Cambridge Assessment International Education

Interactive

Learner Guide

Cambridge IGCSE™/Cambridge IGCSE (9–1)

Computer Science 0478 / 0984

Cambridge O Level

Computer Science 2210

For examination from 2020

Version 1

Cambridge Pathway

In order to help us develop the highest quality resources, we are undertaking a continuous programme of review; not only to measure the success of our resources but also to highlight areas for improvement and to identify new development needs.

We invite you to complete our survey by visiting the website below. Your comments on the quality and relevance of our resources are very important to us.

www.surveymonkey.co.uk/r/GL6ZNJB

Contents

About this guide

This guide explains what you need to know about your Cambridge Computer Science course and examinations. You should use this guide alongside the support of your teacher.

It will help you to:

- ✓ understand what skills you should develop by taking this Cambridge course
- ✓ understand how you will be assessed
- ✓ understand what we are looking for in the answers you write
- ✓ plan your revision programme
- ✓ revise, by providing revision tips and an interactive revision checklist (Section 5).

In this qualification you will study the principles and practices of computer science. You will develop skills in computational thinking, and applying these skills to write computer programs through the development of algorithms. You will explore current and upcoming developments in computer science and explore the benefits of these developments.

Section 1: Syllabus content - what you need to know about

This section gives you an outline of the syllabus content for this course. Ask your teacher for more detail about each topic. You can also find more detail in the Revision checklists of this guide.

In this course you will study:

- computational thinking; looking at a problem in terms of representing it on a computer, how can a problem be split down to smaller parts to make it more manageable, how can a problem be converted into an algorithm and then a program to solve the problem, how the data requirements for a problem can be found.
- how to write computer programs using a high-level programming language
- the components that make up a computer system and their function
- the ways that computers can communicate through both wired and wireless connections, including networks and across the internet.

Make sure you always check the latest syllabus, which is available at www.cambridgeinternational.org

Section 2: How you will be assessed

You will be assessed at the end of the course using two components:

- Paper 1 Theory
- Paper 2 Problem-solving and programming.

Components at a glance

This table summarises the key information about each examination paper. You can find details and advice on how to approach each component in the 'About each paper/component' sub-section.

Component	Time and marks	Skills assessed	Details	Percentage of qualification
Paper 1 Theory	1 hour 45 minutes 75 marks	Section 1 of the subject content – Theory of computer science	Externally assessed written paper consisting of short-answer and structured questions. All questions are compulsory. No calculators are permitted.	60%
Paper 2 Problem-solving and programming	1 hour 45 minutes 50 marks	Section 2 of the subject content – Practical problem-solving and programming	Externally assessed written paper consisting of short answer and structured questions. All questions are compulsory. The paper is split into sections A and B. There is a pre-release that involves writing one or more computer programs. Section A, worth 20 marks, is based on the pre-release where you will be asked questions about the programs you wrote, and may be asked to reproduce sections of your solution, as well as extensions to the pre-release scenario. Section B, worth 30 marks, is based on section 2 of the subject content and will have a range of computational thinking question.	40%

About each paper

Paper 1

You will need to answer all questions on the paper.

This paper is based on Section 1 of the specification. This will include:

- Data representation
- Communication and internet technologies
- Hardware and software
- Security
- Ethics

The questions often include a combination of:

- Ticking boxes
- Filling gaps in sentences
- Writing short answers
- Writing descriptions and explanations.

The questions are usually between 1 and 8 marks each. One question might have multiple question parts, e.g. 1a, 1b.

You may be asked to perform binary conversions, but you will not have access to a calculator to do this, so make sure you double check your answers.

Read the question carefully. For example, in a question where you have to tick boxes, you may need to tick one box per row, or more than one. It is important that you follow these instructions.

It will be helpful if you have a look at past exam questions and mark schemes, because these will help you to identify the amount of detail you need to give in your answers. Look carefully at the command word in the question, for example 'identify' will only require a few of words, whereas 'describe and explain' will need more detail.

Paper 2

You will need to answer all questions on the paper.

