edexcel

Getting Started Guide

GCSE (9-1) Computer Science

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

ALWAYS LEARNING

PEARSON

Getting Started: GCSE Computer Science 2016

Contents

1. Introduction

This Getting Started Guide provides an overview of the new GCSE Computer Science specification. It is designed to help you get to grips with the subject content and assessment and to understand what these mean for you and your students.

1.1 Key principles

Our new GCSE Computer Science specification is intended to develop students' understanding of the principles of computer science and their ability to apply computational thinking to problem solving.

We consulted widely on the proposed content and assessment methodology of the specification. Involved in the consultation were:

- teachers from a number of schools and colleges, with different levels of experience of teaching computer science
- our Expert Stakeholders' Advisory Group, comprising university academics from University of Greenwich, King's College London and Brighton University
- and the wider computer science community, including the Computing at Schools (CAS) group and the British Computer Society (BCS).

Six key principles underpin our new specification. They are:

- **Clear and coherent structure –** making it easy to see what students need to know and be able to do
- **Contemporary content –** emphasising the real-world relevance of computer science and considering the issues and impact of computing technology
- **Focus on computational thinking –** developing an effective approach to problem solving
- **Clear assessment –** assessing theoretical understanding and practical skills
- **Inclusive and accessible –** enabling students of all abilities to study computer science
- **Continuous progression –** building on the knowledge, understanding and skills developed through the computing programme of study in Key Stages 1 to 3 and fostering progression to further study at Key Stage 5 and beyond.

1.2 Support

We are providing a comprehensive package of free support to help you plan and implement our new GCSE in Computer Science.

- **Planning –** An editable course planner and schemes of work that you can adapt to suit your department.
- **Teaching materials –** Editable outline lesson plans, lesson and homework activities with solutions that you can adapt and use to save you time, plus additional teacher support material.
- **Mapping –** The mapping document highlights key differences between the new specification and the previous one, and between different Awarding Organisation's specifications.
- **Understanding the standard –** Sample Assessment Materials (SAMs) and exemplar student work with commentaries, for both the examined and non-examined components, help you to get to grips with the format of the papers and the level of demand.

- **Tracking learner progress –** Our online ResultsPlus service provides detailed analysis of students' examination performance, helping you to identify topics and skills where further learning would benefit your students.
- **examWizard –** A useful examination preparation tool containing a bank of past Edexcel GCSE Computer Science questions, mark schemes and examiners' reports.
- **Advice –** Our subject advisor service, led by Tim Brady, is a direct and personal source of help and support. Sign up to receive Tim's e-newsletter[1] to keep up to date with qualification and service news.
- **Website –** The GCSE 2016 Computer Science pages of our website provide a one-stop shop for information and resources[2].
- **Getting Ready to Teach –** Online and Face2Face (F2F) events helping you to prepare to teach the new GCSE in Computer Science.

1 https://mail.google.com/mail/u/0/?view=cm&fs=1&tf=1&source=mailto&su=Sign+me+up+for+Computer+Science+email+updates&to=TeachingComputerScience@pearson.com&body=Please+sign+me+up+to+receive+regular+Computer+Science+email+updates.

2 www.qualifications.pearson.com/compsci2016

2. What's changed?

2.1 How have GCSEs changed?

- **Updated content –** All GCSEs are being revised, with changes taking place in three phases between 2015 and 2017.
- **Linear –** In future all GCSEs will have a fully linear structure. This means that the content is no longer divided into discrete modules and all assessment takes place at the end of the course.
- **Assessment –** Written examinations are the default method of assessment, except where they cannot provide valid assessment of all the skills required. This is the case with Computer Science. Even where coursework is deemed to be necessary it must contribute no more than 20% of the total marks available and be submitted at the end of the course.
- **Grading –** There is a new grading system that uses numbers instead of letters to represent grades. The new grades are on a scale 9–1, with 9 representing the highest grade and 1 the lowest. See page 19 in the specification for more information on the grading scale and how the grades relate to current GCSE grades.
- **Guided learning hours –** The number of guided learning hours (GLH) allocated to study of a GCSE qualification remains unchanged at 120 GLH.

What does this mean for Computer Science?

For the first time the Department for Education (DfE) has specified subject content for GCSE Computer Science.

