

# INTERNATIONAL GCSE

## Computer Science (9-1)

### GETTING STARTED GUIDE

---

Pearson Edexcel International GCSE in Computer Science (4CP0)

---

For first teaching in September 2017

First examination June 2019



Pearson Education Limited is one of the UK's largest awarding organisations, offering academic and vocational qualifications and testing to schools, colleges, employers and other places of learning, both in the UK and internationally. Qualifications offered include GCSE, AS and A Level, NVQ and our BTEC suite of vocational qualifications, ranging from Entry Level to BTEC Higher National Diplomas. Pearson Education Limited administers Edexcel GCE examinations.

Through initiatives such as onscreen marking and administration, Pearson is leading the way in using technology to modernise educational assessment, and to support teachers and learners.

This guide is Issue 1. We will inform centres of any changes to this issue. The latest issue can be found on the Edexcel website:  
[www.edexcel.com/ial](http://www.edexcel.com/ial)

References to third-party material made in this guide are made in good faith. We do not endorse, approve or accept responsibility for the content of materials, which may be subject to change, or any opinions expressed therein. (Material may include textbooks, journals, magazines and other publications and websites.)

ISBN 978-1-4469-4233-8

All the material in this publication is copyright  
© Pearson Education Limited 2015

<b>A</b>	<b>Getting started for teachers</b>	<b>2</b>
	<u>Introduction</u>	2
	<u>Key features of the qualification</u>	3
	<u>Summary of assessment</u>	4
	<u>Content guidance</u>	5
	<u>Assessment guidance</u>	12
	<u>Suggestions for teaching problem solving and programming</u>	17
	<u>Arrangements for Paper 2 –practical examination</u>	18
	<u>Delivery of the qualification – transferable skills</u>	20
	<u>Suggested resources</u>	21
	<u>Course planner</u>	23
<b>B</b>	<b>Getting started for students</b>	<b>25</b>
	<u>Student guide</u>	25

## Introduction

This Getting Started Guide provides an overview of the Pearson Edexcel International GCSE in Computer Science to help you get to grips with the content and assessment. It will help you to understand what you have to teach and how to prepare your students for the external assessments.

### Support for delivering the new specification

Our package of support to help you plan and implement the new specification includes:

**Planning** – Download and customise our editable course planner and scheme of work.

**Developing skills** – Our subject-specific skills map highlights opportunities for students to develop transferrable skills.

**Understanding the assessment requirements** – The sample assessment material will help you and your students familiarise yourselves with the format and level of demand of the two examination papers.

**Analysing student performance** – Our online ResultsPlus service provides detailed analysis of your students' exam performance, helping you to identify topics and skills where students would benefit from further learning.

**Help and support** – Our subject advisory service, led by Tim Brady, is a direct and personal source of help and support. Sign up to receive Tim's regular e-newsletter, which will keep you up to date with qualification and service news. [TeachingICT@pearson.com](mailto:TeachingICT@pearson.com)

**Training events** – We host a series of training events each year to help teachers deepen their understanding of our qualifications.

## Key features of the qualification

We have consulted widely with all parts of the international school subject community to develop a qualification that meets international learner needs, is of a comparable standard to our regulated GCSE in Computer Science and complements our popular International GCSE in ICT.

Our International GCSE in Computer Science has the following key features:

**Engaging, contemporary content** – emphasising the real-world relevance of computer science.

**Focus on computational thinking** – equipping students with valuable transferable skills.

**Examination only assessment** – consisting of a written examination and a practical programming examination carried out on a computer.

**Clear and straightforward question papers** – designed to be accessible to students of all abilities.

**Choice of three programming languages** – enabling you to select the language best suited to your students.

**Fosters progression** – to further study at A Level and beyond.

### Choosing between ICT and Computer Science

ICT and Computer Science are unique and complementary subjects serving distinct purposes. Pearson offers International GCSEs in both. Centres wishing to teach students how to use computer systems safely and effectively should opt for the International GCSE in ICT. Whereas those who want their students to study computation and learn how it can be applied to solving problems should choose the International GCSE in Computer Science.

Should a student wish to do so, they can study both qualifications.

## Summary of assessment

The International GCSE in Computer Science has two untiered assessment components. Both are set and marked by Pearson.

<b>Paper 1: Principles of Computer Science</b> Paper code: 4CPO/01	<b>Paper 2: Application of Computational Thinking</b> Paper code: 4CPO/02
<b>What is assessed?</b>	
Content from all six topics of the subject content.	Problem solving and programming are the main focus, but will also assess other aspects of the subject content.
<b>How is it assessed?</b>	
<ul style="list-style-type: none"> <li>• A 2-hour written examination</li> <li>• All questions are compulsory</li> <li>• Marked out of 80</li> <li>• Worth 50% of the qualification</li> </ul>	<ul style="list-style-type: none"> <li>• A 3-hour practical examination, requiring the use of a computer</li> <li>• Assessment is available within a window, the dates of which are specified by Pearson. All questions are compulsory</li> <li>• Marked out of 80</li> <li>• Worth 50% of the qualification</li> <li>• Candidates must use one of the three approved high-level programming languages (Python, C# or Java)</li> <li>• Internet access is not permitted-</li> </ul>
<b>What type of questions will there be?</b>	
<p>A mix of multiple-choice, short open-response, open-response and extended open-response questions.</p> <p>Some questions will involve interpreting, completing, correcting and constructing algorithms.</p> <p>There will always be at least one essay-style question worth 6 marks.</p>	<p>Mainly task-based questions. Students will be expected to answer questions about program code, correct and amend program code, and convert algorithms into program code.</p> <p>There will also be some more 'traditional' questions that assess students' knowledge and understanding of computer science.</p>

International GCSE exams and certification for this specification are available for the first time in June 2019 and then every June thereafter for the life of the specification.