You will receive the pre-release before the exam and will work through this with your teacher. You should be creating your own program, or programs, to produce solutions to the tasks. There are usually three tasks that build on each other, for example you usually can't complete Task 2 until you have finished Task 1.

When producing these programs you need to make sure you understand:

- the data structures you use (e.g. variables, constants, arrays)
- the algorithms that you have produced (how they work, what each line of code does)
- how to test your programs using a range of test data.

You might also want to think about ways that you could extend the program, e.g. what else could you add to it? What other features would be useful?

The first part of the exam paper (Section A) will be based on the pre-release. You could be asked questions about the data structures you used, to describe decisions you made, give explanations of how your programs worked, and be asked to reproduce sections of your code.

In Section A, you need to read the question carefully. The question might require you to write code, draw a flowchart, or give an explanation. If you answer an explanation question by writing an algorithm, then you are not giving an explanation and might not gain any marks, even if your code is correct.

Some questions might ask you to write code, but then explain what these statements mean. In this case, you need to give the code, but also explain what that line of code does.

You cannot take any of your materials for the pre-release into the exam, but there will be a copy of the pre-release at the start of the exam paper to which you can refer back.

Section B is worth 30 marks. This will include questions from Section 2 in the specification, including:

- Algorithm design and problem-solving
- Programming
- Databases.

The questions could be a mix of ticking boxes, drawing lines, completing sentences, drawing flowcharts, completing test tables, writing algorithms, giving single word answers, identify errors in a program, complete a QBE (Query by Example) grid as well as descriptions and explanations.

Section 3: What skills will be assessed

The areas of knowledge, understanding and skills that you will be assessed on are called **assessment objectives** (AO).

The examiners take account of the following skills areas (**AO1, AO2 and AO3**) in the examination papers

- Knowledge and understanding
- Application of knowledge and understanding, and skills
- Analysis, evaluation and conclusions

It is important that you know the different weightings (%) of the assessment objectives, as this affects how the examiner will assess your work.

- Assessment objective 1 (AO1) is worth 53% of the total marks in Paper 1, and 20 % of the total marks in Paper 2.
- Assessment objective 2 (AO2) is worth 27% of the total marks in Paper 1, and 60% of the total marks in Paper 2.
- Assessment objective 3 (AO3) is worth 20% of the total marks in Paper 1, and 20% of the total marks in Paper 2.

