ALGORITHM FOR ADDING TWO NUMBERS

Enter first number.
Enter second number.
Calculate total by adding first and second numbers.
Output total.

FLOWCHART

▲ **Figure 1.4** Flowchart showing the adding of two numbers

SUBJECT VOCABULARY

variable a 'container' used to store data. The data stored in a variable is referred to as a value. The value stored in a variable is not fixed. The same variable can store different values during the course of a program and each time a program is run

identifier a unique name given to a variable or a constant. Using **descriptive** names for variables makes code much easier to read

arithmetic operator an **operator** that performs a calculation on two numbers

GENERAL VOCABULARY

operator a character that represents an action, e.g. 'x' represents a multiplication and '/' a division

descriptive describing something clearly

PSEUDOCODE

ALGORITHM FOR ADDING TWO NUMBERS

```
SEND 'Please enter the first number' TO DISPLAY
RECEIVE firstNumber FROM KEYBOARD
SEND 'Please enter the second number' TO DISPLAY
RECEIVE secondNumber FROM KEYBOARD
SET total TO firstNumber + secondNumber
SEND total TO DISPLAY
```

The pseudocode gives clear step-by-step instructions that the computer will be expected to carry out. It also introduces some important programming concepts.

- The numbers entered by the user are stored in two **variables** with the **identifiers** `firstNumber` and `secondNumber`.
- The result of adding the numbers together is stored in the variable `total`.
- Text has to be placed in quotation marks (single or double) if it is to be displayed. For example, 'Please enter the first number' (or "Please enter the first number").
- Quotation marks are not used if a variable is to be displayed. If they were, the word 'total' in the last instruction would be displayed instead of the number it represents.
- **Arithmetic operators** are used to perform calculations. Table 1.1 shows the arithmetic operators.

ARITHMETIC OPERATORS

OPERATOR	FUNCTION	EXAMPLE
+	Addition: add the values together.	8 + 5 = 13 myScore1 + myScore2
–	Subtraction: subtract the second value from the first.	17 – 4 = 13 myScore1 – myScore2
*	Multiplication: multiply the values together.	6 * 9 = 54 numberBought * price
/	Real division: divide the first value by the second value and return the result including decimal places.	13 / 4 = 3.25 totalMarks/numberTests
DIV	**Quotient**: like division, but it only returns the whole number or *integer*.	13 DIV 4 = 3 totalMarks DIV numberTests
MOD	**Modulus**/modulo: this will return the remainder of a division.	13 / 4 = 3 remainder 1 Therefore 13 MOD4 = 1
^	Exponentiation: this is for 'to the **power** of'.	3 ^ 3 = 27 It is the same as writing 3^3

▲ **Table 1.1** Arithmetic operators

VARIABLES AND CONSTANTS

KEY POINT

When you chose a variable name, it should be descriptive of the data it will hold, e.g. distance or length. Long variable names are easier to misspell!

SUBJECT VOCABULARY

constant a 'container' that holds a value that never changes; like variables, constants have unique identifiers

GENERAL VOCABULARY

quotient a result found by dividing one quantity by another

modulus the remainder after the division of one number by another

power the small number written to the right and above another number to show how many times it should be multiplied by itself

Variables play an important role in algorithms and programming. The value stored by a variable can change as a program is running. Variables are extremely useful in programming because they make it possible for the same program to process different sets of data.

A **constant** is the opposite of a variable. It is a 'container' that holds a value that always stays the same. Constants are useful for storing fixed information, such as the value of pi, the number of litres in a gallon or the number of months in a year.

Each variable and constant in an algorithm has to have a unique identifier. It is important to choose descriptive names for identifiers. This will make your code much easier to read. For example, a variable to hold a user's first name could be given the identifier `firstName` to show the data it contains. If it were given the identifier `X` instead, it wouldn't clearly show what data it contained.

