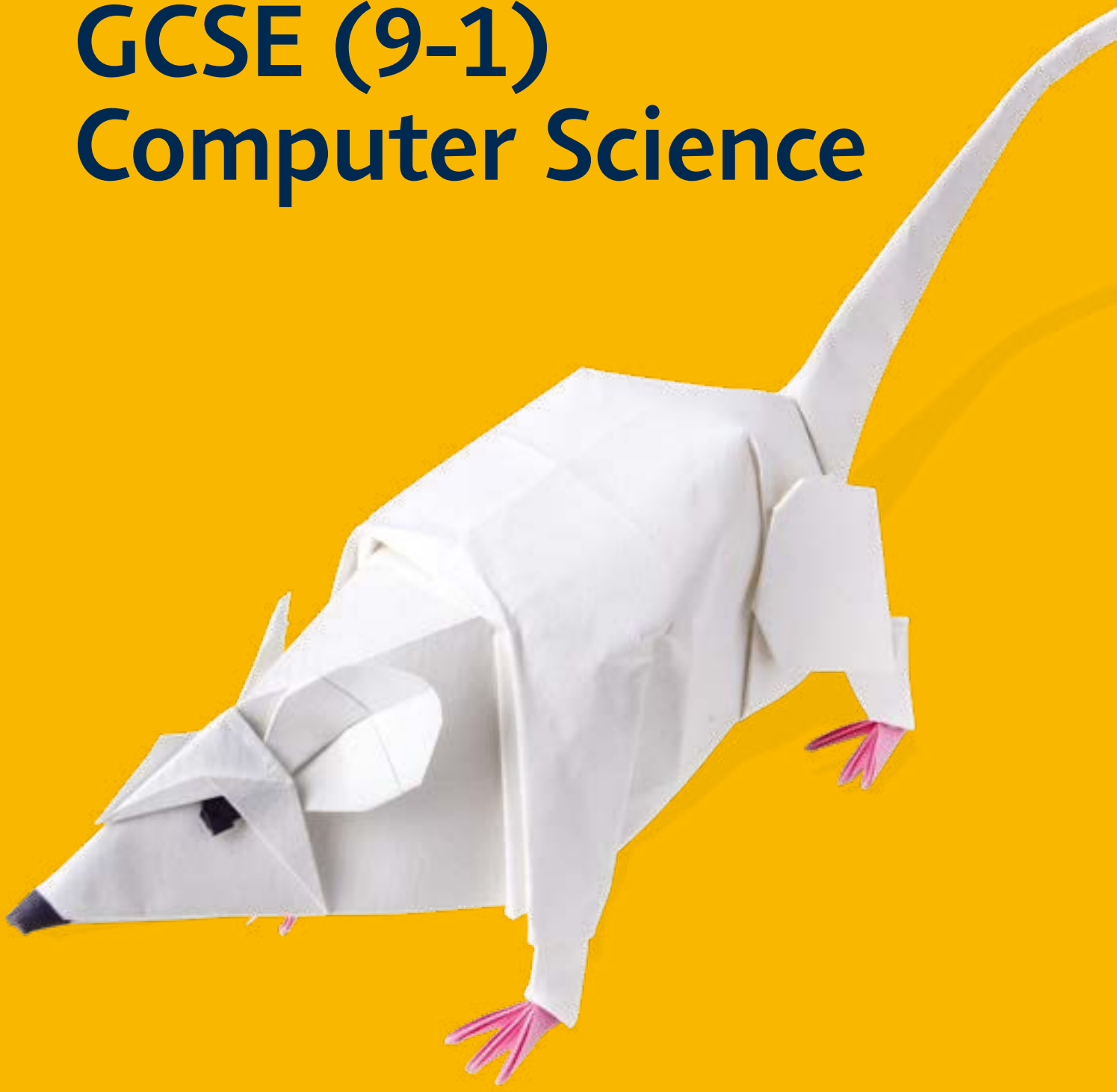
Pearson Edexcel

GCSE (9-1) Computer Science



Specification

Pearson Edexcel Level 1/Level 2 GCSE (9-1) in Computer Science (1CP1)

First teaching from September 2016

First certification from 2018

Issue 4

Pearson Edexcel Level 1/Level 2 GCSE in Computer Science

In February 2018 we made changes to our specification because the programming project no longer contributes to the final grade. These changes were communicated in Issue 3. We have now made further changes to the programming project, where we have relaxed the rules around the conditions under which the programming project can be taken. These changes can be found in this issue (Issue 4).

The changes to Issue 3 were only made three months ago therefore we have kept those summary of changes as well.

Summary of Pearson Edexcel Level 1/Level 2 GCSE in Computer Science specification Issue 3 changes

Summary of changes made between previous issue and this current issue	Page number
The NEA project is no longer being assessed in this qualification, however a computer programming project must be completed by all students although it will not contribute to their final grade. The following changes have been made to the specification to reflect this change.	
Information about the programming project has been updated.	4, 14-19
The mark grids and the instructions on their use for the NEA project have been removed from the specification.	N/A
Teachers can now provide feedback to students at all stages of the programming project.	18
The teacher and students must sign an amended programming project authentication form.	19, 36
The Head of Centre must sign an amended Head of Centre declaration form.	19, 37
Components 1 and 2 are each weighted at 50 percent.	4, 5, 21
The assessment objective weightings for components 1 and 2 have changed.	21
Synoptic assessment will be assessed in Component 2.	21
The rules around re-sitting the qualification has been added.	22

Summary of Pearson Edexcel Level 1/Level 2 GCSE in Computer Science specification Issue 4 changes

Summary of changes made between previous issue and this current issue	Page number
The qualification at a glance, for component 1, has been amended as the pseudocode booklet is no longer used with component 1.	4
The conditions under which the programming project should be taken have been updated to follow the guidance from Ofqual. This includes information about collaboration and online and offline help.	14, 18

Summary of changes made between previous issue and this current issue	Page number
There will now be a bank of project briefs that can be used for the programming project, which will be valid for the lifetime of the qualification.	14
Information about *Appendix 1: Pseudocode command set* has been amended as this is now only used for component 2 and the programming project.	29
Appendix 3: Guide for offline help has been removed from the specification as the students are now allowed access to the internet during their programming project.	N/A

If you need further information on these changes or what they mean, contact us via our website at: qualifications.pearson.com/en/support/contact-us.html.

`

Contents

1 Introduction

Why choose the Edexcel GCSE Computer Science?

We've listened to feedback from all parts of the computer science subject community, including higher education. We've used this opportunity of curriculum change to redesign the qualification to reflect the importance of computation in the modern world today and how it will do so in the future – a qualification that enables your students to apply themselves and give them the skills to succeed in their chosen pathway.

Clear and coherent structure – our qualification has a straightforward structure with six comprehensive topic areas, assessed through two externally-examined components.