Computer Science: GCSE Subject Content (January 2015)[3] sets out the knowledge, understanding and skills common to all GCSEs in Computer Science. Awarding organisations must ensure that their specification provides complete coverage of the subject content. Key features include:

- **Knowledge and understanding –** includes algorithms, contemporary secondary storage, cyber security and network protocol layers, alongside more familiar content such as binary representation of numbers, Boolean logic and systems architecture.
- **Problem solving –** advocates a systematic approach to problem solving using decomposition and abstraction.
- **Programming –** requires students to use one or more high-level programming languages with a textual definition, selected from a list approved by the awarding organisation.
- **Computing-related mathematics –** specifications must include appropriate computing-related mathematics.

Changes to assessment requirements

The *GCSE Subject Level Conditions and Requirements for Computer Science (May 2015)*[4] issued by Ofqual sets out the rules and regulations for GCSEs in Computer Science.

---

3 https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/397550/GCSE_subject_content_for_computer_science.pdf

4 www.gov.uk/government/uploads/system/uploads/attachment_data/file/427399/gcse-subject-level-conditions-and-requirements-for-computer-science.pdf

It stipulates that assessment by examination must contribute 80 per cent of the total marks available, with Non-Examined Assessment (NEA) making up the remaining 20 per cent.

The NEA component must be a single, 20-hour project that requires students to design, write, test and refine a program and produce a written report.

Assessment Objectives

In the same document Ofqual specifies a set of three Assessment Objectives for GCSE qualifications in Computer Science. They are:

AO1 30%	Demonstrate knowledge and understanding of the key concepts and principles of computer science.
AO2 40%	Apply knowledge and understanding of key concepts and principles of computer science.
AO3 30%	Analyse problems in computational terms: • to make reasoned judgements; and • to design, program, evaluate and refine solutions.

These Assessment Objectives place particular emphasis on application of knowledge, understanding and problem solving, reflecting the fact that although Computer Science is an academic discipline with a defined body of knowledge, it is also a highly practical subject.

Further information on how to interpret the Assessment Objectives can be found in the document *GCSE Subject Level Guidance for Computer Science (May 2015)*[5], published by Ofqual.

2.2 The Edexcel Computer Science specification

Updated content

All GCSES are being revised, with changes taking place in three phases between 2015 and 2017.

Our legacy GCSE Computer Science (2013) anticipated many of the changes that Ofqual has now introduced. Consequently, we have been able to carry forward, unaltered, most of the content and approach of the legacy specification into our new 2016 specification. This makes for a very smooth transition from old to new.

That said, it has been necessary to make some changes in order to fully comply with the new subject content and assessment requirements.

Additional content

In response to feedback we received from our stakeholders, we have included some additional topics over and above the mandatory content specified by the DfE. These topics are already in our legacy specification.

- **Databases –** A 'must' given the prevalence of structured data in the world today.
- **Input-process-output –** The model of computation that permeates all aspects of computer science.
- **The internet –** A basic knowledge of the internet is a prerequisite for studying network protocols, cyber-security and contemporary storage.

[5] www.gov.uk/government/publications/gcse-9-to-1-subject-level-guidance-for-computer-science

- **Text files –** Being able to reuse data stored in files is what makes computer processing so powerful, which is why we feel that students should have hands-on programming experience of reading from and writing to files.

How does the new specification compare with the old one?

While the subject content of the two specifications is divided into the same six topics, there are some differences between them. Overall, the 2016 specification has less content than its predecessor.

No longer included:

- Structured English (Topic 1)
- Cartesian coordinates and user interfaces (Topic 2)
- SQL (Topic 3)
- Assembly language (Topic 4)
- HTML/CSS (Topic 5)
- Emerging technologies (Topic 6).

What's new:

- Students have to be able to write pseudo-code as well as read it and there's a greater emphasis on abstraction (Topic 1)
- The record data structure (Topic 2)
- A new sub-section on network security (Topic 5).

There is a mapping document comparing the content of the 2013 specification with that of the new 2016 specification on the GCSE 2016 Computer Science pages of our website[6].

How has the assessment changed?

The 2016 specification has three assessment components rather than two, and the weighting for coursework (see below) has gone down from 25 per cent to 20 per cent.