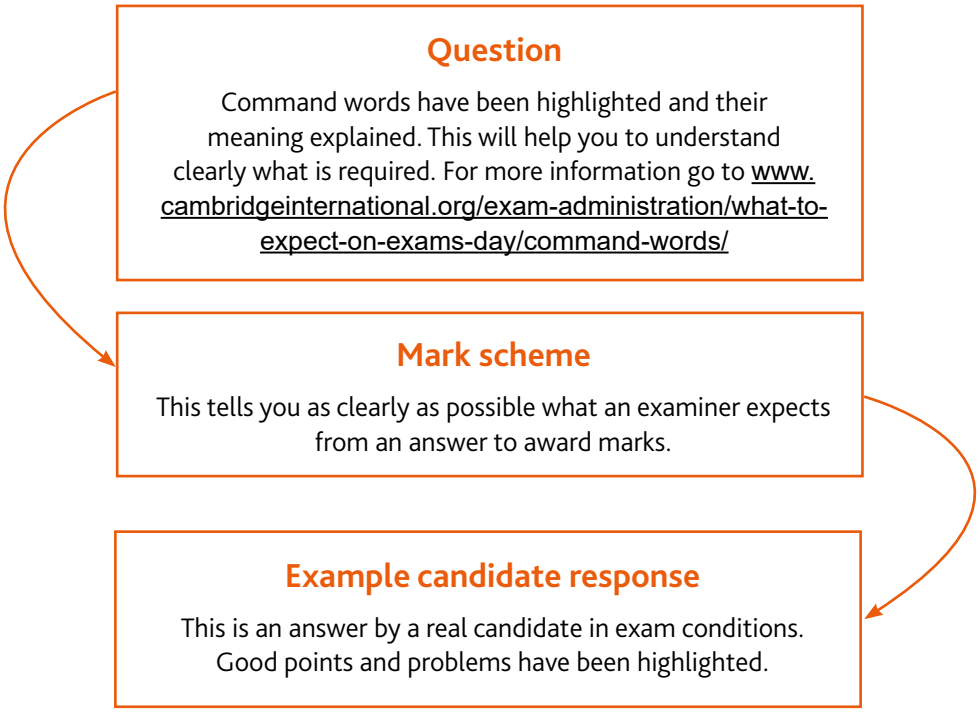
Assessment objectives (AO)	What does the AO mean?	What do you need to be able to do?
AO1 Recall, select and communicate knowledge and understanding of computer technology	This means you will be need to remember definitions and descriptions. These will be generic, i.e. they will not be around a context. You could be asked to define a term, and describe or explain what something means.	You will need to know the definitions of the terms that are in the specification. You will need to be able to give descriptions of what something means, or what something is. You will need to explain how or why something happens, or does not happen. These will all be based around the content that you will have learnt without needing to think about how it could be used in a new way or context.
AO2 Apply knowledge, understanding and skills to solve computing or programming problems	This means you will need to use your knowledge in a specific context. You have to apply what you have learnt.	You will need to consider what you know, and use this in the context given. For example, if the question is about a student needing to store some documents, you need to relate every point you make back to this student, and why your choices, or points, are relevant to them. You will need to use your computational thinking and problem solving knowledge to solve problems. This could be that you are given a flowchart, or program, that you have not seen before. You could be asked questions about that algorithm, and you will need to use what you know and consider it in this example.
AO3 Analysis, evaluate, make reasoned judgements and present conclusions	This means you will need to decide why one decision is more appropriate than another and explain why.	You will need to look at a context and decide what is most appropriate for it. For example, is it more appropriate to use lossy compression for an image instead of lossless. Then, you need to defend your choice, by explaining why it is the best decision. You will need to do this by referring every point to the context, so you are not giving generic definitions or descriptions.

Section 4: Example candidate response

This section takes you through an example question and learner response from a Cambridge past paper. It will help you to see how to identify command words within questions and to understand what is required in your response. A command word is the part of the question that tells you what you need to do with your knowledge. For example, you might need to describe something, explain something, argue a point of view or list what you know.

All information and advice in this section is specific to the example question and response being demonstrated. It should give you an idea of how your responses might be viewed by an examiner but it is not a list of what to do in all questions. In your own examination, you will need to pay careful attention to what each question is asking you to do.

This section is separated as follows:



Question

5 (a) Write an algorithm, using pseudocode and a `FOR ... TO ... NEXT` loop structure, to input 1000 numbers into an array.

.....

.....

.....

.....

.....

.....[2]

The question tells you how the examiner would like you to write the algorithm. It says you must use a FOR …TO … NEXT loop, and that 1000 numbers need to be input into an array.

Mark scheme

5 (a) 1 mark for `FOR ... TO ... NEXT` 1 mark for `INPUT`

```
FOR Count ← 1 TO 1000
  INPUT A[Count]
NEXT (Count)
```

