MY REVISION NOTES

HODDER EDUCATION

AQA

GCSE (9–1)

COMPUTER SCIENCE

3RD EDITION

- Plan and organise your revision
- Reinforce skills and understanding
- Practise exam-style questions

For the 8525 specification

George Rouse
Lorne Pearcey
Gavin Craddock

Boost

HODDER EDUCATION
LEARN MORE

MY REVISION NOTES
AQA GCSE (9–1)
COMPUTER SCIENCE

The Publishers would like to thank the following for permission to reproduce copyright material.

Photo credits
Figure 4.9: Background photograph © Mike Berenson/Colorado Captures/Getty Images

Acknowledgements

Hachette UK's policy is to use papers that are natural, renewable and recyclable products and made from wood grown in well-managed forests and other controlled sources. The logging and manufacturing processes are expected to conform to the environmental regulations of the country of origin.

Orders: please contact Hachette UK Distribution, Hely Hutchinson Centre, Milton Road, Didcot, Oxfordshire, OX11 7HH. Telephone: +44 (0)1235 827827. Email education@hachette.co.uk Lines are open from 9 a.m. to 5 p.m., Monday to Friday. You can also order through our website: www.hoddereducation.co.uk

ISBN: 978 1 3983 2115 1

First published in 2021 by
Hodder Education,
An Hachette UK Company
Carmelite House
50 Victoria Embankment
London EC4Y 0DZ

www.hoddereducation.co.uk

Impression number 10 9 8 7 6 5 4 3 2 1

Year 2025 2024 2023 2022 2021

Cover photo © ToheyVector - stock.adobe.com

Illustrations by Aptara Inc. and Integra Software Services Pvt. Ltd

Typeset in India by Aptara, Inc.

Printed in India

A catalogue record for this title is available from the British Library.

Get the most from this book

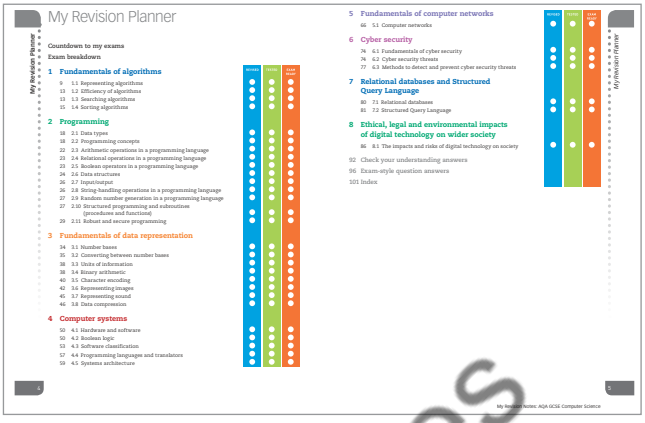
Everyone has to decide his or her own revision strategy, but it is essential to learn your work, review it and test your understanding. These Revision Notes will help you to do that in a planned way, topic by topic. Use this book as the cornerstone of your revision and don't hesitate to write in it – personalise your notes and check your progress by ticking off each section as you revise.

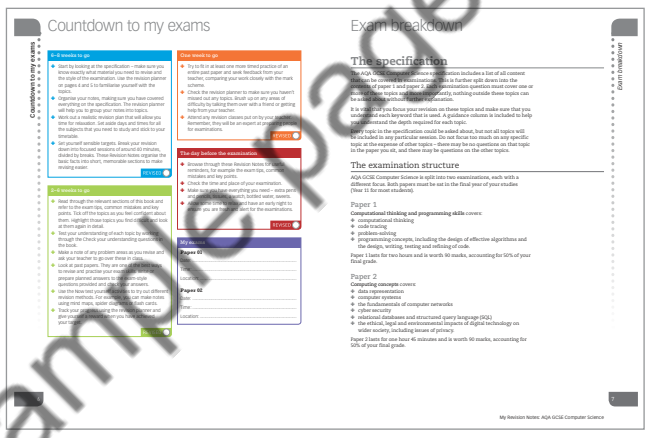
Tick to track your progress

Use the revision planner on pages 4 and 5 to plan your revision, topic by topic.
Tick each box when you have:

- revised and understood a topic
- tested yourself
- practised the exam questions and checked your answers.

You can also keep track of your revision by ticking off each topic heading in the book. You may find it helpful to add your own notes as you work through each topic.