The sample assessment materials (SAMs) are a valuable source of reference. They illustrate the format and style of the two papers and the type of questions students are likely to encounter.

This qualification is graded on a nine-grade scale from 9 to 1 using the total subject mark, where 9 is the highest grade.

## Content guidance

### Overview

The compulsory content of the International GCSE in Computer Science combines knowledge and understanding of the principles of computer science with practical problem-solving and programming skills. It is arranged as six topics.

1. Problem solving
2. Programming
3. Data
4. Computers
5. Communication and the internet
6. The bigger picture

Each topic is divided into a number of sections and each section consists of a set of numbered bullets. As a minimum, all the numbered bullet points in the content must be taught.

### Topic 1: Problem solving

This topic specifies the problem-solving techniques and methods students are expected to understand and be able to use.

The introduction emphasises the need to give students repeated opportunities to tackle computational problems, including some substantial problem-solving tasks.

Section 1.1 deals with algorithms. Students are expected to interpret and create algorithms expressed as flow charts, pseudocode, written descriptions and program code (1.1.1, 1.1.2). They should be familiar with the pseudocode commands and flow chart symbols listed in Appendices 5 and 6 of the specification. Any examination question that uses pseudocode will use the Edexcel version provided in Appendix 5. See, for example, SAM Paper 1 Q7(a) and Q8(c) and SAM Paper 2 Q2(a). When writing their own pseudocode students may use any form, providing it is clear and unambiguous.

Students must be able to deduce how algorithms work (1.1.3), determine the correct output of an algorithm for a given set of data (1.1.4) and find and correct errors in algorithms (1.1.5).

Four standard sorting and searching algorithms are specified in statement 1.1.8. Students need to understand how each of these algorithms work and be able to demonstrate their operation on a given set of data. SAM Paper 2 Q4(b) and Q4(c) illustrate the type of questions students are likely to encounter. Students also need an understanding of how the choice of algorithm is influenced by the data structures and data values to be manipulated (1.1.7)

Students must be able to evaluate the fitness for purpose of algorithms (1.1.9). In doing so, they should consider factors such as the number of lines of code required or the type of iteration used. They do not have to be able to calculate the efficiency of an algorithm.

Section 1.2 focuses on decomposition and abstraction – two very important aspects of computational thinking. Decomposition involves breaking a complex problems down into a number of sub-problems, each of which accomplishes a specific task. Abstraction is the process of removing any unnecessary detail from a problem. Q6 on SAM Paper 2 illustrates how students' practical use of these techniques may be assessed.

### Topic 2: Programming

Topic 2 focuses on the programming knowledge and skills students need to learn and practise in order to turn algorithms into executable programming code and should be taught in conjunction with Topic 1.

The topic introduction states that students should be competent at designing, reading, writing and debugging programs and should be given repeated opportunities to develop and practise their programming skills.

There are three approved high-level programming languages to choose from: Python, C# and Java. Students must develop their programming knowledge and skills in at least one of these.

Section 2.1 covers various aspects of developing and testing program code. Statement 2.1.2 identifies the techniques students should use to improve the readability of their code and explain how it works. SAM Paper 2, Q1(b) illustrates how this content may be assessed.

Types of error and error correction are the focus of statements 2.1.3, 2.1.5 and 2.1.6. SAM Paper 2 Q1(c) and Q5(a) are examples of the type of questions students may encounter.

Statement 2.1.4 deals with test plans and test data and statement 2.1.7 with program evaluation.

The structural components and programming constructs that students are expected to recognise and use in their programs are specified in section 2.2. They include straightforward and nested selection, definite (count controlled) and indefinite iteration, with the terminating condition at the start or the end of the iterative structure. SAM Paper 2 Q1(a) illustrates how students' knowledge and understanding of this content may be assessed.

Section 2.3 specifies the data types and data structures students should understand and be able to use. Depending on the programming language being used, these may have other names. For example, in Python real numbers are referred to as floats and the nearest data structure to an array is a list. In the examinations we will use the generic names given in this section of the specification. SAM Paper 2 Q6 illustrates how students' understanding of two-dimensional arrays and their ability to write code that uses data structures may be assessed.



Section 2.4 deals with writing code that accepts and responds to user input (2.4.1) and working with text files (2.4.3). SAM Paper 2 Q5(b) provides an example of how students' ability to write code that responds to user input may be assessed, whilst SAM Paper 2 Q3(c) illustrates how practical file handling may be tested.

Students are expected to understand the need for data validation and be able to incorporate simple data validation checks in their own code, for example checking that an entered string, such as a password, has a minimum length (2.4.2). See SAM Paper 2 Q2(b) for an example of the type of question on validation that students may encounter.

Section 2.5 specifies the arithmetic, relational and logic operators students need to understand and be able to use in their algorithms/programs.

Section 2.6 focuses on subprograms. Students are expected to understand the benefits of using subprograms and be able to write programs that make use of subprograms they have written themselves as well as pre-existing library modules. They must understand the difference between functions and procedures, even if the high-level programming language they are using does not differentiate between the two.