Two exam papers instead of one

Instead of having just one written paper covering all six topics, the new specification has two papers – one focusing on principles of computer science and the other on computational thinking – each with a weighting of 40 per cent.

Nevertheless, the general approach remains the same: contextualised questions, split into a number of parts, papers 'ramped' so questions become more difficult as students move through the paper and a variety of question styles.

Non-Examined Assessment replaces controlled assessment

Non-Examined Assessment (NEA) replaces controlled assessment (CA). Instead of tackling three related programming tasks in 15 hours as in the legacy 2013 specification, students have 20 hours in which to undertake one substantial project.

In this specification students must analyse a problem and design, implement, test, refine and evaluate a solution. NEA must be carried under similar controlled conditions to those pertaining to controlled assessment (see page 10 for more details of the assessment).

[6] See footnote 2.

3. Planning

3.1 Planning and delivering a linear course

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

We recommend a minimum of two one-hour lessons a week (or equivalent) to adequately cover the content, embed skills such as problem solving and programming into your teaching, and to prepare students for the assessments.

Because this is a linear qualification, teachers have more flexibility in structuring the course and more time for teaching in the first year.

However, when planning delivery, it is important to leave sufficient time for revision in the second year, particularly to revisit topics studied in the first year.

In addition, skills development needs to be ongoing so that students have the necessary problem-solving and programming know-how to tackle the NEA, which must be completed sometime during the second year of the course.

We have developed a free, downloadable course planner together with outline lesson plans and activities, which you may find useful. You can use the materials as given, modify them to suit your students or simply refer to them to get ideas.

These planning materials can be downloaded from the GCSE 2016 Computer Science pages of our website[7].

3.2 Suggested resources

Generic resources

There are some fantastic generic resources available online. CAS has a huge bank of materials. The Khan Academy has some excellent materials designed for self-directed study. There are also a number of introductory computer science MOOCs provided by HE consortia, such as Coursera, Edx and FutureLearn. Some of these may be suitable for students working in self-study mode.

Specification-specific resources

Pearson-published resources provide comprehensive support for the new Edexcel GCSE Computer Science specification. They consist of:

- a Student Book and Institutional Activebook to support great computer science teaching through a scenario-based approach to problem solving and computational thinking.
- a Revision Guide and Workbook to help students with mock and final exam preparation.

We are also working with a range of other publishers, including Hodder, who are seeking endorsement for the resources they are developing for the new specification.

An up-to-date list of suggested resources can be viewed on the GCSE 2016 Computer Science pages of our website[8].

[7] See footnote 2.
[8] See footnote 2.

4. Content guidance

4.1 Overview

The subject content of the specification combines knowledge and understanding of the principles of computer science with practical problem solving and programming skills.

4.2 Subject content

The subject content is divided into 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 one or more sections and each section consists of a number of statements.

The topics are interlinked. It would not be appropriate to work through the specification teaching one topic after another sequentially. This applies particularly to Topics 1 and 2, which are intended to be taught together and to form a continuous 'thread' throughout the course.

Topic 1: Problem solving

The introduction to Topic 1 states that students are expected to develop a set of computational thinking skills that enable them to understand how computer systems work, and design, implement and analyse algorithms for solving problems.

It should be noted that computational thinking is a core component of the entire qualification and is not just confined to Topic 1, which focuses on techniques.

The topic is divided into two sections. Section 1.1 focuses on algorithms and Section 1.2 on decomposition and abstraction.

Students are expected to be able to interpret and express algorithms as written descriptions, flowcharts, pseudo-code and program code.

Any questions in the written examinations that use pseudo-code will use the pseudo-code command set provided in *Appendix 1* of the specification.

Please note – students are not obliged to use this pseudo-code. Any valid alternative is acceptable. However, familiarity with the given pseudo-code could increase confidence when interpreting questions.

The flowchart symbols students are expected to know and be able to use can be found in *Appendix 2* of the specification. There are only four simple flowchart symbols; students **can** use flowchart templates during the examination although these often contain many more symbols than are needed and may be restrictive in the defined space on a question paper.

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

Statement 1.1.8 lists the standard algorithms students are expected to know. These are bubble sort, merge sort, linear searches and binary searches. Students don't

have to use these algorithms in their own programs but they do need to know how they work and be able to demonstrate how they operate on a given set of data.