Features to help you succeed

Exam tip

Expert tips to help polish your exam technique and maximise your chances in the exam.

Common mistake

Common mistakes that candidates make and how to avoid them.

Check your understanding

Questions to test your understanding of basic facts.

Definitions of **key terms** that need additional explanation are provided where they first appear.

Now test yourself

Activities to encourage note taking and revision.

Key point

Further explanation of some important issues.

Exam-style questions

Practice exam questions to consolidate your revision and practise your exam skills.

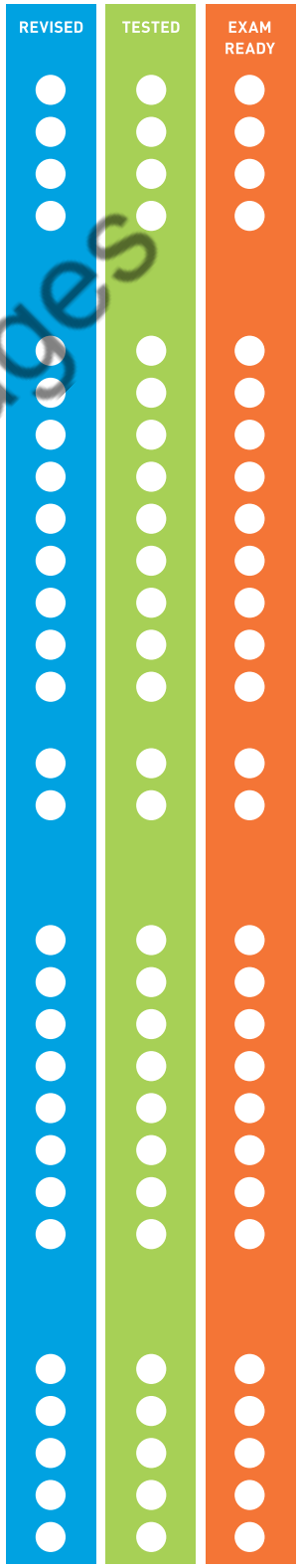
Worked example

Worked examples illustrate methods, calculations and explanations.

My Revision Planner

Countdown to my exams

Exam breakdown

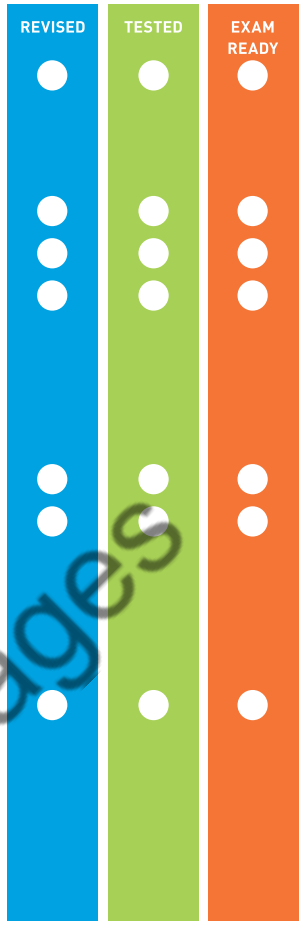


1 Fundamentals of algorithms

2 Programming

3 Fundamentals of data representation

4 Computer systems



Countdown to my exams

6–8 weeks to go

- Start by looking at the specification – make sure you know exactly what material you need to revise and the style of the examination. Use the revision planner on pages 4 and 5 to familiarise yourself with the topics.
- Organise your notes, making sure you have covered everything on the specification. The revision planner will help you to group your notes into topics.
- Work out a realistic revision plan that will allow you time for relaxation. Set aside days and times for all the subjects that you need to study and stick to your timetable.
- Set yourself sensible targets. Break your revision down into focused sessions of around 40 minutes, divided by breaks. These Revision Notes organise the basic facts into short, memorable sections to make revising easier.

REVISED

2–6 weeks to go

- Read through the relevant sections of this book and refer to the exam tips, common mistakes and key points. Tick off the topics as you feel confident about them. Highlight those topics you find difficult and look at them again in detail.
- Test your understanding of each topic by working through the Check your understanding questions in the book.
- Make a note of any problem areas as you revise and ask your teacher to go over these in class.
- Look at past papers. They are one of the best ways to revise and practise your exam skills. Write or prepare planned answers to the exam-style questions provided and check your answers.
- Use the Now test yourself activities to try out different revision methods. For example, you can make notes using mind maps, spider diagrams or flash cards.
- Track your progress using the revision planner and give yourself a reward when you have achieved your target.