Students should understand the different parts of a subprogram definition. They should be able to generalise a subprogram by adding parameters to its header, using the parameters inside the subprogram and amending the call to the subprogram to pass a value or values to the subprogram.

Students must be able to use global and local variables when implementing subprograms (2.3.5). SAM Paper 2 Q3(a) and Q3(b) illustrate the type of questions students are likely to encounter.

Please note that students are not expected to develop anything more than simple text (console) mode programs and do not need to know how to define objects and classes or write interactive programs with a graphical interface.

### Topic 3: Data

This is a wide-ranging topic. Section 3.1 deals with how binary is used to represent and manipulate different types of number. Students are expected to be familiar with binary, decimal and hexadecimal number bases and be able to convert between them. SAM Paper 1 Q1(d) and Q2(b)(i) illustrate how this content is likely to be assessed.

Students are expected to understand two methods of representing signed integers – sign and magnitude and two's complement (3.1.2). SAM Paper 1Q5(b)(iii) illustrates how students' ability to apply their understanding may be assessed.

Students do not need to know how real numbers are represented.

Statement 3.1.4 specifies the binary arithmetic students must be able to perform. They must learn about the concept of overflow in relation to binary addition and are expected to understand and use both logical and arithmetic binary shifts to perform multiplication/division.

Section 3.2 focuses on how different types of data are represented in binary. Statement 3.2.1 focuses on how characters are encoded. Students are expected to learn about ASCII and Unicode and understand that the latter allows a far greater number of characters to be represented. They should be aware that character codes are grouped and run in sequence and – given the code for one character – be able to derive the code for another.

Statement 3.2.2 focuses on representation of bitmap images. SAM Paper 1Q6(a) and Q6(b) provide examples of how this content may be assessed.

Statement 3.2.3 deals with the process of converting an analogue sound signal into digital format. Students should understand the effect on accuracy and file size of increasing the sampling frequency. SAM Paper 1 Q6(d) illustrates how this content may be assessed.

Statement 3.2.4 focuses on the limitations of binary representation and the trade-off between file size and quality.

Section 3.3 covers data storage and compression. The binary and denary multiples students need to be able to work with are listed in statement 3.3.1. As SAM Paper 1 Q6(c) illustrates, students must be able to construct expressions to convert from one unit of information to another.

Students are expected to understand the need for data compression and the characteristics of lossless and lossy compression methods (3.3.2). They need to know how a lossless run-length encoding (RLE) algorithm works and be able to produce the frequency/value pairs for a given set of data (3.3.3), see SAM Paper 1 Q1(c).

Section 3.4 focuses on data encryption. Students should understand the need for encryption (3.4.1) and be able to describe/demonstrate how four simple encryption ciphers (Pigpen, Caesar, Vigenère and Rail Fence) work (3.4.2). SAM Paper 1 Q1(b) illustrates the type of question addressing this content that students are likely to encounter.

### Topic 4: Computers

Topic 4 is concerned with the hardware and software components of a computer system. The introduction makes it clear that students are expected to recognise that computers take many forms, from embedded microprocessors to distributed clouds.

Section 4.1 covers the input-process-output model and specifies three computational models (sequential, parallel and multi-agent) that students should be familiar with (4.1.2).

Section 4.2 focuses on hardware. Students are expected to understand the function of the components of a computer system listed in statement 4.2.1. They should know how main

memory differs from secondary storage and why secondary storage is necessary. They should understand the purpose of different types of memory (4.2.2). SAM Paper 1 Q4(a)(i) and (ii) illustrate the use of a diagram to assess students' understanding of some aspects of this content.

Statement 4.2.3 hones in on the stored program concept. Students should be aware that both data and instructions are stored in memory, how the fetch-decode-execute cycle works and the role of specific components of the CPU in the process. They should understand how factors such as clock speed, number of cores, size/type of cache affect the performance of the CPU (4.2.4).

Statements 4.2.5 and 4.2.6 focus on storage. Students should understand how data is stored on magnetic, optical and solid state storage devices. They need to understand the concept of 'cloud storage', be able to compare storing data in the cloud with local data storage and have an awareness of other contemporary secondary storage options. SAM Paper 1 Q3(a) is an extended open-response question that assesses students' ability to reason, using their understanding of contemporary secondary storage.

Statement 4.2.7 focuses on the need for and functions of embedded systems. SAM Paper 1 Q4(c) illustrates how this content could be assessed.

Section 4.3 covers logic. Students are expected to understand the operation of the AND, OR and NOT operators and be aware of the order of precedence rules for logic (brackets, NOT, AND, OR), so as to interpret and construct truth tables (4.3.1) and produce logic statements (4.3.2). SAM Paper 1 Q5(b)(i) and (ii) illustrate the type of question on this content students are likely to encounter.

Students do not need to be able to draw logic circuit diagrams.

Section 4.4 covers the characteristics and uses of different types of software - operating systems (4.4.1), utility software (4.4.2), simulations (4.4.3) and application software (4.4.4). SAM Paper 1 Q4(b) assesses students' knowledge and understanding of utility software.