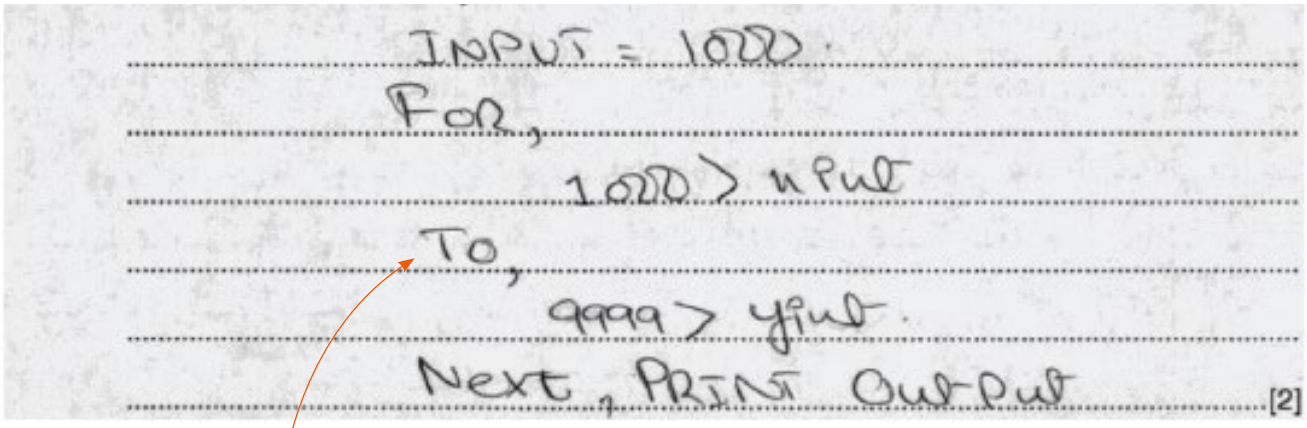
[2]

In the mark scheme the first mark is for using an appropriate FOR .. TO .. NEXT loop. This has the format:

```
FOR loopCounter = startValue TO endValue
    code inside the loop
NEXT loopCounter
```

Example candidate response

5 (a) Write an algorithm, using pseudocode and a `FOR ... TO ... NEXT` loop structure, to input 1000 numbers into an array.



In this answer, the candidate has not used a suitable FOR..TO..NEXT loop structure. They are missing the loop counter, so cannot be awarded the first mark point. For example:

```
FOR count = 0 TO 999
    code inside the loop
NEXT count
```

The second mark point is for reading values 1000 into an array, and the candidate is not awarded this mark point. An array needs to have an identifier followed by the required (square or round) brackets for the index. For example, `array[count]`

Section 5: Revision

This advice will help you revise and prepare for the examinations. It is divided into general advice for all papers and more specific advice for Paper 1 and Paper 2.

Use the tick boxes ☐ to keep a record of what you have done, what you plan to do or what you understand.

General advice

Before the examination

- ☐ Find out when the examinations are and plan your revision so you have time to revise. Create a revision timetable and divide it into sections to cover each topic.
- ☐ Find out how long each paper is, how many questions you have to answer, how many marks there are for each question, and work out how long you have for each question.
- ☐ Find out the choices you have on each paper, make sure you know how many sections there are and which sections you should answer from.
- ☐ When there is a choice of questions in a section, make sure you revise enough to have a choice.
- ☐ Know the meaning of the command words used in questions and how to apply them to the information given. Look at past examination papers and highlight the command words and check what they mean.
- ☐ Make revision notes. Try different styles of notes.
- ☐ Work for short periods then have a break. Revise small sections of the syllabus at a time.
- ☐ Test yourself by writing out key points, redrawing diagrams, creating key cards with the words on one side and definitions and/or examples on the back, etc.
- ☐ Make sure you practice writing selection and iteration constructs. This should include converting between, for example, two different types of loop. You can practice these with pseudocode, or use your chosen programming language to test them and make sure they work.
- ☐ Definitions must not reuse the words to be defined. E.g. *selection* means selecting which code to run. Instead it determines which code to run based on a condition.
- ☐ Make your own dictionary or draw up a glossary of key terms for each section of the syllabus.
- ☐ Look at the programs you have written during lessons and walk through what each line of code does, then working out what the whole program does. This can be tested by running the program and seeing if you are correct.
- ☐ Take the descriptions for programs you have written during lessons, and create the programs again without looking at the previous solutions. Test your programs and amend them until they meet the requirements.
- ☐ Revisit the programs you wrote for the pre-release. Practice writing parts of the program from memory, as well as writing descriptions of how your solutions work. Think about ways that you could extend the programs, or improve the programs and implement your ideas.
- ☐ Have a look at past questions so that you are clear of what to expect in an examination.
- ☐ Look at mark schemes to help you to understand how the marks are awarded for each question.