REVISED

One week to go

- Try to fit in at least one more timed practice of an entire past paper and seek feedback from your teacher, comparing your work closely with the mark scheme.
- Check the revision planner to make sure you haven't missed out any topics. Brush up on any areas of difficulty by talking them over with a friend or getting help from your teacher.
- Attend any revision classes put on by your teacher. Remember, they will be an expert at preparing people for examinations.

REVISED

The day before the examination

- Browse through these Revision Notes for useful reminders, for example the exam tips, common mistakes and key points.
- Check the time and place of your examination.
- Make sure you have everything you need – extra pens and pencils, tissues, a watch, bottled water, sweets.
- Allow some time to relax and have an early night to ensure you are fresh and alert for the examinations.

REVISED

My exams

Paper 01

Date: .....

Time: .....

Location: .....

Paper 02

Date: .....

Time: .....

Location: .....

1 Fundamentals of algorithms

1.1 Representing algorithms

Computers are only able to follow instructions given by humans. A computer will only be able to solve a problem if it is given the right set of instructions. An **algorithm** is a step-by-step set of instructions to solve a problem. An algorithm is not a computer program but provides a blueprint for creating one.

Decomposition and abstraction

REVISED

Writing accurate algorithms that describe a solution to a problem is often quite complex. **Computational thinking** is an approach that uses three key principles that help us to write effective algorithms: **decomposition**, **abstraction** and **algorithmic thinking**:

Decomposition means breaking the complex problem down into smaller more manageable parts. Dealing with each small part of the problem is much simpler than trying to deal with a complex problem. Breaking the problem down allows us to work on each sub-problem individually, then combine those individual solutions into a solution for the whole problem. For very large problems, a team of programmers can work on individual parts of the problem.

Abstraction involves taking a real-life situation and creating a model of it so that it can be analysed. Removing or hiding unnecessary detail means the programmer can focus on the important aspects of the problem.

Algorithmic thinking is a method for defining the problem in a series of steps to be carried out in order to solve the problem. Often, algorithms can be the basis for solutions to similar problems. Looking for patterns can identify existing solutions to similar problems that can be adapted.

An algorithm must be precise enough to define a problem accurately and should include all the necessary steps in the process. It should be clear what instructions need to be followed and what decisions need to be made and when.

Check your understanding

1 Explain what is meant by *abstraction*.

2 Explain what is meant by *decomposition*.

Answers on p. 92

Algorithms in terms of their input, output and processing

REVISED

Input refers to the data given to the algorithm. This may be user input typed in at the keyboard, automatically collected by sensors or provided from a data file.

Output is the data given back by the algorithm to the user. This may be a message on screen, printed output, sounds, images or data stored to a data file.

Processing is how the data supplied is manipulated by the algorithm to provide the desired output.

Figure 1.1 Input–process–output

Structure diagrams

REVISED

A structure diagram is a visual representation of a top-down design that is used to decompose a problem. At each step the next level is developed by deciding what is required to complete that step.

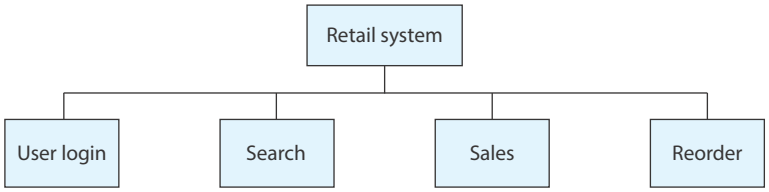


Figure 1.2 Two-level structure diagram

This structure can be expanded further by deciding what is required for the sales module.

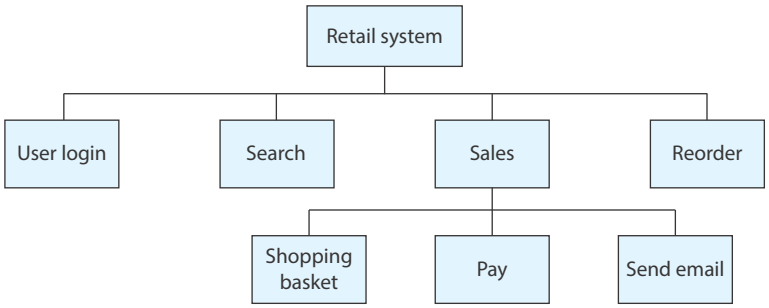


Figure 1.3 Three-level structure diagram

The process is repeated for all the steps at this level as necessary.