Section 4.5 deals with high- and low-level programming languages. Students need to understand that every processor has its own set of machine code instructions and that a program written in any language other than machine code has to be translated before it can be executed. They should know how each of the three common types of translator work and the advantages and disadvantages of each (4.5.2). SAM Paper 1 Q7(c) illustrates one way in which this content may be assessed. Students are required to complete a table to demonstrate their knowledge of characteristics of translators.

There is no need for students to have any practical experience of writing/interpreting programs written in assembly language.

### Topic 5: Communication and the internet

Topic 5 focuses on networks and the internet. Section 5.1 covers network essentials. Students must understand the reasons why devices are connected on networks (5.1.1) and characteristics of different types of network and usage model (5.1.2). They need to understand how devices on a network communicate using wired and/or wireless connections and understand the benefits and risks of each of these forms of connectivity (5.1.3).

Students must know how network data speeds are calculated and be able to construct expressions to convert from one measure of data speed to another (5.1.4). They must also be able to write arithmetic expressions to show the time needed to transmit a data file of a specified size at a given speed.

Statement 5.1.5 deals with the need for network protocols and the role of specific protocols. Statement 5.1.6 hones in on the four-layer TCP/IP model. Students must be able to name the four layers, put them in the correct order and describe the main function of each. SAM Paper 1 Q2(d)(i) and (ii) illustrate how this content may be assessed.

The network topologies students are expected to learn about are listed in statement 5.1.7. They must understand the characteristics of each topology and be able to draw a diagram of it.

Statement 5.1.8 focuses on mobile communication standards. Students are expected to understand that mobile phones use radio waves to transmit and receive data and that the width of the frequency band used determines how much data throughput per second can be achieved. They should have an understanding of the evolution of wireless generations, from 1G through to 5G, and be able to explain why this progression has occurred.

Network security is the focus of Section 5.2. Students must understand the importance of network security and be able to explain techniques for securing networks (5.2.1). They should understand that deploying a combination of techniques provides a greater level of security than using just one.

Statement 5.2.2 focuses on security issues associated with cloud and other contemporary storage methods and should be taught in conjunction with statement 4.2.6.

Statement 5.2.3 lists the forms of cyber attack that students are expected to understand. Statement 5.2.4 deals with methods of identifying vulnerabilities and statement 5.2.5 deals with ways of protecting systems from cyber attack and developing resilient software systems. SAM Paper 1 Q8 assesses various aspects of cybersecurity.

Section 5.3 covers the internet and World Wide Web (WWW). Students are expected to understand the difference between the two. They should learn how the internet is structured (5.3.1), components of the WWW (5.3.2), the need for IP addressing standards (5.3.3) and the role of components used to access the internet (5.3.4).

There is no need for students to have any practical experience of using a mark-up language.

## Topic 6: The bigger picture

Topic 6 is concerned with the impact – positive and negative – of computing technology. Statement 6.1.1 focuses on the environment, in particular issues associated with health, energy use and resources.

Statement 6.1.2 considers the ethical impact, honing in on privacy, inclusion and professionalism. SAM Paper 1 Q5(a) assesses students' understanding of one aspect of this.

Statement 6.1.3 is designed to raise awareness of legal and ownership issues associated with computing technology – important considerations for budding computer scientists. Students should be aware of legislation related to computer misuse, copyright and patents, privacy and manufacture/disposal of computing equipment.

Four specific current and emerging trends in computing technology that students should be aware of are identified in statement 6.1.4. They are quantum computing, DNA computing, artificial intelligence and nanotechnology.

## Assessment guidance

### Assessment Objectives

The International GCSE in Computer Science has three Assessment Objectives.

		%
<b>AO1</b>	Demonstrate knowledge and understanding of the key principles of computer science	27.5%
<b>AO2</b>	Apply knowledge and understanding of key concepts and principles of computer science	42.5%
<b>AO3</b>	Analyse problems in computational terms: ■ to make reasoned judgements ■ to design, program, test, evaluate and refine solutions	30.0%

### Understanding AO1

In the context of AO1, 'demonstrate' means showing knowledge and understanding, for example by stating, describing or explaining an aspect of the subject content. This Assessment Objective tests knowledge recall.

*The ability to share data via networks has many advantages, but there are also disadvantages.*

- (a) *Describe the role of protocols in networks. (2 marks)*
- (b) *A patch for an instant messaging application is available.  
Explain why users of the application should apply the patch. (2 marks)*

In this example, Part (a) assesses students' knowledge of the role of protocols. To gain both marks students need to refer to the fact that protocols allow devices on a network to communicate with each other (1) by using a common set of rules (1).

Part (b) assesses understanding. Students need to explain what a patch does, i.e. fixes vulnerabilities in the code (1) and why this is important in the context of an instant messaging application, i.e. it protects the device running the application from cyber attacks (1).

## Understanding AO2

In the context of AO2, 'apply' means using knowledge and understanding of a particular aspect of the subject content in a particular context or contexts.

*A programmer is writing software for a new set-top receiver for satellite TV.*

*Explain why she should use a compiler rather than an interpreter to translate the code. (2 marks)*

This question assesses students' ability to apply their knowledge of translators. Their explanation must take the context into account, i.e. that a set-top box must process data quickly (1) making a compiler more appropriate since compiled code runs faster than interpreted code (1).