Continuous progression – students are introduced to core principles of computer science and develop skills in problem solving and computational thinking. This builds on skills learned in Key Stages 1 to 3 in Computer Science/IT while also ensuring that students new to the subject are supported appropriately. Following on from more visual programming environments, programming skills are further developed using high-level textual programming languages.

Helps develop an understanding of computer science methods in the real world – students will decompose and model aspects of real-world situations, and as a result be able to design, build and test a fully-programmed solution to a problem.

Provides a real study of computation – the new specification enables students to apply computational thinking in context, which is evidenced through examined assessment and a programming project. This is supported by comprehensive coverage of computer science principles.

Reflects today's global world – students develop knowledge and understanding of how technology can be used to help proactively with current issues that impact on modern society, preparing them for their next steps in today's global world.

Supports progression to Key Stage 5 – the content will enable students to move on to GCE Computer Science or to BTEC Technical Levels in Computing with a clear knowledge and understanding of the subject.

Develops transferable skills for progression to higher education – students will develop 'underpinning' concepts, which are useful in many subjects, for example mathematics, science, engineering. The rigorous approach to the subject will facilitate a smooth transition to the next level of study.

Supporting you in planning and implementing this qualification

Planning

- Our **Getting Started** guide gives you an overview of the new GCSE qualification to help you to get to grips with the changes to content and assessment and to help you understand what these changes mean for you and your students.
- We will give you an editable **course planner** and **scheme of work** that you can adapt to suit your department.
- **Our mapping documents** highlight key differences between the new and the 2013 qualification.

Teaching and learning

There will be lots of free teaching and learning support to help you deliver the new qualification, including:

- suggested resource lists
- case studies and activities
- a student guide
- materials for your options evenings.

Preparing for exams

We will also provide a range of resources to help you prepare your students for the assessments, including marked exemplars of student work with examiner commentaries.

ResultsPlus

ResultsPlus provides the most detailed analysis available of your students' exam performance. It can help you identify the topics and skills where further learning would benefit them.

Get help and support

Our Subject Advisor service, led by Tim Brady, and online community will ensure you receive help and guidance from us and that you can share ideas and information with other teachers. You can sign up to receive e-newsletters from Tim Brady to keep up to date with qualification updates and product and service news.

Subject Advisor contact details

UK: 0844 372 2186
International: +44 (0) 207 010 2161
Email: TeachingComputerScience@pearson.com
Twitter: @Pearson_CS

Learn more at qualifications.pearson.com

Qualification at a glance

Content and assessment overview

The Pearson Edexcel Level 1/Level 2 GCSE (9–1) in Computer Science consists of two externally-examined components.

Students must also undertake a programming project. This project will not count towards their overall grade for the qualification but a single project must be completed to fulfil the requirements of the qualification. Centres must submit all projects to Pearson by 15th May in the same year as that in which students sit the examined components.

Component 1: Principles of Computer Science (*Component code: 1CP1/01)
Written examination: 1 hour and 40 minutes ***50% of the qualification*** ***80 marks***
Content overview This component will assess all topics. • Understanding of what algorithms are, what they are used for and how they work; ability to interpret, amend and create algorithms. • Understand the requirements for writing program code. • Understanding of binary representation, data representation, data storage and compression, encryption and databases. • Understanding of components of computer systems; ability to construct truth tables, produce logic statements. • Understanding of computer networks, the internet and the worldwide web. • Awareness of emerging trends in computing technologies, and the impact of computing on individuals, society and the environment, including ethical, legal and ownership issues.
Assessment overview This paper consists of multiple-choice, short open-response, open-response and extended open-response answer questions. All questions are mandatory.

*See *Appendix 7: Codes* for a description of this code and all other codes relevant to this qualification.

Component 2: Application of Computational Thinking (Component code: 1CP1/02)
Written examination: 2 hours ***50% of the qualification*** ***80 marks***
Content overview The main focus of this component is: • understanding what algorithms are, what they are used for and how they work; ability to interpret, amend and create algorithms • understanding how to develop program code and constructs, data types, structures, input/output, operators and subprograms. This component may also draw on: • understanding binary representation, data representation, data storage and compression, encryption and databases • understanding components of computer systems; ability to construct truth tables, produce logic statements and read and interpret pseudocode • understanding computer networks, the internet and the worldwide web • awareness of emerging trends in computing technologies, the impact of computing on individuals, society and the environment, including ethical, legal and ownership issues.
Assessment overview This paper is based on a scenario. It consists of short open-response, open-response and extended open-response answer questions. All questions are mandatory.

2 Subject content

Qualification aims and objectives

The aims and objectives of this qualification are to enable students 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 impact of digital technology on the individual and on wider society
- apply mathematical skills relevant to computer science.

Content

Topic 1: Problem solving

Students are expected to develop a set of computational thinking skills that enable them to understand how computer systems work, and to design, implement and analyse algorithms for solving problems.

Students should be given repeated opportunities to tackle computational problems of various sorts, including some substantial problem-solving tasks.

Students will be expected to use the pseudocode listed in *Appendix 1* and the flowchart symbols shown in *Appendix 2*.

Subject content	Students should:	
1.1 Algorithms	1.1.1	understand what an algorithm is, what algorithms are used for and be able to interpret algorithms (flowcharts, pseudocode, written descriptions, program code)
	1.1.2	understand how to create an algorithm to solve a particular problem, making use of programming constructs (sequence, selection, iteration) and using appropriate conventions (flowchart, pseudocode, written description, draft program code)
	1.1.3	understand the purpose of a given algorithm and how an algorithm works
	1.1.4	understand how to determine the correct output of an algorithm for a given set of data
	1.1.5	understand how to identify and correct errors in algorithms
	1.1.6	understand how to code an algorithm in a high-level language
	1.1.7	understand how the choice of algorithm is influenced by the data structures and data values that need to be manipulated
	1.1.8	understand how standard algorithms (bubble sort, merge sort, linear search, binary search) work
	1.1.9	be able to evaluate the fitness for purpose of algorithms in meeting specified requirements efficiently using logical reasoning and test data
1.2 Decomposition and abstraction	1.2.1	be able to analyse a problem, investigate requirements (inputs, outputs, processing, initialisation) and design solutions
	1.2.2	be able to decompose a problem into smaller sub-problems
	1.2.3	understand how abstraction can be used effectively to model aspects of the real world
	1.2.4	be able to program abstractions of real-world examples

Topic 2: Programming

Learning to program is a core component of a computer science course. Students should be competent at designing, reading, writing and debugging programs. They must be able to apply their skills to solve real problems and produce robust programs.

Students should be given repeated opportunities to develop and practise their programming skills.

Students will be expected to use the pseudocode listed in *Appendix 1*, the flowchart symbols shown in *Appendix 2* and at least one of the programming languages listed in the Programming project section.