The process continues by adding levels below these where we need to clarify further what is required. This process carries on until each step is clearly defined.

A systematic approach to problem solving including flowcharts, pseudo-code and program code

REVISED

Flowcharts

The inputs, processes, outputs and decisions required for an algorithm can be represented graphically using a flowchart.

A flowchart may use any of the following symbols:

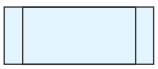
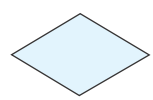
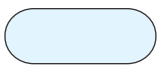
	Line	An arrow represents control passing between the connected shapes.
	Process	This shape represents something being performed or done.
	Subroutine	This shape represents a subroutine call that will relate to separate, non-linked flowchart.
	Input/ output	This shape represents the input or output of something into or out of the flowchart.
	Decision	This shape represents a decision (Yes/No or True/False) that results in two lines representing the different possible outcomes.
	Terminal	This shape represents the 'Start' and 'End' of the process.

Figure 1.4 Flowchart symbols

- All flowcharts start (and often end) with the terminal shape.
- Inputs and outputs are represented by a parallelogram.
- Decisions are represented by a rhombus shape with two outgoing paths, Yes/No or True/False.
- Loops are represented using one or more rhombus shapes.
- Processes are represented by rectangles.
- The symbol to call a **subroutine** is the rectangle with two vertical lines.

For the retail system we could define a reordering system for when stock is low using the following flowchart:

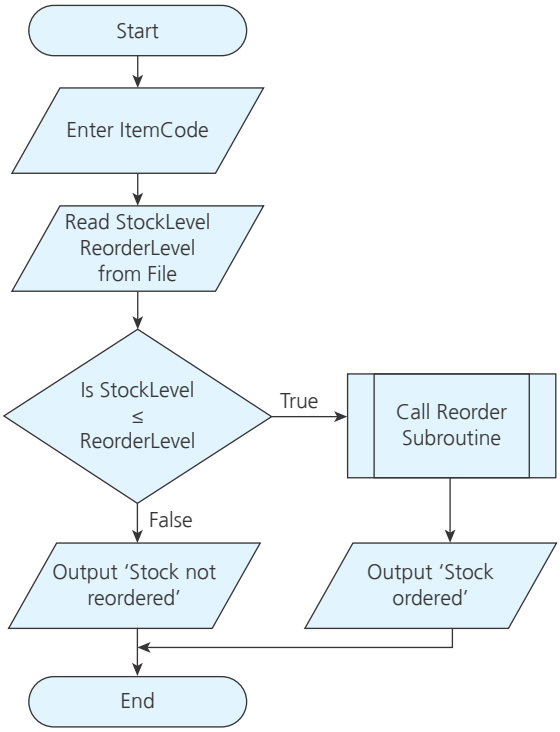


Figure 1.5 Flow diagram for retail reordering system

Subroutines are separate pieces of code that can be called from within a program. After the subroutine has completed its task, control returns to the main program.

Pseudo-code

Pseudo-code is an alternative way of presenting an algorithm using a structured form of English. The pseudo-code algorithm will show the structure of the solution without using the formal **syntax** of a high-level language.

The ordering system represented by the flowchart could be written in pseudo-code:

```
01 INPUT ItemCode
02 READ StockLevel, ReorderLevel From File
03 IF StockLevel <= ReorderLevel THEN
04   Call ReOrder()
05   PRINT 'Stock ordered'
06 ELSE
07   PRINT 'Stock not reordered'
08 END IF
```

Syntax refers to the formal structure used when writing a program. Computers only understand commands if they are typed precisely. The syntax includes the spelling of key words and the punctuation and grammar required.

Exam tip

Pseudo-code is not strictly defined and variations on the exact words used are often acceptable. AQA has a specific pseudo-code it will use in the exam; it is a good idea to become familiar with this. The guide is available from the AQA website.

Program code

Once the design process is complete the next step is to write the program in a suitable programming language. Using flowcharts and/or pseudo-code to prepare a working solution should improve the chances of the coded solution working effectively.