In the examination

- ☐ Read the instructions carefully and answer the right number of questions from the right sections.
- ☐ Do not answer more questions than are needed, as this will not gain you more marks in the examination.
- ☐ Plan your time according to the marks for each question. For example, a question worth 3 marks requires less time and a shorter answer than one worth 10 marks. If a question has several parts, then the parts with more marks will need more time and more developed answers.
- ☐ Do not leave out questions or parts of questions. Remember, no answer means no mark.

- [ ] Read each question very carefully.
  - Identify the command words – you could underline or highlight them.
  - Identify the other key words and perhaps underline them too.
  - Try to put the question into your own words to understand what it is really asking.
- [ ] **Answer the question.** This is very important!
- [ ] Use your knowledge and understanding.
- [ ] Do not just write all you know, only write what is needed to answer the question.
- [ ] Plan your answers. Clear, concise, well-ordered, well-argued, well-supported answers get more marks than long, rambling, muddled, repetitious answers. Quality is better than quantity.
- [ ] Double check your calculations to make sure that you have not made an arithmetic error.
- [ ] Make sure your writing is clear and easy to read. It is no good writing a brilliant answer if the examiner cannot read it!

Paper 1 advice

- [ ] Check the command word that is used, for example describe requires more than a statement. Explain requires more detail as to why something happens, or why you have selected the answer you have.
- [ ] When drawing logic gate diagrams repeat the process backwards to make sure your answer is correct. For example draw the diagram for the statement, then write the statement or your diagram and check that they match.
- [ ] Check whether tick box questions require one, or possibly more ticks per row. If it is one tick per row then make sure you don't put two. If there could be more check each possibility, it is very likely that at least one of the rows requires two or more ticks.
- [ ] If a question gives a context, and the wording of the command refers back to this context, then make sure you use it in your answer. Don't just give a generic answer, use the context in each point you make.
- [ ] If a question asks for differences between two or more items, then make sure you give all sides. For example just stating that parallel transmission has multiple wires is not enough without also stating that serial has just one wire.
- [ ] When converting numbers between different types (e.g. binary, denary, hexadecimal), always check your answers by working it the reverse. For example, if converting binary to denary, convert your denary answer back to binary and make sure that they both match.

Paper 2 advice

- [ ] In Section 1 make sure you re-read the pre-release before answering each question part to make sure you are using the correct Task in your answer.
- [ ] Make sure you have answered the algorithm questions in the way they want the answer. If the question asks for a flowchart, make sure you draw a flowchart. If a question asks for a description, then only drawing a flowchart or writing code will not be acceptable.
- [ ] Split your time appropriately between Section 1 and Section 2. Section 1 should be approximately 40 minutes. Do not got too much over this otherwise you may not have enough time for the rest of the exam paper, you can always come back later and complete it.
- [ ] If you get stuck on an algorithm, don't spend a long time trying to fix it. Move on and you can always come back later after you've answered the other questions, in that time away you might be able to look at it afresh.
- [ ] Test run any algorithms you write. Use some sample data and work through each line of your algorithm to make sure it works. If it doesn't, then you should be able to find out where the problem is.
- [ ] Check your spelling and case (lower or upper) especially with any database questions, e.g. queries. Spelling and case must be exact, if in the database table it is called HOUSE, then writing house is inaccurate. After you have answered each question check the words you have used against those in the question.
- [ ] If a question asks for an example, make sure you give one as there will be at least 1 mark available for this.

Revision checklists

In the next part of this guide we have provided some revision checklists. These include information from the syllabus that you should revise. They don't contain all the detailed knowledge you need to know, just an overview. For more detail see the syllabus and talk to your teacher.