Subject content	Students should:	
2.1 Develop code	2.1.1	be able to write programs in a high-level programming language
	2.1.2	understand the benefit of producing programs that are easy to read and be able to use techniques (comments, descriptive names (variables, constants, subprograms), indentation) to improve readability and to explain how the code works
	2.1.3	be able to differentiate between types of error in programs (logic, syntax, runtime)
	2.1.4	be able to design and use test plans and test data (normal, boundary, erroneous)
	2.1.5	be able to interpret error messages and identify, locate and fix errors in a program
	2.1.6	be able to determine what value a variable will hold at a given point in a program (trace table)
	2.1.7	be able to determine the strengths and weaknesses of a program and suggest improvements
2.2 Constructs	2.2.1	understand the structural components of a program (variable and type declarations, command sequences, selection, iteration, data structures, subprograms)
	2.2.2	be able to use sequencing, selection and iteration constructs in their programs
2.3 Data types and structures	2.3.1	understand the need for, and understand how to use, data types (integer, real, Boolean, char)
	2.3.2	understand the need for, and understand how to use, data structures (records, one-dimensional arrays, two-dimensional arrays)
	2.3.3	understand the need for, and how to manipulate, strings
	2.3.4	understand the need for, and how to use, variables and constants
	2.3.5	understand the need for, and how to use, global and local variables when implementing subprograms

Subject content	Students should:	
2.4 Input/output	2.4.1	understand how to write code that accepts and responds appropriately to user input
	2.4.2	understand the need for, and how to implement, validation
	2.4.3	be able to write code that reads/writes from/to a text file
2.5 Operators	2.5.1	understand the purpose of, and how to use, arithmetic operators (add, subtract, divide, multiply, modulus, integer division)
	2.5.2	understand the purpose of, and how to use, relational operators (equal to, less than, greater than, not equal to, less than or equal to, greater than or equal to)
	2.5.3	understand the purpose of, and how to use, logic operators (AND, OR, NOT)
2.6 Subprograms	2.6.1	understand the benefits of using subprograms and be able to write code that uses user-written and pre-existing (built-in, library) subprograms
	2.6.2	understand the concept of passing data into and out of subprograms (procedures, functions)
	2.6.3	be able to create subprograms that use parameters

Topic 3: Data

Computers are able to store and manipulate large quantities of data. They use binary to represent different types of data.

Students are expected to learn how different types of data are represented in a computer.

Subject content	Students should:	
3.1 Binary	3.1.1	understand that computers use binary to represent data (numbers, text, sound, graphics) and program instructions
	3.1.2	understand how computers represent and manipulate numbers (unsigned integers, signed integers (sign and magnitude, two's complement))
	3.1.3	be able to convert between binary and denary whole numbers (0–255)
	3.1.4	understand how to perform binary arithmetic (add, shifts (logical and arithmetic)) and understand the concept of overflow
	3.1.5	understand why hexadecimal notation is used and be able to convert between hexadecimal and binary
3.2 Data representation	3.2.1	understand how computers encode characters using ASCII
	3.2.2	understand how bitmap images are represented in binary (pixels, resolution, colour depth)
	3.2.3	understand how sound, an analogue signal, is represented in binary
	3.2.4	understand the limitations of binary representation of data (sampling frequency, resolution) when constrained by the number of available bits
3.3 Data storage and compression	3.3.1	understand how to convert between the terms 'bit, nibble, byte, kilobyte (KB), megabyte (MB), gigabyte (GB), terabyte (TB)'
	3.3.2	understand the need for data compression and methods of compressing data (lossless, lossy) and that JPEG and MP3 are examples of lossy algorithms
	3.3.3	understand how a lossless, run-length encoding (RLE) algorithm works
	3.3.4	understand that file storage is measured in bytes and be able to calculate file sizes
3.4 Encryption	3.4.1	understand the need for data encryption
	3.4.2	understand how a Caesar cipher algorithm works
3.5 Databases	3.5.1	understand the characteristics of structured and unstructured data
	3.5.2	understand that data can be decomposed, organised and managed in a structured database (tables, records, fields, relationships, keys)

Topic 4: Computers

Students must be familiar with the hardware and software components that make up a computer system and recognise that computers take many forms from embedded microprocessors to distributed clouds.

Subject content	Students should:	
4.1 Machines and computational modelling	4.1.1	understand the input-process-output model
4.2 Hardware	4.2.1	understand the function of the hardware components of a computer system (CPU, main memory, secondary storage, input and output devices) and how they work together
	4.2.2	understand the function of different types of main memory (RAM, ROM, cache)
	4.2.3	understand the concept of a stored program and the role of components of the CPU (control unit (CU), arithmetic/logic unit (ALU), registers, clock, address bus, data bus, control bus) in the fetch-decode-execute cycle (the Von Neumann model)
	4.2.4	understand how data is stored on physical devices (magnetic, optical, solid state)
	4.2.5	understand the concept of storing data in the 'cloud' and other contemporary secondary storage
	4.2.6	understand the need for embedded systems and their functions
4.3 Logic	4.3.1	be able to construct truth tables for a given logic statement (AND, OR, NOT)
	4.3.2	be able to produce logic statements for a given problem
4.4 Software	4.4.1	know what an operating system is and how it manages files, processes, hardware and the user interface
	4.4.2	understand the purpose and functions of utility software (managing, repairing and converting files; compression; defragmentation; backing up; anti-virus, anti-spyware)
	4.4.3	understand how software can be used to simulate and model aspects of the real world
4.5 Programming languages	4.5.1	understand what is meant by high-level and low-level programming languages and understand their suitability for a particular task
	4.5.2	understand what is meant by an assembler, a compiler and an interpreter when translating programming languages and know the advantages and disadvantages of each.

Topic 5: Communication and the internet

Computer networks and the internet are now ubiquitous. Many computer applications in use today would not be possible without networks. Students should understand the key principles behind the organisation and of computer networks. Ideally, they should be able to experiment by setting up a simple network.

Subject content	Students should:	
5.1 Networks	5.1.1	understand why computers are connected in a network
	5.1.2	understand the different types of networks (LAN, WAN) and usage models (client-server, peer-to-peer)
	5.1.3	understand wired and wireless connectivity
	5.1.4	understand that network data speeds are measured in bits per second (Mbps, Gbps)
	5.1.5	understand the role of and need for network protocols (Ethernet, Wi-Fi, TCP/IP, HTTP. HTTPS, FTP, email (POP3, SMTP, IMAP))
	5.1.6	understand that data can be transmitted in packets using layered protocol stacks (TCP/IP)
	5.1.7	understand characteristics of network topologies (bus, ring, star, mesh)
5.2 Network security	5.2.1	understand the importance of network security and be able to use appropriate validation and authentication techniques (access control, physical security and firewalls)
	5.2.2	understand security issues associated with the 'cloud' and other contemporary storage
	5.2.3	understand different forms of cyberattack (based on technical weaknesses and behaviour) including social engineering (phishing, shoulder surfing), unpatched software, USB devices, digital devices and eavesdropping
	5.2.4	understand methods of identifying vulnerabilities including penetration testing, ethical hacking, commercial analysis tools and review of network and user policies
	5.2.5	understand how to protect software systems from cyber attacks, including considerations at the design stage, audit trails, securing operating systems, code reviews to remove code vulnerabilities in programming languages and bad programming practices, modular testing and effective network security provision
5.3 The internet and the world wide web	5.3.1	understand what is meant by the internet and how the internet is structured (IP addressing, routers)
	5.3.2	understand what is meant by the world wide web (WWW) and components of the WWW (web server URLs, ISP, HTTP, HTTPS, HTML)

Topic 6: The bigger picture

Students should be aware of the influence of computing technology and recognise that computing has an impact on nearly every aspect of the world in which they live.