Determine the purpose of simple algorithms using trace tables and visual inspection

REVISED

A trace table is a method for following through an algorithm to identify any errors.

The contents of each variable are shown after each line of the algorithm has been completed.

By manually tracing the algorithm for a number of values we can identify if it works as expected or where any errors occur.

For the reorder system above, we can test the algorithm for different items and stock levels.

In this example the trace table has been completed for item 009, with a current stock level of 8 and a reorder level of 3.

Line	itemCode	StockLevel	ReorderLevel	Output	Comments
`01: INPUT ItemCode`	009				Item code 009 entered by user
`02: READ StockLevel, ReorderLevel From File`	009	8	3		Values read in from external file
`03: IF StockLevel <= ReorderLevel THEN`	009	8	3		8 is NOT smaller than or equal to 3, so line 7 executed next
`07: PRINT 'Stock not reordered'`	009	8	3	'Stock not reordered'	Correct output printed

We would repeat the trace for a range of values, including borderline and low stock levels, for example 3 and 1.

Check your understanding

3 This algorithm is designed to decide what bus fare rate to charge. Complete the trace table for the input value 7.

```
01 age ← USERINPUT
02 IF age < 5 THEN
03   fare ← 'Free'
04 ELSEIF age < 16 THEN
05   fare ← 'Half fare'
06 ELSE
07   fare ← 'Full fare'
08 OUTPUT fare
09 ENDIF
```

Line	age	fare	Output	Comment

Answers on p. 92

Exam tip

While some values are input on one line and do not change, the value of the variable at each stage of the process should be shown.

1.2 Efficiency of algorithms

Understand that more than one algorithm can be used to solve the same problem

REVISED

It is important to note that there are usually many ways of solving the same problem.

```
#Version 1
OUTPUT 1
OUTPUT 2
OUTPUT 3
OUTPUT 4
OUTPUT 5
#Version 2
FOR x ← 1 TO 5
    OUTPUT x
ENDFOR
```

Compare the efficiency of algorithms

REVISED

These two algorithms produce the same results, but for 1000 numbers it would take longer to write the code for the first algorithm. This does not make it less efficient; the efficiency of an algorithm is the time taken for a computer to complete it. Given that computers do not all run at the same speed, counting the number of steps required to complete the process is a way of estimating how efficient an algorithm is.

The efficiency of an algorithm will also depend on the input values; using small values will often show little difference, but using large numbers can often show significant differences in efficiency. This difference can be seen in the possible number of steps taken by a linear search compared to a binary search for large data sets.

1.3 Searching algorithms

Understand and explain how linear search and binary search algorithms work

REVISED

A search algorithm should be able to check a list of items and locate an item or report that the item is not in the list.

Linear search

A linear search looks at each item in turn until it finds the target item or reaches the end of the list. If it reaches the end of the list without finding the target item, it reports that the item has not been found.

Exam tip

You may not be able to remember the searching and sorting algorithms in pseudo-code but you need to understand how they work.

Worked example

Position	0	1	2	3	4
Item	Banana	Grape	Pear	Apple	Strawberry
	↑	↑	↑	↑	

When searching for 'Apple' in this list it looks at each position in turn until it finds Apple.

It will return that Apple was found at position 3.

It will have taken four steps to find the item.

Binary search

A binary search locates an item by continuously splitting the list in half and checking if the midpoint is the value being searched for. If it is not it discards the half that does not contain the item. It requires a list to be sorted in order to do this.

Once it reaches a list with just one item that is not the target item it reports that the item is not found.

When splitting the list into two halves, if there is no midpoint, we take the value to the left of midpoint. (Either side could be taken provided we are consistent about this.)

Worked example

To search for B in this list:

Position	0	1	2	3	4	5	6	7
Letter	A	B	C	D	E	F	G	H

There are eight items. There is no exact midpoint, so we choose D, to the left of the midpoint.

Position	0	1	2	3	4	5	6	7
Letter	A	B	C	D	E	F	G	H
				midpoint				

D is not the item being searched for, and B is before D alphabetically, so we discard the upper half (right-hand side), including the midpoint.

Position	0	1	2
Letter	A	B	C
		midpoint	

There are three items. The midpoint is B.

The algorithm has located B at position 1.