The table headings are explained below:

Topic	You should be able to	R	A	G	Comments
These are the core topics from the specification.	Content in the syllabus you need to cover.	You can use the tick boxes to show when you have revised an item and how confident you feel about it. R = RED means you are really unsure and lack confidence; you might want to focus your revision here and possibly talk to your teacher for help A = AMBER means you are reasonably confident but need some extra practice G = GREEN means you are very confident. As your revision progresses, you can concentrate on the RED and AMBER items in order to turn them into GREEN items. You might find it helpful to highlight each topic in red, orange or green to help you prioritise.			You can: • Add further information of your own, such as names of case studies needed. • add learning aids, such as rhymes, poems or word play • pinpoint areas of difficulty you need to check further with your teacher or textbooks • include reference to a useful resource

Note: the tables below cannot contain absolutely everything you need to know, but it does use examples wherever it can.

Paper 1 Reading and writing

1.1 Data representation

Topic	You should be able to	R	A	G	Comments
1.1.1 Binary systems	• convert positive denary integers into binary	☐	☐	☐	
	• convert positive binary integers into denary	☐	☐	☐	
	• describe the concept of a byte	☐	☐	☐	
	• identify how many bits are in a byte	☐	☐	☐	
	• describe how a byte is used to measure memory size	☐	☐	☐	
	• describe the use of binary in registers	☐	☐	☐	
1.1.2 Hexadecimal	• convert positive denary integers into hexadecimal	☐	☐	☐	
	• convert positive hexadecimal integers into denary	☐	☐	☐	
	• convert positive hexadecimal integers into binary	☐	☐	☐	
	• convert positive binary integers into hexadecimal	☐	☐	☐	
	• describe how numbers stores in registers and main memory could be in hexadecimal	☐	☐	☐	
	• identify and describe common uses of hexadecimal numbers	☐	☐	☐	
1.1.3 Data storage	• describe how data in a computer can represent sound, pictures, video, text and numbers	☐	☐	☐	
	• describe how data can be stored in different formats	☐	☐	☐	
	• describe the need for error detection and correction	☐	☐	☐	
	• complete parity bits and describe parity checks	☐	☐	☐	

You should be able to	Ways to practise skills	R	A	G	Comments
	• describe the use of check digits, and calculate check digits				
	• describe the use of checksums, and calculate checksums				
	• describe the use of Automatic Repeat reQuests (ARQ)				
	• describe what is meant by a Musical instrument Digital Interface (MID) files				
	• describe the characteristics of MIDI files, JPEG files, MP2 and MP4 files				
	• describe the principles of lossless compression in music/video, photos and text files				
	• describe the principles of lossy compression in music/video, photos and text files				

1.2 Communication and internet technologies

You should be able to	Ways to practise skills	R	A	G	Comments
1.2.1 Data transmission	• describe what is meant by the transmission of data				
	• describe serial and parallel data transmission				
	• describe simplex, duplex and half-duplex data transmission				
	• justify the use of serial or parallel data transmission in a scenario				
	• justify the use of simplex, duplex and half-duplex data transmission				
	• describe the need to check for errors in data transmission				
	• describe how parity bits can be used in data detection				
	• complete parity bits for a byte of data				
	• describe the use of serial and parallel data transmission in USB ports and Integrated circuit				

You should be able to	Ways to practise skills	R	A	G	Comments
1.2.2 Security aspects	• describe the security threats associated with using the Internet, including: – malware (viruses and spyware) – hacking • describe how anti-virus and other protection software can protect the user from the security risks	☐	☐	☐	
1.2.3 Internet principles of operation	• describe the role of the web browser • describe the role of an Internet Service Provider (ISP) • describe what is meant by hypertext transfer protocol (http and https) • describe what is meant by HTML • describe the differences between HTML structure and presentation • describe the use MAC addresses • describe the use of IP addresses • describe the use of Uniform Resource Locators (URLs) • describe the use of cookies	☐	☐	☐	

1.3 Hardware and software

You should be able to	Ways to practise skills	R	A	G	Comments
1.3.1 Logic gates	• describe the use of logic gates in electronic circuits • describe the function of NOT, AND, OR, NAND, NOR and XOR (EOR) gates • give the binary output produced by each gate for all possible 2 binary inputs • draw a truth table for a logic gate • recognise a logic gate from its truth table	☐	☐	☐	