Subject content	Students should:
6.1 Emerging trends, issues and impact	6.1.1 understand the environmental impact of technology (health, energy use, resources) on society
	6.1.2 understand the ethical impact of using technology (privacy, inclusion, professionalism) on society
	6.1.3 understand the legal impact of using technology (intellectual property, patents, licensing, open source and proprietary software, cyber-security) on society

Programming project

Overview

The GCSE in Computer Science requires each student to undertake a programming project, where they will develop a computer program. The project is not assessed and does not contribute towards the final grade.

The purpose of the project is to enable students to develop skills in:

- analysing the requirement of a problem
- designing and then implementing a programming solution
- testing, refining and evaluating their solution.

We will provide a project brief that describes a problem that students will need to solve by developing a computer program.

The project must use one programming language from the following:

- Python
- Java
- Pascal/Object Pascal
- Visual Basic.NET
- C-Derived (C, C++, C#).

The project must be taken under the conditions specified in the *Security* section.

Centres must ensure that students undertake the project within 20 hours of timetabled classroom time.

Students will submit a fully-working program with a report showing how the program was developed, tested and evaluated.

Programming project requirements

The project will require students to create a program that will include the following:

- data input and storage
- processing data
- producing output based on processed data.

Students must submit a report of the development, testing and evaluation of their program together with their program.

The report is also part of the project, not just the program.

Carrying out the programming project

Project brief

Teachers should give each student a copy of the project brief.

There will be a bank of project briefs published on the Pearson website, which are valid for the lifetime of the qualification. Students must complete all activities within one project brief. Centres should ensure that students have the necessary knowledge and skills required before undertaking the programming project.

If a data file is required to complete the project this will be provided by Pearson.

Stage 1: Analysis

The purpose of the analysis stage is to identify the requirements of the problem and what the proposed solution will do to meet the requirements.

The analysis tasks are to:

- analyse the given problem and identify the requirements of the program that will be designed, implemented and tested
- decompose the problem into manageable sub-problems, with an explanation of each.

An introduction to the problem, in prose, will demonstrate an understanding of abstraction. The decomposed list of requirements can be presented in prose or as a bulleted list, with each requirement clearly identified.

Decomposition requires choices to be made, in this case by breaking the given problem down into sub-problems that will be designed and implemented later. A description of what each sub-problem will do is required, it can be presented in prose or as a bulleted list. An explanation, in prose, of the reasons why the decomposition submitted is the most appropriate to meet requirements must also be included.

Report content for analysis

For this stage, the report should include:

- a short introduction to the problem
- a list of the requirements of the problem that will be programmed
- decomposition of the problem into sub-problems, including
  - a short description of what each of the sub-problems will do
  - a short explanation of the reasoning behind the decomposition submitted.

Stage 2: Design

The purpose of the design stage is to describe what has to be done when implementing the solution and to suggest an appropriate strategy to test the solution.

2.1 Solution design

An algorithm or algorithms should be designed that meet/s the requirements of the problem using appropriate conventions (flowchart, pseudocode, written description). Program code using the chosen language must **not** be included in the design solution.

The algorithm(s) should:

- show detailed decomposition into sub-problems and how they link together (if appropriate)
- demonstrate clear abstraction (for example by including parameterisation, links between components)
- include inputs, processes and outputs
- use all three basic programming constructs: sequence, selection and iteration.

Report content for solution design

For this stage, the report should include:

- the algorithm(s)
- any refinements to the design identified during implementation, with reasons for the refinements.

2.2 Test strategy and initial test plan

A test strategy for the solution should be devised based on meeting the requirements of the problem. The test strategy should explain the approaches that the student will use when testing the program. The proposed strategy should be followed when creating an initial test plan, this must be completed before implementation and will be updated before the program is actually tested.

The test strategy should be presented in prose.

An example of a table that could be used for the initial test plan is shown below. When constructing test data for the initial test plan, normal data is data that the program will accept. Erroneous data is inaccurate data that the program will not accept. Boundary data is typically on the 'edge' of a range of possible values that may or may not be accepted. Not all tests may require data entry.

Report content for test strategy and initial test plan

For this stage, the report should include:

- the test strategy
- the initial test plan, using the headings shown, with all four columns completed. It should be labelled 'Initial Test Plan'.

Test no	Purpose of the test	Test data	Expected result

Stage 3: Implementation

The purpose of this stage is to program the solution to the problem.

It may be that amendments to the original design solution become apparent during this stage, these refinements should be implemented and documented as additions to the design and in the program code by using comments as descriptors. A refinement can include many things, for example a more efficient way of programming a sub-problem or an additional component that has not been previously considered but which will add to the functionality of the solution. Refinements will be considered in stage 4.

The two sub-stages in implementation (3.1 and 3.2) will happen concurrently as the solution develops.

Report content for implementation

For this stage, the report should include:

- a copy of the program code; any refinements should be noted as comments in the final program.
- screenshots demonstrating effective use of debugging skills to correct errors.

3.1 Implementing the design

The design algorithm(s), as abstract decomposition, need(s) to be translated into the high-level programming language that has been chosen. This process requires applying an understanding of programming concepts when using this programming language during the implementation.

3.2 Building the solution

The final programmed solution should address all the requirements listed in the analysis stage. It should be functional, which will be aided by the choice of appropriate programming concepts. Computing techniques (comments, naming conventions, indentation) should be used to improve readability and aid understanding.

If errors arise during this stage, the language's debugging tools and hand-tracing should be used, as appropriate.

NOTE: formal, recorded testing is carried out in stage 4.

Stage 4: Testing, refining and evaluation

The purpose of this stage is to show that the final program solution has been tested, along with any refinements, and that the solutions have been evaluated against the original requirements.

Columns to show the 'Actual result' and 'Action needed/comments' should be added to the initial test plan and completed when each test is run. An example is shown. Any errors should have 'Action Needed/Comments' entries. An attempt should be made to correct and retest all errors.