*A Caesar cipher is a simple encryption algorithm based on shifting.*

*Explain whether a shift of +7 followed by a shift of -2 is more secure than a single shift when applied to the word 'pink'.*

*You should include a diagram in your answer. (3 marks)*

To answer this question, students must apply their understanding of how the Caesar cipher algorithm works to demonstrate that the outcome of a single +5 shift applied to the word 'pink' is 'unsp' (1), that the outcome of applying a +7 shift to the word 'pink' is 'wpur', which in turn becomes 'unsp' when a -2 shift is subsequently applied (1), meaning that the double shift is no more secure than the single shift(1).

## Understanding AO3

In the context of AO3, 'analyse' involves decomposing a problem/issue so that it can be addressed using automated computation. There are two strands to this Assessment Objective:

- (a) make reasoned judgements
- (b) design, program evaluate and refine solutions.

'Reasoned judgements' are judgements based on logical thinking and application of knowledge and understanding.

*At the start of a race, cars are placed on a starting grid in a staggered pattern.*

*When a car makes a false start it breaks a light beam and this notifies the computer system. The race is stopped if a car makes a false start.*

*There is a maximum of 20 cars in a race. The programmer has chosen to use a one-dimensional array to store this data.*

*Assess how appropriate a one-dimensional array is as opposed to using separate variables when storing this data. (5 marks)*

This question requires students to demonstrate logical reasoning to reach the conclusion that using a one-dimensional array would be more appropriate than using 20 individual variables (1). One reason they might give is that using a separate variable for each car will make the code less efficient because it will require complex selection statements, whereas using an array will probably require just one loop (2); another that an array would require just one meaningful name, whereas 20 distinct names would be needed if variables were used (2).

There are three elements in the second strand of AO3. They are:

- a. design solutions
- b. program solutions
- c. evaluate and refine solutions.

Questions addressing strands 2(a) and 2(b) may use pseudocode, flow charts, written descriptions or program code.

*Each car has direction indicators, which are controlled by an indicator switch operated by the driver.*

- The car has two indicator lamps, one on the right and one on the left.*
- The indicator switch has three positions.*
- The middle position is the off position.*
- The up position indicates a right turn.*
- The down position indicates a left turn.*
- When a lamp is turned on, it flashes (on, off) in 0.5 second intervals, until the switch is moved to the off position.*

*Construct a flow chart to show this process. (6 marks)*

This question assesses students' ability to design logical solutions to problems.

*Open the file named Q06 in the code editor.*

*In file Q06, the names and years of birth of artists are stored in a two-dimensional data structure.*

*Labels for their work need to be created by joining the first letter of the artist's last name, the first letter of their first name, and their year of birth.*

*For example, a label for ('Andy', 'Warhol', 1928) would be 'WA1928'.*

*Write a program to:*

- process each artist to create a label*
- store all the labels in the data structure named 'theLabels'*
- display the labels for all the artists*
- find and display the name and year of birth of the youngest artist.*



***Your program should function correctly, even if 'theArtists' data structure has more, fewer, or different artists.***

*You **must** use the data structures in file **Q06**.*

*Save your amended code as **Q06FINISHED** with the correct file extension for the programming language. (20 marks)*

In this question, nine marks are allocated for designing a solution and a further for programming the solution. The remaining two marks are for applying knowledge and understanding to the task.

Evaluating and refining solutions can be targeted in relation to students' own or given solutions.

### Command words

Students should be encouraged to read questions carefully and – where appropriate – ensure that their responses take account of the context of the question.

They should be taught the meaning of the command words used in examination questions and the significance of the number of marks allocated.

Command words associated with AO1 include: identify, state, describe, name, give and calculate.

Command words associated with AO2 include: describe, explain, compare, construct and show.

Command words associated with AO3 include: deduce, devise, construct, amend and write.

A useful taxonomy of the command words used in examination questions can be found in Appendix 7 of the specification.

Students should be encouraged to employ relevant subject-specific terminology and key words in their answers and use examples to illustrate their response where appropriate.

### Relationship of Assessment Objectives to papers

As the table below illustrates, Paper 1: Principles of Computer Science primarily targets AO1 and AO2, whereas the focus of Paper 2: Application of Computational Thinking is application of knowledge and understanding and problem solving.

Unit number	Assessment Objective		
	AO1	AO2	AO3
Paper 1: Principles of Computer Science	21.5%	21%	7.5%
Paper 2: Application of Computational Thinking	6%	21.5%	22.5%
<b>Total</b>	27.5%	42.5%	30%

It's worth bearing in mind that together application of knowledge and understanding (AO2) and analysing problems in computational terms (AO3) have a weighting of 72.5%. This reflects the fact that, although Computer Science is an academic discipline with a defined body of knowledge, it is also a highly practical subject. Computational thinking needs to be developed, practised and reinforced on an ongoing basis throughout the course.

### Computer-related mathematics

Students' use of computer-related mathematics is assessed in context in both examination papers. Students may be asked to:

- convert between number bases e.g. SAM Paper 1, Q1(d) and Q2(b)(i)
- carry out arithmetic/logical calculations, e.g. SAM Paper 1 Q2(b)(ii) and Q5(b)(iii)
- construct and interpret expressions and logic statements, e.g. SAM Paper 1 Q2(a) and SAM Paper 2 Q5(b)
- convert between units of measurement, e.g. SAM Paper 1 Q6(c).