You should be able to	Ways to practise skills	R	A	G	Comments
	• identify the standard symbols for each gate	☐	☐	☐	
	• produce a truth table for a given logic circuit	☐	☐	☐	
	• produce a logic circuit to either solve a problem, or implement a written logic statement	☐	☐	☐	
1.3.2 Computer architecture and the fetch-execute cycle	• describe the Von Neumann model for a computer system	☐	☐	☐	
	• describe the stored program concept	☐	☐	☐	
	• describe the stages of the fetch-execute cycle	☐	☐	☐	
	• describe the use of registers in the fetch-execute cycle	☐	☐	☐	
	• describe the use of buses in the fetch-execute cycle	☐	☐	☐	
1.3.3 Input devices	• describe the principle operation of – 2D scanners – 3D scanners – barcode readers – Quick Response (QR) code readers – digital cameras – keyboards – mice – touch screens – interactive whiteboards – microphones	☐	☐	☐	
	• describe the use of input and output devices in real-life scenarios	☐	☐	☐	
	• describe the use sensors as input devices including – light sensor – temperature sensor – magnetic field sensor – gas sensor – pressure sensor – moisture sensor	☐	☐	☐	

You should be able to	Ways to practise skills	R	A	G	Comments
	– humidity sensor – pH sensor – motion sensor • describe the use of sensors in real-life scenarios	☐ ☐	☐ ☐	☐ ☐	
1.3.4 Output devices	• describe the principle operation of: – inkjet, laser and 3D printers – 2D and 3D cutters – speakers and headphones – actuators – flat-panel display screens (LCD and LED) – LCD projectors and Digital Light projectors • describe the use of output devices in real-life scenarios	☐ ☐	☐ ☐	☐ ☐	
1.3.5 Memory, storage devices and media	• describe the use of primary memory including ROM and RAM • describe the use of secondary memory include HDD and SSD • describe the use of off-line storage including DVD, CD, Blu-ray, USB flash memory, removable HDD • describe the principle operation of magnetic storage devices • describe the principle operation of optical storage devices • describe the principle operation of solid state storage devices • identify whether a given storage device is optical, magnetic or solid state • calculate the storage requirement of a file	☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐	☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐	☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐	
1.3.6 Operating systems	• describe the purpose of an operating system • describe the function of an operating system • describe the need for interrupts	☐ ☐ ☐	☐ ☐ ☐	☐ ☐ ☐	

You should be able to	Ways to practise skills	R	A	G	Comments
1.3.7 High- and low-level languages and their translators	• describe the need for high-level languages • describe the need for low-level languages • describe the need for and use of compilers • describe the need for and use of interpreters • describe the need for assemblers	☐	☐	☐	

1.4 Security

You should be able to	Ways to practise skills	R	A	G	Comments
1.4.1	• describe the need to keep data safe from accidental damage, including corruption and human errors • describe the need to keep data safe from malicious actions, including unauthorised viewing, deleting, copying and corruption	☐	☐	☐	
1.4.2	• identify methods of keeping data safe when stored and transmitted including the use of: – passwords (both keyboard and biometric) – firewalls (software, hardware and proxy servers) – security protocols Secure Socket Layer (SSL) and Transport Layer Security (TLS) – symmetric encryption (plain text, cypher text and use of a key including that the length of a key increases the strength of the encryption)	☐	☐	☐	
1.4.3	• describe the need to keep online systems safe from attacks including denial of service, phishing and pharming	☐	☐	☐	
1.4.4	• apply the points from 1.4.1, 1.4.2 and 1.4.3 to real-life scenarios	☐	☐	☐	

1.5 Ethics

You should be able to	Ways to practise skills	R	A	G	Comments
1.5 Ethics	• describe the need for computer ethics • describe the need for and use of copyright issues and plagiarism	☐	☐	☐	

You should be able to	Ways to practise skills	R	A	G	Comments
	• describe free software, freeware and shareware				
	• describe the ethical issues raised by the spread of electronic communication and computer systems, including hacking, cracking and production of malware				