Search method	Advantages	Disadvantages
linear	can search an unordered list	may be slow if the value is near the end of the list needs to search through the complete list to report if a value is not found
binary	usually much faster than a linear search the number of items in the list halves after each iteration to find a value in 1 million numbers takes just 21 iterations; a linear search may need 1 million iterations	the list must be ordered

1.4 Sorting algorithms

Understand and explain how merge sort and bubble sort algorithms work

REVISED

Merge sort

The merge sort splits an unsorted list into individual lists then merges them together into a sorted list.

Consider how two sorted lists can be merged into a single sorted list.

- Comparing the first two items in each list.
- Removing the one that comes first and adding it to a new list.
- Repeating this process until one list is empty.
- Adding the remaining items in the non-empty list to the new list.

Worked example

For the two lists:

List 1	List 2
3, 5	4, 8, 9, 11

Compare 3 and 4. 3 is the smallest so it is removed and placed in the new list.

New list	List 1	List 2
3	5	4, 8, 9, 11

Now we compare 5 and 4. 4 is the smallest so we remove it and place it in the new list.

New list	List 1	List 2
3, 4	5	8, 9, 11

Now we compare 5 and 8. 5 is the smallest so we remove it and place it in the new list.

New list	List 1	List 2
3, 4, 5		8, 9, 11

List 1 is now empty so we append the contents of list 2 onto the new list.

New list
3, 4, 5, 8, 9, 11

If we are starting from an unordered list, we create a set of ordered lists by breaking the unordered list down into lists containing just one item.

We merge each pair of lists in turn into two item lists, then four item lists, and so on, until we have a single sorted list.

Worked example

For the list:

D	A	C	F	B	E	H	G

Split into single item lists:

D	A	C	F	B	E	H	G

Now we merge each pair into two item lists:

A D	C F	B E	G H

Merge in pairs again:

A C D F	B E G H

Merge the list until we have a single sorted list:

A B C D E F G H

Bubble sort

- Start at the beginning of the list and compare the first two items.
- If they are in order, leave them.
- If they are not in order, swap them over and record that a swap has taken place by setting a **flag** to True.
- Now look at items 2 and 3, then 3 and 4, until the end of the list is reached.
- If a swap has been made, start again at the beginning.
- Repeat this process until no swaps are made on a complete pass through the list.

A **flag** is a Boolean variable used to signal if something has happened; in the case of sorting, if a swap has taken place, the flag is set to True. The flag is used to decide if the list was in order and no swaps have taken place or not in order and some swaps have taken place.

Worked example

For the list D A C B E:

D	A	C	B	E	D and A are not in order so we swap them and set a swaps flag to True.
A	D	C	B	E	D and C are not in order so we swap and the swaps flag is True.
A	C	D	B	E	D and B are not in order so we swap and the swaps flag is True.
A	C	B	D	E	D and E are in order so no action.
A	C	B	D	E	Pass completed. The swaps flag is True so we set it to False and start again.
A	C	B	D	E	A and C are in order no action.
A	C	B	D	E	C and B are not in order so we swap and set the swaps flag to True.
A	B	C	D	E	C and D are in order so no action.
A	B	C	D	E	D and E are in order so no action.
A	B	C	D	E	Pass completed. The swaps flag is True so we set it to False and start again.

Since the data is now in order the next pass will complete without any swaps and the swaps flag will be False at the end of the pass. The algorithm stops.

Sort	Advantages	Disadvantages
bubble	easy to program	takes many steps to complete the process, even for a small list inefficient and not suited to large lists
merge	more efficient than a bubble sort for large lists takes the same time for a list regardless of how unordered it is	slow for small lists takes up a lot of memory because it creates so many lists

Check your understanding

4 Show the process for sorting the list C D B A using:
- **a)** bubble sort
- **b)** merge sort.

5 Show the process for finding F in the list A B C D E F G using a binary search.

Answers on p. 92

Exam tip

You may be asked to complete a search or a sort for a given data set. Show the steps and annotate the process to explain how you have completed the task.