## Suggestions for teaching problem solving and programming

Contrary to what you might expect, it is not necessary, or appropriate, for students to use a computer in every computer science lesson. Group work, role play, using pencil and paper and 'show and tell' activities can help clarify topics that can later be reinforced through computer use. For example, students could act out a binary search. They can then discuss the algorithm used and try to implement it. CS Unplugged ([www.csunplugged.org](http://www.csunplugged.org)) and CS4FN ([www.cs4fn.org](http://www.cs4fn.org)) both have a range of resources for engaging students in active non-computer-based tasks.

Try dividing the class into groups who work together to solve problems. Introduce the concept of coaching other students within their group to ensure everyone finishes within the set time. Have competitions between groups. Teach students techniques for locating and correcting the mistakes they have made in their code.

Get students into the 'think, pair, share' way of working, i.e. think about the problem by themselves, discuss the problem in a pair, then share it with a group around them before asking you for help.

Provide code as images (print screen) to force students to type in the program code, rather than just use copy and paste. They will learn more by making mistakes and having to correct them.

Use two projectors – one to display code and the other information or questions.

Aim to create a classroom environment of mutual respect, and acceptance that people learn through their mistakes. Invite students to 'show and tell' a program – what does each line of code do, what are its good and bad features, how could it be improved?

### Arrangements for Paper 2 – the practical exam

Paper 2 is a practical, computer-based examination offered within a five-day window. It is three hours in duration.

#### **Prior to the examination**

The examination can be scheduled on any day within the five-day window. Centres with large entries may stagger examination sittings during the window. Exam sittings must be arranged to minimise the possibility of students colluding. Centres must produce a schedule showing the dates and times of each exam session to be held in the window.

Candidates must sit the whole examination in one go and may only sit the examination once during the five-day window.

Workstations must be arranged so as to prevent candidates from viewing each other's work.

Centres must set up a separate user area for each candidate sitting the examination. User areas must be allocated sufficient storage space to allow candidates to save their work.

The secure data files required for the examination will be made available on the Pearson website one week before the start of the examination window. The files should be downloaded, checked for compatibility with the software to be used and then copied into each candidate's secure user area. Candidates must not be given access to the data files in advance of their exam sitting.

Candidates may not bring portable storage media into the examination.

#### **During the examination**

Centres must ensure that appropriate hardware and software are available to candidates. No extra time can be allowed to compensate for slow machines or networks that run slowly. At least one invigilator should be conversant with the software and IT system to be used by candidates so that they are able to deal with any technical difficulties that may arise.

No scheduled breaks during the examination are allowed.

Candidates must answer all questions. Some questions will require written responses, whilst others will involve writing code.

Some questions needing a written response will require candidates to open a program code file in a suitable code editor and answer questions about it, see for example SAM Paper 2 Q1(a) and Q2(b). Others, like SAM Paper 2 Q4, will not make reference to any specific code.

SAM Paper 2 Q1(c)(ii), Q2(a) and Q3(c) are all examples of questions that require candidates to write code. Where this is the case, they must save their work in the specified folder in their user area. The file naming conventions specified in the question paper must be followed when saving work.

Candidates must not be able to save files produced during the examination in a central, unsecure location.

They must only have access to the files required for the examination. They may only use one of the three approved high-level programming languages (Python, C# or Java). The same language must be used throughout the examination.

Access to the internet during the examination is not permitted. Nor are candidates permitted to refer to textbooks or centre-prepared manuals during the examination. They may, however, use the offline help facilities available in code editors and/or software-specific manuals.

Communication between candidates at any time during the examination is not permitted.

### **After the examination**

Candidates' paper scripts must be returned to Pearson. Their digital responses must be uploaded to Pearson using the secure file transfer platform, or saved on an Optical or USB flash drive and sent to the allocated Pearson examiner.

Candidates' user areas should be removed at the end of the examination once their work has been copied to an appropriate storage medium.

If more than one examination sitting is held, any common user areas accessible to candidates must be cleared of all work saved during the examination immediately after each sitting.

### **Instructions for the Conduct of the Examination (ICE)**

An ICE document will be published on the Pearson website giving detailed instructions on how the examination is to be conducted.

### Delivery of the qualification – transferable skills

#### Why transferable skills?

Ensuring that International GCSE qualifications will help improve student outcomes through the acquisition of transferable skills, as well as subject content and skills, is a key aim for Pearson.

In recent years, higher education institutions and employers have consistently flagged the need for students to develop a range of transferable skills to enable them to respond with confidence to the demands of undergraduate study and the world of work.

Through our teaching materials and support offered we want to:

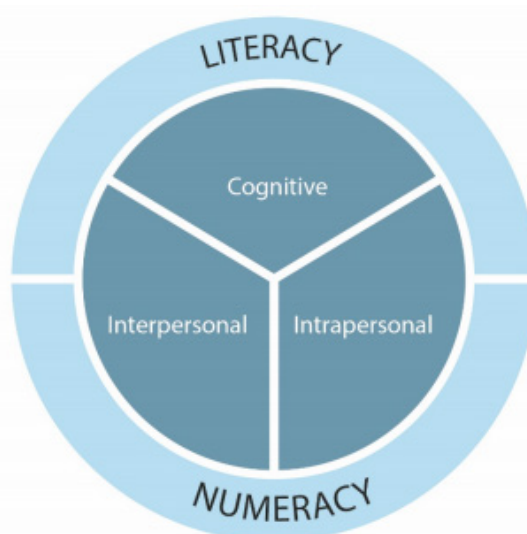
1. increase awareness of transferable skills that are already being assessed (for both students and teachers)
2. indicate where, for teachers, there are opportunities to teach additional skills that won't be formally assessed, but that would be of benefit to students.