Statements 1.2.1 and 1.2.2 encapsulate the first stage of problem solving. Students will be expected to adopt this approach when undertaking the NEA project.

Statements 1.2.3 and 1.2.4 cover abstraction; a theme that is revisited in the section on subprograms in Topic 2.

Topic 2: Programming

Topic 2 focuses on the programming knowledge and skills students need to learn and practise. The topic introduction states that they should be competent at designing, reading, writing and debugging computer programs and should be given repeated opportunities to develop and practise their programming skills throughout the course.

Section 2.1 spells out the approach that students should use when developing and testing program code.

Statement 2.1.3 focuses on the three different types of errors that occur in programs.

Statement 2.1.4 covers test plans and test data and Statement 2.1.5 looks at identifying and correcting errors in code.

Sections 2.2 to 2.6 specify the structural components of programs, the programming constructs and the data structures that students are expected to understand and use. In some instances this may mean working with the nearest equivalent. For example, Python doesn't have a built-in data type for an array; the nearest equivalent is a list, which can be used in much the same way as an array. Nor does Python have a dedicated record data type; the built-in dictionary type is the nearest equivalent.

Students are required to use a high-level programming language with a textual definition to complete the NEA assignment. They must choose from one of the Edexcel-approved languages. These are Python, Java, Visual Basic.NET, Pascal or Object Pascal or a C-derived language (C, C++ or C#).

Students are not required to develop programs that have a graphical interface. Simple text (console) mode programs will suffice.

Topic 3: Data

This is a wide-ranging topic covering binary representation of data, data compression, encryption and relational databases. It provides a useful context for developing computing-related maths skills.

Section 3.1 covers binary representation of numbers, conversion from binary to denary and vice versa, binary arithmetic and hexadecimal.

Section 3.2 deals with how text, images and sound are represented in binary and considers the limitations of binary representation.

Section 3.3 focuses on data storage and compression – lossless and lossy. In statement 3.3.1 the terms kilobyte (KB), megabyte (MB), gigabyte (GB) and terabyte (TB) refer to binary representation, for example a kilobyte (KB) is 1024 bytes. Statement 3.3.3 stipulates that students should understand how a lossless run-length encoding algorithm works. Question 2 in the Paper 1 SAM assesses understanding of compression. Statement 3.3.4 focuses on file storage and states that students must be able to calculate file sizes.

Section 3.4 considers the need for data encryption. Statement 3.4.2 requires students to understand how a Caesar cipher algorithm works.

Section 3.5 focuses on databases – how data can be decomposed and structured in a relational database. There is no requirement for students to have any practical experience of using structured query language.

Topic 4: Computers

Topic 4 is concerned with the hardware and software components of a computer system. Students are expected to recognise that computers take many forms from embedded microprocessors to distributed clouds.

Section 4.1 focuses on the input-process-output model of computation, which permeates all aspects of computer science. It describes how computers execute programs – by receiving inputs (instructions and data), working out what to do with them and then outputting some form of result. It encapsulates the process of writing software, a computer program consists of a series of instructions, which accept input, carry out some processing and return output, and it underpins the design of the Von Neumann architecture in Section 4.2.

Section 4.2 focuses on the hardware components of a computer system, the role of the components of the CPU in the fetch-decode-execute cycle and contemporary secondary storage, including the 'cloud'. Statement 4.2.6 hones in on the need for and functions of embedded systems.

Logic is covered in Section 4.3. Students must be able to construct truth tables and produce logic statements for given problems.

Section 4.4 covers operating systems and utility software. Statement 4.4.3 is another 'take' on abstraction. Students are expected to understand how software can be used to simulate and model aspects of the real world.

Section 4.5 covers high- and low-level programming languages. Statement 4.5.2 focuses on different methods of translating high-level languages into machine code, including the role of an assembler, a compiler and an interpreter.

Topic 5: Communication and the internet

Topic 5 focuses on the key principles behind the organisation of computer networks.

Section 5.1 covers networks essentials, including types of networks, usage models and topologies. Statement 5.1.5 is concerned with the role of and need for network protocols. It includes a list of the protocols students are expected to know about.