Report content for testing, refining and evaluation

For this stage, the report should include:

- the updated and complete Test Plan (labelled 'Final Test Plan')
- the evaluation.

Test no	Purpose of the test	Test data	Expected result	Actual result	Action needed/comments

Tests for any refinements completed in the design and/or implementation stages should be added to the end of the test plan and carried out.

The evaluation should include a thorough and critical evaluation of the program. This should include how successfully the program meets each of the original requirements and the reason for adding refinements to the solution.

Security

The project must be carried out in a supervised environment.

There is no requirement for students to be set up with special programming project accounts. However, students can only work on their program within the 20 timetabled hours.

Students will be carrying out their project over several separate sessions. At the end of each session they must save their work.

When working on the project, students should have access to the program (for example, Python) that they are using to develop their project and to a word processor. They are permitted to access the internet or intranet as required but they must reference any material they take directly from these sites and use in their program.

Students must not post their solutions onto the internet.

Collaboration

Students must be supervised. The project be the students own, but students should be encouraged to discuss ideas with others whilst working independently. The project submitted must be the students own work as authenticated by the teacher.

Online and offline help

Students are not expected to remember the different features of the programming language that they are using to develop their project. Consequently, they can be given a syntax guide for the language that they are using. This could be a printed guide or an electronic guide.

Students can be given a copy of the booklet containing the pseudocode command set (available on our website). This is the same set of commands given in *Appendix 1*.

Students may bring reference materials into the supervised environment. Other information, in addition to the syntax guide, can be searched for on the internet although these sites must be referenced if material is taken from them. If students are allowed to copy and paste large sections of code into their projects then teachers will find it difficult to authenticate the work as the student's own.

Teacher support

Teachers can give feedback to students at all stages of the programming project.
To allow students to improve their work, teachers can review student work and provide both oral and written advice.

Discussions between students is allowed, however teachers must ensure that the work produced by each student is their own.

Authentication

The student and teacher must sign the programming project authentication form (*Appendix 3*) to confirm that the work is that of the student, and that any sources used in the work have been referenced.

The Head of Centre must sign the Head of Centre declaration form (*Appendix 4*) to state that all the requirements listed for the programming project have been adhered to.

The forms and all students work must be submitted at the same time to Pearson by the submission deadline date. Centres are reminded that if they do not submit student work and the forms then it may result in malpractice/maladministration.

Security and backups

It is the centre's responsibility to keep the work that students have submitted for the project secure. For materials stored electronically, centres are strongly advised to utilise firewall protection and virus-checking software, and to employ an effective backup strategy, so that an up-to-date archive of students' evidence is maintained.

3 Assessment information

Component 1: Principles of Computer Science (Paper code: 1CP1/01)

- First assessment: May/June 2018.
- The assessment is 1 hour and 40 minutes.
- The assessment is out of 80 marks.
- Students must answer all questions.
- The paper may include multiple-choice, short-open and extended-open response questions.
- The paper will include questions that target computer-related mathematics.
- Calculators must not be used in the examination.

Content assessed

- Algorithms.
- Programming.
- Data.
- Components of computer systems.
- Networks.
- The bigger picture.
- All content/topics will be assessed in this component.

Component 2: Application of Computational Thinking (Paper code: 1CP1/02)

- First assessment: May/June 2018.
- The assessment is 2 hours.
- The assessment is out of 80 marks.
- Questions are based on a scenario.
- Students must answer all questions.
- The paper may include short-open and extended-open response questions.
- The paper will include questions that target computer-related mathematics.
- Calculators must not be used in the examination.

Content assessed

This component will focus on:

- algorithms
- programming.

This component may also draw on:

- data
- components of computer systems
- networks
- the bigger picture.

Pseudocode booklet

Students will not be expected to remember the syntax for the pseudocode. A pseudocode booklet will be made available electronically on the Pearson website. A clean copy of this booklet should be provided for each student for Component 2 and for the programming project.

Assessment Objectives

Students must:		% in GCSE
AO1	Demonstrate knowledge and understanding of the key concepts and principles of computer science	37
AO2	Apply knowledge and understanding of key concepts and principles of computer science	45
AO3	Analyse problems in computational terms: • to make reasoned judgements • to design, program, evaluate and refine solutions.	18
	Total	**100%**

Breakdown of Assessment Objectives

Component	Assessment Objectives			Total for all Assessment Objectives
	AO1 %	AO2 %	AO3 %	
Component 1: Principles of Computer Science	27	23	0	50
Component 2: Application of Computational Thinking	10	22	18	50
Total for GCSE	**37**	**45**	**18**	**100%**

Synoptic assessment

Synoptic assessment requires students to work across different parts of a qualification and to show their accumulated knowledge and understanding of a topic or subject area.

Synoptic assessment enables students to show their ability to combine their skills, knowledge and understanding with breadth and depth of the subject.

Synopticity will be assessed in Component 2.

Sample assessment materials

Sample papers and mark schemes can be found in the *Pearson Edexcel Level 1/Level 2 GCSE (9–1) in Computer Science* Sample Assessment Materials (SAMs) document.

4 Administration and general information

Entries

Details of how to enter students for the examinations for this qualification can be found in our *UK Information Manual*. A copy is made available to all examinations officers and is available on our website: qualifications.pearson.com

Re-taking the qualification

Students wishing to re-take this qualification are required to re-take both external components in May/June in any single year. With regard to the programming project, there are two options available.

Option 1

Students can carry out the project again, or select a different project from our website, to improve their programming skills.

Option 2

Students can carry forward their previous project. If they choose this option, centres must submit their project along with the projects developed by the other students' and all relevant associated forms by the submission deadline date.

Forbidden combinations and discount code

Centres should be aware that students who enter for more than one GCSE, or other Level 2 qualifications with the same discount code, will have only the grade for their 'first entry' counted for the purpose of the School and College Performance Tables (please see *Appendix 7: Codes*). For further information about what constitutes 'first entry' and full details of how this policy is applied, please refer to the DfE website: www.gov.uk/government/organisations/department-for-education

Students should be advised that if they take two GCSEs with the same discount code, schools and colleges they wish to progress to are likely to take the view that this achievement is equivalent to only one GCSE. The same view may be taken if students take two GCSE or other Level 2 qualifications that have different discount codes but which have significant overlap of content. Students or their advisers who have any doubts about their subject combinations should check with the institution to which they wish to progress before embarking on their programmes.

Access arrangements, reasonable adjustments, special consideration and malpractice

Equality and fairness are central to our work. Our equality policy requires all students to have equal opportunity to access our qualifications and assessments, and our qualifications to be awarded in a way that is fair to every student.

We are committed to making sure that:

- students with a protected characteristic (as defined by the Equality Act 2010) are not, when they are undertaking one of our qualifications, disadvantaged in comparison to students who do not share that characteristic
- all students achieve the recognition they deserve for undertaking a qualification and that this achievement can be compared fairly to the achievement of their peers.