#### What are transferable skills?

The Organisation for Economic Co-operation and Development (OECD) defines skills, or competencies, as 'the bundle of knowledge, attributes and capacities that can be learned and that enable individuals to successfully and consistently perform an activity or task and can be built upon and extended through learning.'<sup>[1]</sup>

To support the design of our qualifications, the Pearson Research Team selected and evaluated seven global 21st-century skills frameworks. Following on from this process, we identified the National Research Council's (NRC) framework <sup>[2]</sup> as the most evidence-based and robust skills framework, and have used this as a basis for our adapted skills framework.

The framework includes cognitive, intrapersonal skills and interpersonal skills.



[1] (OECD (2012), Better Skills, Better Jobs, Better Lives (2012):<http://skills.oecd.org/documents/OECDskillsStrategyFINALENG.pdf>)

[2] Koenig, J. A. (2011) Assessing 21st Century Skills: Summary of a Workshop, National Research Council)

## Suggested Resources

- BCS Glossary of Computing and ICT, 2013- 13th edition (ISBN 9781780171500)
- Articles about teaching coding: [www.edsurge.com/guide/teaching-kids-to-code](http://www.edsurge.com/guide/teaching-kids-to-code)
- The Computing at Schools group (CAS) – a great source of help and support including a huge bank of downloadable resources produced by practitioners: <http://www.computingschool.org.uk>
- Computer Science Unplugged – a collection of free learning activities that teach computer science through engaging games and puzzles: <http://csunplugged.org>
- CS4FN (Computer Science for Fun): a magazine for students available in digital and paper form: <http://www.cs4fn.org>
- Peyton-Jones, S. 2014. Teaching Creative Computer Science: <http://tedxexeter.com/2014/05/06/simon-peyton-jones-teaching-creative-computer-science>
- Bentley, P.J. 2012. Digitized: The Science of Computers and How it Shapes our World (Oxford University Press)
- Berners-Lee, T. 'Answers for young people': [www.w3.org/people/Berners-Lee/Kids.html](http://www.w3.org/people/Berners-Lee/Kids.html)
- Barr, D, Harrison J, and Conery L. Computational Thinking: A Digital Age Skill for Everyone: <http://www.iste.org/docs/learning-and-leading-docs/march-2011-computational-thinking-ll386.pdf>
- There are a number of introductory computer science MOOCs provided by HE consortia, such as Coursera, Edx and FutureLearn
- Subject advisor Tim Brady, is a direct and personal source of help and support. Email him at [TeachingICT@pearson.com](mailto:TeachingICT@pearson.com)

## Python resources

- Python (free open source software): [www.python.org/getit/](http://www.python.org/getit/)
- Official Python documentation (also available through help in IDLE): [www.docs.python.org/3/](http://www.docs.python.org/3/)
- Python Summer School from Anglia Ruskin University is an excellent resource with videos and programming challenges: <http://www.pythonschool.net/>
- 'Python code for kids' is a clearly written summary of the Python language written in accessible language: [www.pythondictionary.code-it.co.uk/](http://www.pythondictionary.code-it.co.uk/)
- 'Python in 10 minutes' is a quick run through of the basic concepts: [www.korokithakis.net/tutorials/python/](http://www.korokithakis.net/tutorials/python/)
- 'Too much doing, not enough understanding' is a 20-minute video produced by Quintin Cutts (one of the author's of Haggis) containing useful ideas and concepts on how to teach programming: [www.youtube.com/watch?v=Pim4aYfiZiY](http://www.youtube.com/watch?v=Pim4aYfiZiY)

- The Khan Academy has some excellent materials designed for self-directed study: <https://www.khanacademy.org>
- Codecademy – interactive programming courses in a number of languages, including Python and Java: <https://www.codecademy.com>
- Grok Learning – an introduction to Python programming: <https://groklearning.com>
- Sentdex Youtube channel - collection of video tutorials: <https://www.youtube.com/user/sentdex/videos>
- ‘Think Python How to Think Like a Computer Scientist’ is a free online book with programming challenges at the end of each chapter: [www.greenteapress.com/thinkpython/thinkpython.pdf](http://www.greenteapress.com/thinkpython/thinkpython.pdf)
- ‘A Byte of Python’ is an excellent online book, though it does not use IDLE as the editor: [www.swaroopch.com/notes/Python/](http://www.swaroopch.com/notes/Python/)
- Python Programming for the Absolute Beginner, M. Dawson (published Course Technology 2010) (ISBN 9781435455009) is an excellent book with clear explanations of each bit of code together with free downloads and example games built into each chapter
- Python for Kids: A Playful Introduction to Programming, J. Briggs (published No Starch Press 2013) (ISBN 9781593274078)
- Invent your own computer games with Python, Al Sweigart (4th edition due out in December 2016) (ISBN 9781593277956 ) uses source code for games to teach programming concepts

### Cybersecurity

- The Cybersecurity Challenge Schools Programme – useful resources to help teach section 5.2: <https://cybersecuritychallenge.org.uk/education/schools>

### Hardware

- Most school computer labs are equipped with the hardware and software necessary to deliver this qualification. See [www.computingatschool.org.uk/data/uploads/CASInfrastructure.pdf](http://www.computingatschool.org.uk/data/uploads/CASInfrastructure.pdf) for detailed requirements.
- Taking apart old computers can aid students’ understanding of hardware. Be aware of health and safety considerations and ensure you undertake a risk assessment for such activities.
- Small computers such as the Raspberry Pi enable students to see individual hardware components, offer access to operating system functionality, such as the command line, that might be restricted in school computer labs and – with the addition of a router and some cables – the basics of networking to be taught. For further details see ‘The Raspberry Pi Education Manual’ (Computing at School, 2012): [http://pi.cs.man.ac.uk/download/Raspberry\\_Pi\\_Education\\_Manual.pdf](http://pi.cs.man.ac.uk/download/Raspberry_Pi_Education_Manual.pdf).