NAMING CONVENTIONS FOR VARIABLES AND CONSTANTS

It is sensible to write identifiers in the same way throughout an algorithm. A common method is to use 'camel case' for compound words (e.g. `firstName`, `secondName`) with no space between words and the second word starting with a capital letter. Alternatively, you could capitalise the first letter of both words, e.g. `FirstName, SecondName`, or separate the words with an underscore, e.g. `first_name, second_name`, known as 'snake case'.

SKILLS PROBLEM SOLVING, ANALYSIS

ACTIVITY 5

WRITING ALGORITHMS IN PSEUDOCODE

Here is a written description of an algorithm:

Enter the first number.
Enter the second number.
The third number is equal to the first number multiplied by the second number.
Display the third number.

Express this algorithm in pseudocode.

KEY POINT

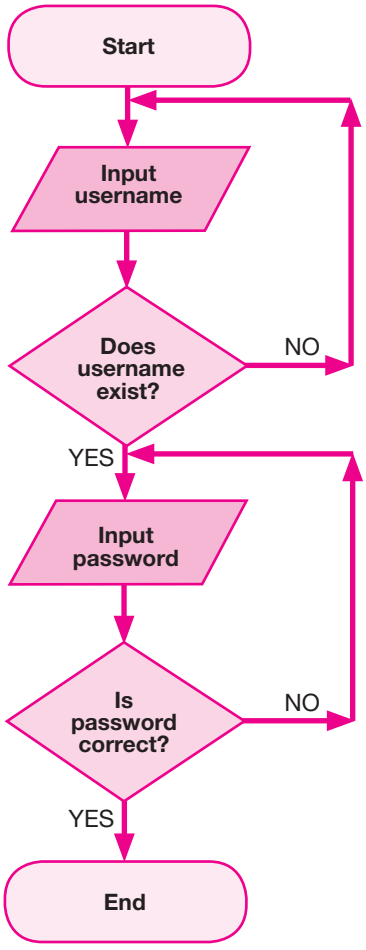
Whichever method you use for naming conventions, you must use it consistently and not keep switching between the different methods.

SKILLS PROBLEM SOLVING

ACTIVITY 6

WRITTEN DESCRIPTIONS OF ALGORITHMS

This algorithm is displayed as a flowchart.



▲ **Figure 1.5** Flowchart of an algorithm

Produce a written description of this algorithm.

EXTEND YOUR KNOWLEDGE

When you enter a search term into Google®, a list of links to websites is returned. But why are they presented in that particular order? With a partner, research the `PageRank` algorithm that Google uses to rate the importance of websites and write a short report about your findings.

CHECKPOINT

Strengthen

SKILLS CRITICAL THINKING

S1 Produce a written description of an algorithm for borrowing a book from the library.

SKILLS DECISION MAKING

S2 What is the function of each of the seven arithmetic operators?

SKILLS REASONING

S3 What is a variable? Why are they useful?

SKILLS CRITICAL THINKING

S4 What is the difference between a variable and a constant?

Challenge

SKILLS CRITICAL THINKING, PROBLEM SOLVING

C1 Produce a flowchart describing an algorithm for making a cheese sandwich.

SKILLS PROBLEM SOLVING

C2 Write an algorithm expressed in pseudocode that receives three numbers from the keyboard, then calculates and displays the average.

How confident do you feel about your answers to these questions? If you're not sure you answered them well, try the following activities again.

- For S1 re-read 'The relationship between algorithms and programs'.
- For S2 study Table 1.1.
- For S3 and S4 look again at 'Variables and constants'.

SUMMARY

- An algorithm is a precise method for solving a problem.
- Algorithms can be displayed as written descriptions, flowcharts and in pseudocode.
- Pseudocode is a **structured**, code-like language.
- Pseudocode is translated into program code.
- Arithmetic operators are used in calculations.
- Variables and constants are 'containers' for storing data. The value stored in a variable can change, whereas the value of a constant never changes.
- Selecting descriptive names for identifiers makes code easier to read.

GENERAL VOCABULARY

structured organised so that the parts work well together