Language of assessment

Assessment of this qualification will be available in English. All student work must be in English.

Access arrangements

Access arrangements are agreed before an assessment. They allow students with special educational needs, disabilities or temporary injuries to:

- access the assessment
- show what they know and can do without changing the demands of the assessment.

The intention behind an access arrangement is to meet the particular needs of an individual student with a disability, without affecting the integrity of the assessment. Access arrangements are the principal way in which awarding bodies comply with the duty under the Equality Act 2010 to make 'reasonable adjustments'.

Access arrangements should always be processed at the start of the course. Students will then know what is available and have the access arrangement(s) in place for assessment.

Reasonable adjustments

The Equality Act 2010 requires an awarding organisation to make reasonable adjustments where a person with a disability would be at a substantial disadvantage in undertaking an assessment. The awarding organisation is required to take reasonable steps to overcome that disadvantage.

A reasonable adjustment for a particular person may be unique to that individual and therefore might not be in the list of available access arrangements.

Whether an adjustment will be considered reasonable will depend on a number of factors, which will include:

- the needs of the student with the disability
- the effectiveness of the adjustment
- the cost of the adjustment; and
- the likely impact of the adjustment on the student with the disability and other students.

An adjustment will not be approved if it involves unreasonable costs to the awarding organisation, timeframes or affects the security or integrity of the assessment. This is because the adjustment is not 'reasonable'.

Special consideration

Special consideration is a post-examination adjustment to a student's mark or grade to reflect temporary injury, illness or other indisposition at the time of the assessment, which has had, or is reasonably likely to have had, a material effect on a candidate's ability to take an assessment or demonstrate their level of attainment in an assessment.

Further information

Please see our website for further information about how to apply for access arrangements and special consideration.

For further information about access arrangements, reasonable adjustments and special consideration, please refer to the JCQ website: www.jcq.org.uk.

Malpractice

Candidate malpractice

Candidate malpractice refers to any act by a candidate that compromises or seeks to compromise the process of assessment or which undermines the integrity of the qualifications or the validity of results/certificates.

Candidate malpractice discovered in the project before the candidate has signed the programming project authentication form does not need to be reported to Pearson.

Candidate malpractice found in the project after the declaration of authenticity has been signed, and in examinations, **must** be reported to Pearson on a *JCQ M1 Form* (available at www.jcq.org.uk/exams-office/malpractice). The completed form can be emailed to pqsmalpractice@pearson.com or posted to Investigations Team, Pearson, 190 High Holborn, London, WC1V 7BH. Please provide as much information and supporting documentation as possible. Note that the final decision regarding appropriate sanctions lies with Pearson.

Failure to report candidate malpractice constitutes staff or centre malpractice.

Staff/centre malpractice

Staff and centre malpractice includes both deliberate malpractice and maladministration of our qualifications. As with candidate malpractice, staff and centre malpractice is any act that compromises or seeks to compromise the process of assessment or undermines the integrity of the qualifications or the validity of results/certificates.

All cases of suspected staff malpractice and maladministration **must** be reported immediately, before any investigation is undertaken by the centre, to Pearson on a *JCQ M2(a) Form* (available at www.jcq.org.uk/exams-office/malpractice). The form, supporting documentation and as much information as possible can be emailed to pqsmalpractice@pearson.com or posted to Investigations Team, Pearson, 190 High Holborn, London, WC1V 7BH. Note that the final decision regarding appropriate sanctions lies with Pearson.

Failure to report malpractice itself constitutes malpractice.

More-detailed guidance on malpractice can be found in the latest version of the document *JCQ General and Vocational Qualifications Suspected Malpractice in Examinations and Assessments,* available at www.jcq.org.uk/exams-office/malpractice.

Awarding and reporting

This qualification will be graded, awarded and certificated to comply with the requirements of Ofqual's General Conditions of Recognition.

This GCSE qualification will be graded and certificated on a nine-grade scale from 9 to 1, using the total subject mark where 9 is the highest grade. Individual components are not graded.

Students whose level of achievement is below the minimum judged by Pearson to be of sufficient standard to be recorded on a certificate will receive an unclassified U result.

The first certification opportunity for this qualification is 2018.

Student recruitment and progression

Pearson follows the JCQ policy concerning recruitment to our qualifications in that:

- they must be available to anyone who is capable of reaching the required standard
- they must be free from barriers that restrict access and progression
- equal opportunities exist for all students.

Prior learning and other requirements

There are no prior learning or other requirements for this qualification.

Progression

Students can progress from this qualification to:

- further studies, for example A Levels, BTECs in Computer Science
- employment, where further training may be available.

Appendices

Appendix 1: Pseudocode command set

Questions in the written examination for Component 2 that involve code will use this pseudocode for clarity and consistency. However, students may answer questions using any valid method.

Data types

INTEGER

REAL

BOOLEAN

CHARACTER

Type coercion

Type coercion is automatic if indicated by context. For example, 3 + 8.25 = 11.25
(integer + real = real)

Mixed mode arithmetic is coerced like this:

	INTEGER	REAL
INTEGER	INTEGER	REAL
REAL	REAL	REAL

Coercion can be made explicit. For example, RECEIVE age FROM (INTEGER) KEYBOARD assumes that the input from the keyboard is interpreted as an INTEGER, not a STRING.

Constants

The value of constants can only ever be set once. They are identified by the keyword CONST. Two examples of using a constant are shown.

CONST REAL PI

SET PI TO 3.14159

SET circumference TO radius * PI * 2

Data structures

ARRAY

STRING

Indices start at zero (0) for all data structures.

All data structures have an append operator, indicated by &.

Using & with a STRING and a non-STRING will coerce to STRING. For example, SEND `Fred' & age TO DISPLAY, will display a single STRING of `Fred18'.

Identifiers

Identifiers are sequences of letters, digits and '_', starting with a letter, for example MyValue, myValue, My_Value, Counter2

Functions

LENGTH()

For data structures consisting of an array or string.

RANDOM(n)

This generates a random number from 0 to n.

Comments

Comments are indicated by the # symbol, followed by any text.

A comment can be on a line by itself or at the end of a line.

Devices

Use of KEYBOARD and DISPLAY are suitable for input and output.

Additional devices may be required, but their function will be obvious from the context. For example, CARD_READER and MOTOR are two such devices.

Notes

In the pseudocode on the following pages, the < > indicates where expressions or values need to be supplied. The < > symbols are not part of the pseudocode.