### Software

- Audacity (free open source, cross-platform software for recording and editing sounds): [www.audacity.sourceforge.net](http://www.audacity.sourceforge.net)



## Course planner

### Planning and delivering a linear course

The International GCSE in Computer Science has a notional time allocation of 120-140 guided learning hours.

We recommend a minimum of two one-hour lessons a week (or equivalent) to cover the content adequately, develop and practise computational thinking skills and prepare students for the assessments. This is based on the assumption that students have had some previous experience of computer science.

In this plan the course has been split into six terms. Practical problem solving and programming (Topics 1 and 2) are taught in parallel throughout the course to ensure that students have the necessary computational thinking and programming know-how. Programming challenges are used to make connections between theory and practice.

The programming language used is Python 3.0. Centres can of course adapt the materials to use one of the other approved languages, namely C# or Java.

It is important to leave sufficient time for revision in the second year, particularly to revisit topics studied in the first year.

## A Getting started for teachers

Year of study	Term	Topic	GLH	Notes
1	1	Problem solving and programming (Topics 1 and 2)	10	2 lessons a week for first 5 weeks
1	1	Problem solving and programming (Topics 1 and 2)	10	1 lesson per week for remaining 10 weeks of term
1	1	Hardware, software and programming languages (4.1, 4.2, 4.4, 4.5)	5	Overview of what a computer is
1	1	Binary representation of numbers (3.1)	5	
1	2	Problem solving and programming	8	
1	2	Networks (5.1)	7	
1	2	Data representation (3.2)	2	Text (3.2.1, 3.2.4)
1	2	Logic (4.3)	3	
1	2	The bigger picture (6.1)	4	Environmental impact of computing (6.1.1)
1	3	Problem solving and programming (Topics 1 and 2)	12	
1	3	Data representation (3.2)	4	Bitmap images and sound (3.2.2, 3.2.3, 3.2.4)
1	3	Hardware (4.2)	4	4.2.1, 4.2.2, 4.2.4, 4.2.4, 4.2.7
1	3	Network security (5.2)	6	
1	3	The bigger picture (6.1)	2	Legal impact (6.1.3)
2	1	Problem solving and programming (Topics 1 and 2)	12	
2	1	Data storage and compression (3.3)	4	
2	1	Secondary storage (4.2)	5	
2	1	Encryption (3.4)	5	
2	1	The internet and World Wide W+eb (5.3)	4	
2	2	Role of components used to access the internet (5.3.4)	1	
2	2	Mobile communication standards (5.1.8)	1	
2	2	Current and emerging technologies	4	
2	2	The bigger picture (6.1)	2	Ethical impact of computing (6.1.2)
2	3	Revision	-	

## Student guide

### Why study the Pearson Edexcel International GCSE in Computer Science?

This course will enable you to:

- understand and apply the fundamental principles and concepts of computer science, including abstraction, decomposition, logic, algorithms and data representation
- analyse problems in computational terms through practical experience of solving such problems, including designing, writing and debugging programs
- think creatively, innovatively, analytically, logically and critically
- understand the components that make up digital systems, and how they communicate with one another and with other systems
- understand the impacts of digital technology to the individual and to wider society
- apply mathematical skills relevant to computer science.

### What do I need to know, or be able to do, before taking this course?

We recommend that students are able to read and write in English at Level B2 of the Common European Framework of Reference for Languages, otherwise there are no prior learning requirements for this qualification.

### Is this the right subject for me?

Have a look at our qualification overview to get an idea of what's included in this qualification. Then, why not get in touch with our student services, [students@pearson.com](mailto:students@pearson.com), to discuss any outstanding questions you might have?

You could also have a look at <http://qualifications.pearson.com/en/campaigns/pearsonqualifications-around-the-world.html#tab-Edexcel> to find out what students and education experts around the world think about our qualifications.

### How will I be assessed?

Through 100% examination: one written paper and one practical paper. You will need access to a computer and printer to complete this qualification, refer to our ICE (Instructions for the Conduct of the Examination) document on the subject page for more information.

**What can I do after I've completed the course?** You can progress onto further study of Computer Science or Computing at AS and A Levels, and then onto higher education.

### What next?

Talk to your subject teacher at school or college for further guidance, or if you are a private candidate you should visit <http://qualifications.pearson.com/en/support/support-for-you/students.html>

For information about Edexcel, BTEC or LCCI qualifications  
visit [qualifications.pearson.com](http://qualifications.pearson.com)

Edexcel is a registered trademark of Pearson Education Limited

Pearson Education Limited. Registered in England and Wales No. 872828  
Registered Office: 80 Strand, London WC2R 0RL  
VAT Reg No GB 278 537121

Getty Images: Alex Belmonlinsky