Exam checklist

Representing algorithms
- ✚ Understand and explain the terms decomposition and abstraction
- ✚ Understand and explain the term algorithmic thinking
- ✚ Explain algorithms in terms of their input, output and processing
- ✚ Use a systematic approach to problem solving including flowcharts, pseudo-code and program
- ✚ Determine the purpose of simple algorithms using trace tables and visual inspection

Efficiency of algorithms
- ✚ Understand that more than one algorithm can be used to solve the same problem
- ✚ Compare the efficiency of algorithms

Searching and sorting algorithms
- ✚ Understand and explain how linear search and binary search algorithms work
- ✚ Understand and explain how merge sort and bubble sort algorithms work

Now test yourself

TESTED

1 List the **three** key principles of computational thinking and their descriptions.

2 Outline how a linear search and a binary search work and the major differences between them.

3 Outline how bubble and merge sorts work.

Exam-style questions

1 Draw a flowchart to describe the following process. [6 marks]
- ✚ Enter a number between 1 and 10 inclusive.
- ✚ Reject values not in range and request the input again.
- ✚ If it is a valid input calculate the square of the number.
- ✚ Output the value and its square.

2 Write a pseudo-code algorithm to describe a program that asks a user to input five numbers.

The program will output the average of those five numbers. [6 marks]

3 Look at this list of plants:

ivy, broom, poppy, fern, privet, lilac, dogwood, rose

- **a)** Show the steps for a merge sort to put the list of plants into ascending alphabetical order. [4 marks]
- **b)** Explain why a merge sort might be problematic for a set of data with several thousand items in it. [2 marks]

4 Look at this list of animals:

cat, dog, frog, goat, horse, kangaroo, lion, monkey, parrot

- **a)** For the list of animals, show the steps in a binary search to locate **lion**. [3 marks]
- **b)** State how many steps it would take to locate **lion** using a linear search. [1 mark]
- **c)** Explain when you would have to use a linear search to locate an item in a list. [2 marks]

Answers on p. 96

INDEX

T

U

V

W

X

MY **REVISION** NOTES

AQA GCSE (9–1)

COMPUTER SCIENCE

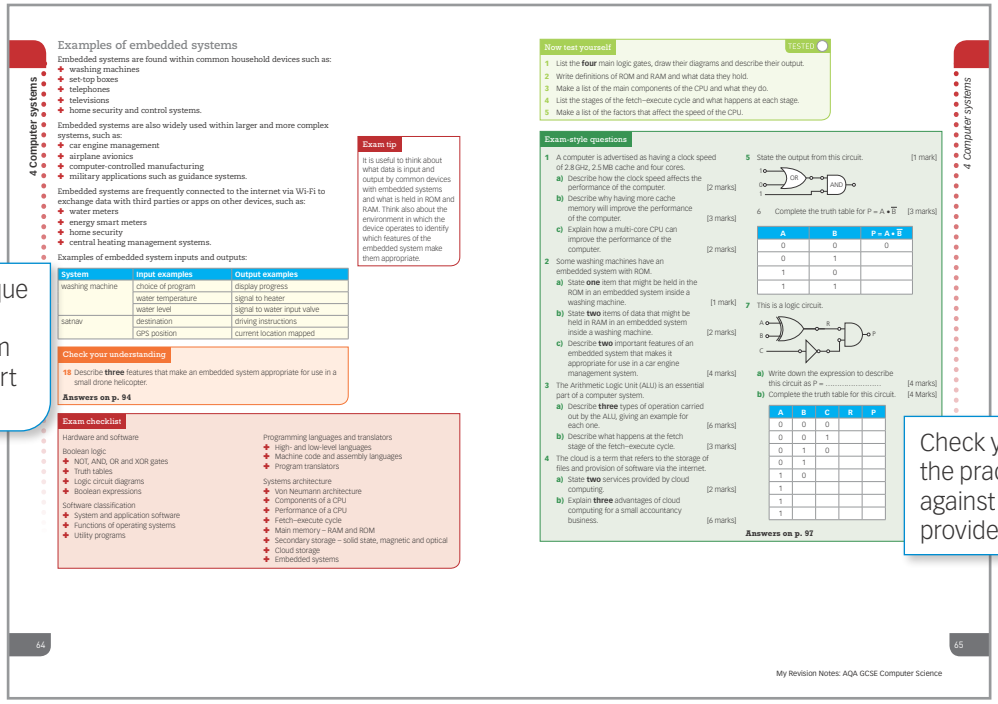
Target exam success with *My Revision Notes*. Our updated approach to revision will help you learn, practise and apply your skills and understanding. Coverage of key content is combined with practical study tips and effective revision strategies to create a guide you can rely on to build both knowledge and confidence.