Variables and arrays		
Syntax	**Explanation of syntax**	**Example**
SET Variable TO <value>	Assigns a value to a variable.	SET Counter TO 0 SET MyString TO 'Hello world'
SET Variable TO <expression>	Computes the value of an expression and assigns to a variable.	SET Sum TO Score + 10 SET Size to LENGTH(Word)
SET Array[index] TO <value>	Assigns a value to an element of a one-dimensional array.	SET ArrayClass[1] TO 'Ann' SET ArrayMarks[3]TO 56
SET Array TO [<value>, …]	Initialises a one-dimensional array with a set of values.	SET ArrayValues TO [1, 2, 3, 4, 5]
SET Array [RowIndex, ColumnIndex] TO <value>	Assigns a value to an element of a two dimensional array.	SET ArrayClassMarks[2,4] TO 92

Selection		
Syntax	**Explanation of syntax**	**Example**

IF <expression> THEN
  <command>
END IF
	If <expression> is true then command is executed.	IF Answer = 10 THEN
		  SET Score TO Score + 1
		END IF

IF <expression> THEN
  <command>
ELSE
  <command>
END IF
	If <expression> is true then first <command> is executed, otherwise second <command> is executed.	IF Answer = 'correct' THEN
		  SEND 'Well done' TO DISPLAY
		ELSE
		  SEND 'Try again' TO DISPLAY
		END IF

Repetition		
Syntax	**Explanation of syntax**	**Example**

WHILE <condition> DO
  <command>
END WHILE
	Pre-conditioned loop. Executes <command> whilst <condition> is true.	WHILE Flag = 0 DO
		  SEND 'All well' TO DISPLAY
		END WHILE

REPEAT <command> UNTIL <expression>	Post-conditioned loop. Executes <command> until <condition> is true. The loop must execute at least once.	REPEAT SET Go TO Go + 1 UNTIL Go = 10
REPEAT <expression> TIMES <command> END REPEAT	Count controlled loop. The number of times <command> is executed is determined by the expression.	REPEAT 100-Number TIMES SEND '*' TO DISPLAY END REPEAT

FOR <id> FROM <expression> TO
<expression> DO
  <command>
END FOR
	Count controlled loop. Executes <command> a fixed number of times.	FOR Index FROM 1 TO 10 DO
		  SEND ArrayNumbers[Index] TO DISPLAY
		END FOR

FOR <id> FROM <expression> TO
<expression> STEP
<expression> DO
  <command>
END FOR
	Count controlled loop using a step.	FOR Index FROM 1 TO 500 STEP 25 DO
		  SEND Index TO DISPLAY
		END FOR

FOR EACH <id> FROM <expression> DO <command> END FOREACH	Count controlled loop. Executes for each element of an array.	SET WordsArray TO ['The', 'Sky', 'is', 'grey'] SET Sentence to '' FOR EACH Word FROM WordsUArray DO SET Sentence TO Sentence & Word & ' ' END FOREACH

Input/output		
Syntax	**Explanation of syntax**	**Example**
SEND <expression> TO DISPLAY	Sends output to the screen.	SEND 'Have a good day.' TO DISPLAY
RECEIVE <identifier> FROM (type) <device>	Reads input of specified type.	RECEIVE Name FROM (STRING) KEYBOARD RECEIVE LengthOfJourney FROM (INTEGER) CARD_READER RECEIVE YesNo FROM (CHARACTER) CARD_READER

File handling		
Syntax	**Explanation of syntax**	**Example**
READ <File> <record>	Reads in a record from a <file> and assigns to a <variable>. Each READ statement reads a record from the file.	READ MyFile.doc Record
WRITE <File> <record>	Writes a record to a file. Each WRITE statement writes a record to the file.	WRITE MyFile.doc Answer1, Answer2, 'xyz 01'

Subprograms		
Syntax	**Explanation of syntax**	**Example**
PROCEDURE <id> (<parameter>, …) BEGIN PROCEDURE <command> END PROCEDURE	Defines a procedure.	PROCEDURE CalculateAverage (Mark1, Mark2, Mark3) BEGIN PROCEDURE SET Avg to (Mark1 + Mark2 + Mark3)/3 END PROCEDURE

FUNCTION <id> (<parameter>, …)
BEGIN FUNCTION
  <command>
RETURN <expression>
END FUNCTION
	Defines a function.	FUNCTION AddMarks (Mark1, Mark2, Mark3)
		BEGIN FUNCTION
		  SET Total to (Mark1 + Mark2 + Mark3)/3
		  RETURN Total
		END FUNCTION

<id> (<parameter>, …)	Calls a procedure or a function.	Add (FirstMark, SecondMark)

Arithmetic operators	
Symbol	**Description**
+	Add
-	Subtract
/	Divide
*	Multiply
^	Exponent
MOD	Modulo
DIV	Integer division

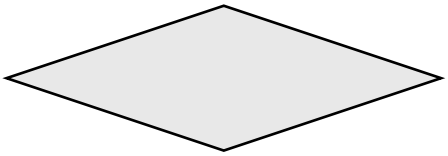
Relational operators	
Symbol	**Description**
=	equal to
<>	not equal to
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to

Logical operators	
Symbol	**Description**
AND	Returns true if both conditions are true.
OR	Returns true if any of the conditions are true.
NOT	Reverses the outcome of the expression; true becomes false, false becomes true.

Appendix 2: Flowchart symbols

Denotes the start and end of an algorithm.

Denotes a process to be carried out.



Denotes a decision to be made.

Shows the logical flow of the program.

Appendix 3: Programming project authentication form

Pearson Edexcel Level 1/Level 2 GCSE in Computer Science

Centre no:						Student number:						
Month/year of examination:						Student name:						
Programming language used:												
I hereby certify that the work submitted for this project is my own. I have clearly referenced any sources used in the work. I understand that false declaration is a form of malpractice. IMPORTANT: **Both the student and the teacher must sign this form.**												
Student name:												
Student signature:							Date:					
Teacher name:												
Teacher signature:							Date:					
Additional student declaration												
By signing this additional declaration you agree to your work being used to support professional development, online support and training. If you have any concerns regarding this please email: ePortfolio@pearson.com						Sign:						

Appendix 4: Head of Centre declaration form

Pearson Edexcel Level 1/Level 2 GCSE in Computer Science

Head teacher declaration

I declare that all students entered for the Pearson Edexcel Level 1/Level 2 GCSE (9–1) in Computer Science have carried out their programming project in accordance with all requirements and that:

(a) each student has had the opportunity to undertake the programming project and has had 20 hours set aside in the timetable to allow them to undertake the project, and

(b) that their written accounts of their programming project represent their individual work, cover each part of the project and reference any resources used or support given.

Centre name:			
Centre number:			
Head teacher name:			
Head teacher signature:		Date:	

Appendix 5: The context for the development of this qualification

All our qualifications are designed to meet our World Class Qualification Principles[1] and our ambition to put the student at the heart of everything we do.

We have developed and designed this qualification by:

- reviewing other curricula and qualifications to ensure that it is comparable with those taken in high-performing jurisdictions overseas
- consulting with key stakeholders on content and assessment, including learned bodies, subject associations, higher-education academics and teachers to ensure this qualification is suitable for a UK context
- reviewing the legacy qualification and building on its positive attributes.