Paper 2 Practical Problem-solving and Programming

2.1 Algorithm design and problem-solving

You should be able to	Ways to practise skills	R	A	G	Comments
2.1.1 Problem-solving and design	• Explain how computer systems are made up of sub-systems that are in turn made up of further sub-systems				
	• use a top-down approach to design a solution including: – structure diagrams – flowcharts – pseudocode – library routines – sub-routines				
	• work out the purpose of an algorithm				
	• describe standard methods of solutions				
	• identify appropriate test data				
	• use appropriate test data to test a system				
	• describe the need for validation and verification checks on data input				
	• describe the validation checks range check, length check, type check and check digit				
	• complete trace tables to find the value of variables at each step in an algorithm				
	• identify errors in algorithms and suggest ways of removing these errors				
	• produce an algorithm for a given problem using a flowchart				

You should be able to	Ways to practise skills	R	A	G	Comments
	• produce an algorithm for a give problem using pseudocode	☐	☐	☐	
	• comment on the effectiveness of a given solution	☐	☐	☐	
2.1.2 Pseudocode and flowcharts	• understand and use ← for pseudocode assignment	☐	☐	☐	

	• understand and use the conditional statements
	`IF .. THEN .. ELSE .. ENDIF`
	`CASE .. OR ..OTHERWISE .. ENDCASE`
		☐	☐	☐	

	• understand and use the pseudocode loop structures
	`FOR .. TO .. NEXT`
	`REPEAT .. UNTIL`
	`WHILE .. DO .. ENDWHILE`
		☐	☐	☐	

	• understand and use the following pseudocode commands and statements
	`INPUT`
	`OUTPUT`
		☐	☐	☐	

	• understand and write programs that use totalling	☐	☐	☐	
	• understand and write programs that use counting	☐	☐	☐	
	• understand and use standard symbols to represent the given statements, commands and structures	☐	☐	☐	

2.2 Programming

You should be able to	Ways to practise skills	R	A	G	Comments
2.2.1 Programming concepts	• describe the purpose of variables	☐	☐	☐	
	• declare and use variables	☐	☐	☐	
	• describe the purpose of constants	☐	☐	☐	
	• declare and use constants	☐	☐	☐	

You should be able to	Ways to practise skills	R	A	G	Comments
	• describe and use the basic data types: Integer, Read, Char, String and Boolean • describe and use the concepts of: – sequence – selection – repetition (iteration/loops) – totalling – counting • use predefined procedures/functions	☐	☐	☐	
2.2.2 Data structures; arrays	• describe the concept of an array • declare and use one-dimensional arrays • use variables as an index in an array • read values into an array using a `FOR .. TO .. NEXT` loop • write values into an array using `FOR .. TO .. NEXT` loop	☐	☐	☐	

2.3 Databases

You should be able to	Ways to practise skills	R	A	G	Comments
2.3 Databases	• define a single-table database from given data storage requirements • choose and specify suitable data types for fields • describe the purpose and characteristics of a primary key • choose a suitable primary key for a database table • follow a query-by-example for a given database table • complete a query-by example for given search criteria	☐	☐	☐	

Section 6: Useful websites

The websites listed below are useful resources to help you study for your Cambridge IGCSE Computer Science course.

www.bbc.co.uk/bitesize/subjects/z34k7ty
Includes several theory and programming aspects relevant to this specification with quizzes.

www.khanacademy.org/
Website allows learners to sign up and practice their programming techniques independently, particularly useful for Paper 2.

www.101computing.net/LMC/
Simulation of a processor including the different registers. Learners can enter pre-written programs and watch how and when the values changes.

www.computerscience.gcse.guru
Website with theory and quizzes, written for a range of specifications so not all will be relevant.

Cambridge Assessment International Education
The Triangle Building, Shaftesbury Road, Cambridge, CB2 8EA, United Kingdom
t: +44 1223 553554
e: info@cambridgeinternational.org www.cambridgeinternational.org