***My Revision Notes: AQA GCSE (9–1) Computer Science* will help you:**

Strengthen subject knowledge and key terms by working through clear and focused key content

Test understanding and identify areas for improvement with 'check your understanding' questions

Enhance exam technique through exam-style questions and tips from a leading team of expert authors



Examples of embedded systems

Embedded systems are found within common household devices such as:
- washing machines
- set-top boxes
- telephones
- televisions
- home security and control systems.

Embedded systems are also widely used within larger and more complex systems, such as:
- car engine management
- airplane avionics
- computer-controlled manufacturing
- military applications such as guidance systems.

Embedded systems are frequently connected to the internet via Wi-Fi to exchange data with third parties or apps on other devices, such as:
- water meters
- energy smart meters
- home security
- central heating management systems.

Examples of embedded system inputs and outputs:

System	Input examples	Output examples
washing machine	choice of program	display progress
	water temperature	signal to heater
	water level	signal to water input valve
satnav	destination	driving instructions
	GPS position	current location mapped

Exam tip

It is useful to think about what data is input and output by common devices with embedded systems and what is held in ROM and RAM. Think also about the environment in which the device operates to identify which features of the embedded system make them appropriate.

Check your understanding

18 Describe **three** features that make an embedded system appropriate for use in a small drone helicopter.

Answers on p. 94

Exam checklist

Hardware and software

Boolean logic
- NOT, AND, OR and XOR gates
- Truth tables
- Logic circuit diagrams
- Boolean expressions

Software classification
- System and application software
- Functions of operating systems
- Utility programs

Programming languages and translators
- High- and low-level languages
- Machine code and assembly languages
- Program translators

Systems architecture
- Von Neumann architecture
- Components of a CPU
- Performance of a CPU
- Fetch–execute cycle
- Main memory – RAM and ROM
- Secondary storage – solid state, magnetic and optical
- Cloud storage
- Embedded systems

Now test yourself TESTED

1 List the **four** main logic gates, draw their diagrams and describe their output.
2 Write definitions of ROM and RAM and what data they hold.
3 Make a list of the main components of the CPU and what they do.
4 List the stages of the fetch–execute cycle and what happens at each stage.
5 Make a list of the factors that affect the speed of the CPU.

Exam-style questions

1 A computer is advertised as having a clock speed of 2.8 GHz, 2.5 MB cache and four cores.
a) Describe how the clock speed affects the performance of the computer. [2 marks]
b) Describe why having more cache memory will improve the performance of the computer. [3 marks]
c) Explain how a multi-core CPU can improve the performance of the computer. [2 marks]

2 Some washing machines have an embedded system with ROM.
a) State **one** item that might be held in the ROM in an embedded system inside a washing machine. [1 mark]
b) State **two** items of data that might be held in RAM in an embedded system inside a washing machine. [2 marks]
c) Describe **two** important features of an embedded system that makes it appropriate for use in a car engine management system. [4 marks]

3 The Arithmetic Logic Unit (ALU) is an essential part of a computer system.
a) Describe **three** types of operation carried out by the ALU, giving an example for each one. [6 marks]
b) Describe what happens at the fetch stage of the fetch–execute cycle. [3 marks]

4 The cloud is a term that refers to the storage of files and provision of software via the internet.
a) State **two** services provided by cloud computing. [2 marks]
b) Explain **three** advantages of cloud computing for a small accountancy business. [6 marks]

5 State the output from this circuit. [1 mark]

6 Complete the truth table for $P = A \cdot \overline{B}$ [3 marks]

A	B	P = A • $\overline{B}$
0	0	0
0	1	
1	0	
1	1	

7 This is a logic circuit.

a) Write down the expression to describe this circuit as P = ... [4 marks]
b) Complete the truth table for this circuit. [4 Marks]

A	B	C	R	P
0	0	0		
0	0	1		
0	1	0		
0	1			
1	0			
1				
1				
1				

Answers on p. 97

4 Computer systems

64

65

My Revision Notes: AQA GCSE Computer Science

Check your answers to the practice questions against the answers provided

Plan and manage a successful revision programme with the 'exam breakdown', 'countdown to the exams' and 'now test yourself' sections

HODDER EDUCATION
t: 01235 827827
e: education@hachette.co.uk
w: hoddereducation.co.uk

ISBN 978-1-3983-2115-1

FSC