This qualification has also been developed to meet criteria stipulated by Ofqual in their documents *GCSE (9 to 1) Qualification Level Conditions and Requirements* and *GCSE Subject Level Conditions and Requirements for* Computer Science, published in April 2014.

[1] Pearson's World Class Qualification Principles ensure that our qualifications are:

- **demanding**, through internationally benchmarked standards, encouraging deep learning and measuring higher-order skills
- **rigorous**, through setting and maintaining standards over time, developing reliable and valid assessment tasks and processes, and generating confidence in end users of the knowledge, skills and competencies of certified students
- **inclusive**, through conceptualising learning as continuous, recognising that students develop at different rates and have different learning needs, and focusing on progression
- **empowering**, through promoting the development of transferable skills, see *Appendix 6*

From Pearson's Expert Panel for World Class Qualifications

May 2014

"The reform of the qualifications system in England is a profoundly important change to the education system. Teachers need to know that the new qualifications will assist them in helping their learners make progress in their lives.

When these changes were first proposed we were approached by Pearson to join an 'Expert Panel' that would advise them on the development of the new qualifications.

We were chosen, either because of our expertise in the UK education system, or because of our experience in reforming qualifications in other systems around the world as diverse as Singapore, Hong Kong, Australia and a number of countries across Europe.

We have guided Pearson through what we judge to be a rigorous qualification development process that has included:

- Extensive international comparability of subject content against the highest-performing jurisdictions in the world
- Benchmarking assessments against UK and overseas providers to ensure that they are at the right level of demand
- Establishing External Subject Advisory Groups, drawing on independent subject-specific expertise to challenge and validate our qualifications
- Subjecting the final qualifications to scrutiny against the DfE content and Ofqual accreditation criteria in advance of submission.

Importantly, we have worked to ensure that the content and learning is future oriented. The design has been guided by what is called an 'Efficacy Framework', meaning learner outcomes have been at the heart of this development throughout.

We understand that ultimately it is excellent teaching that is the key factor to a learner's success in education. As a result of our work as a panel we are confident that we have supported the development of qualifications that are outstanding for their coherence, thoroughness and attention to detail and can be regarded as representing world-class best practice."

Sir Michael Barber (Chair)

Chief Education Advisor, Pearson plc

Bahram Bekhradnia

President, Higher Education Policy Institute

Dame Sally Coates

Principal, Burlington Danes Academy

Professor Robin Coningham

Pro-Vice Chancellor, University of Durham

Dr Peter Hill

Former Chief Executive ACARA

Professor Lee Sing Kong

Director, National Institute of Education, Singapore

Professor Jonathan Osborne

Stanford University

Professor Dr Ursula Renold

Federal Institute of Technology, Switzerland

Professor Bob Schwartz

Harvard Graduate School of Education

All titles correct as at May 2014.

Appendix 6: Transferable skills

The need for transferable skills

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.

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.' [1]

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 as the most evidence-based and robust skills framework. We adapted the framework slightly to include the Program for International Student Assessment (PISA) ICT Literacy and Collaborative Problem Solving (CPS) Skills.

The adapted National Research Council's framework of skills involves: [2]

Cognitive skills

- **Non-routine problem solving** – expert thinking, metacognition, creativity.
- **Systems thinking** – decision making and reasoning.
- **Critical thinking** – definitions of critical thinking are broad and usually involve general cognitive skills such as analysing, synthesising and reasoning skills.

ICT literacy – access, manage, integrate, evaluate, construct and communicate. [3]

Interpersonal skills

- **Communication** – active listening, oral communication, written communication, assertive communication and non-verbal communication.
- **Relationship-building skills** – teamwork, trust, intercultural sensitivity, service orientation, self-presentation, social influence, conflict resolution and negotiation.
- **Collaborative problem solving** – establishing and maintaining shared understanding, taking appropriate action, establishing and maintaining team organisation.

Intrapersonal skills

- **Adaptability** – ability and willingness to cope with the uncertain, handling work stress, adapting to different personalities, communication styles and cultures, and physical adaptability to various indoor and outdoor work environments.
- **Self-management and self-development** – ability to work remotely in virtual teams, work autonomously, be self-motivating and self-monitoring, willing and able to acquire new information and skills related to work.

Transferable skills enable young people to face the demands of further and higher education, as well as the demands of the workplace, and are important in the teaching and learning of this qualification. We will provide teaching and learning materials, developed with stakeholders, to support our qualifications.

---

[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

[3] PISA (2011) The PISA Framework for Assessment of ICT Literacy, PISA

Appendix 7: Codes

Type of code	Use of code	Code
Discount codes	Every qualification eligible for performance tables is assigned a discount code indicating the subject area to which it belongs. Discount codes are published by DfE in the RAISEonline library (www.raiseonline.org).	Please see the GOV.UK website*
Regulated Qualifications Framework (RQF) codes	Each qualification title is allocated an Ofqual Regulated Qualifications Framework (RQF) code. The RQF code is known as a Qualification Number (QN). This is the code that features in the DfE Section 96 and on the LARA as being eligible for 16–18 and 19+ funding, and is to be used for all qualification funding purposes. The QN will appear on students' final certification documentation.	The QN for this qualification is: 601/8058/4
Subject codes	The subject code is used by centres to enter students for a qualification. Centres will need to use the entry codes only when claiming students' qualifications.	GCSE – 1CP1
Component codes	These codes are provided for reference purposes. Students do not need to be entered for individual components.	Component 1: 1CP1/01 Component 2: 1CP1/02

*www.gov.uk/government/publications/key-stage-4-qualifications-discount-codes-and-point-scores

Db180718V:\LT\PD\GCSE 2017\9781446958469_GCSE2016_12_COMPSCI_ISS4\9781446958469_GCSE2016_12_COMPSCI_ISS4.DOC.1–48/0

Edexcel, BTEC and LCCI qualifications

Edexcel, BTEC and LCCI qualifications are awarded by Pearson, the UK's largest awarding body offering academic and vocational qualifications that are globally recognised and benchmarked. For further information, please visit our qualification website at qualifications.pearson.com. Alternatively, you can get in touch with us using the details on our contact us page at qualifications.pearson.com/contactus

About Pearson

Pearson is the world's leading learning company, with 35,000 employees in more than 70 countries working to help people of all ages to make measurable progress in their lives through learning. We put the learner at the centre of everything we do, because wherever learning flourishes, so do people. Find out more about how we can help you and your learners at qualifications.pearson.com

Original origami artwork: Mark Bolitho
Origami photography: Pearson Education Ltd/Naki Kouyioumtzis

ISBN 978 1 446 95846 9

For information about Pearson Qualifications, including Pearson Edexcel, BTEC and LCCI qualifications visit qualifications.pearson.com

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

VAT Reg No GB 278 537121

Pearson
Edexcel

ISBN 978-1-4469-5846-9