Network and cyber-security is the focus of Section 5.2 and is a key new topic in the *Computer Science: GCSE Subject Content (January 2015)*[9]. Statement 5.2.3 specifies the forms of cyberattack that students must know about, statement 5.2.4 covers methods of identifying vulnerabilities and statement 5.2.5 demonstrates how to develop resilient software systems that are resistant to cyberattacks.

Section 5.3 covers the internet and world wide web, focusing on the structure of the internet in statement 5.3.1 and the components of the world wide web in statement 5.3.2. There is no need for students to have any practical experience of using a mark-up language.

Topic 6: The bigger picture

Topic 6 focuses on an awareness of the influence of computing technology on emerging trends, the issues and the impact on today's world. Statement 6.1.1 is concerned with the environmental impact, in particular the issues associated with health, energy use and resources. Statement 6.1.2 considers the ethical impact, honing in on privacy, inclusion and professionalism. Statement 6.1.3 is designed to raise awareness of legal and ownership issues associated with computing technology – important considerations for budding computer scientists!

[9] See footnote 3.

5. Assessment guidance

5.1 Overview

The GCSE in Computer Science has three assessment components.

Component	Title	Overview	Summary of assessment
Component 1 40% 1 hour and 40 minutes	Principles of Computer Science	All topics	Examination Multiple choice, short-open, extended-open and open response questions
Component 2 40% 2 hours	Application of Computational Thinking	Focuses mainly on Topics 1 and 2, but may also draw on content from the other four topics	Scenario-based examination Short, extended and open response questions
Component 3 20% 20 hours	Project	A program that is designed, implemented, tested, plus a written report	Non-Examined Assessment A levels-based mark scheme, with separate grids for each of the four stages of development

The raw marks for Components 1 and 2 in this qualification will be scaled to represent the relative weighting of 40 per cent for Component 1 and 40 per cent for Component 2.

This is an awarded qualification. Following the marking of scripts at the end of an examination series, an awarding committee will decide where to set the raw mark grade boundary for each of the three components. Raw mark grade boundaries can vary from series to series.[10] A uniform mark scale (UMS) is used to convert component raw marks into uniform marks. Uniform mark grade boundaries are fixed and do not change from series to series.

Students must sit both examinations at the end of the course. Their NEA assignment marks will be submitted on the date specified by the Joint Council for Qualifications (JCQ).

[10] You can find out more about how grades are awarded by visiting: www.qualifications.pearson.com/en/support/support-topics/results-certification/understanding-marks-and-grades.html

5.2 Examination papers

There are two written examination papers; both are taken at the end of the course.

Paper 1: Principles of Computer Science

Paper 1 targets Assessment Objectives 1 and 2. It assesses content from across the specification.

It is 1 hour and 40 minutes in length, is marked out of 80 and has a weighting of 40 per cent.

All questions are compulsory.

Calculators are not allowed.

A clean copy of the pseudo-code booklet (available electronically on the Pearson website) should be printed and made available to each student.

The first examination will take place in June 2018.

The paper assesses students' understanding of:

- what algorithms are, what they are used for and how they work
- the requirements for writing program code
- binary representation, data representation, data storage and compression, encryption and databases
- components of computer systems
- computer networks, the internet and the world wide web
- the influence of computing technology and the impact on society, including environmental, ethical, legal and ownership issues

and their ability to:

- interpret, amend and create algorithms
- construct truth tables and produce logic statements.

Every question has several parts and is set in a context. A variety of question styles are used: multiple-choice, short-open response, open response and extended-open response. Most questions have some straightforward parts and some more challenging ones. See page 17 for more details of the question styles used in the written paper.

The paper is 'ramped', meaning that questions become more difficult as you move through the paper.

Although spelling, punctuation and grammar are not directly assessed, students are expected to employ good writing skills in their longer answers. For a written extended-response question the maximum number of marks will be 6.

Paper 2: Application of Computational Thinking

As its title makes clear, the focus of Paper 2 is computational thinking. The paper targets all three Assessment Objectives. It draws mainly on Topics 1 and 2, but may also touch upon content from the other four topics.

It is 2 hours in length. This is to ensure that students have sufficient 'thinking' time for problem solving. Like Paper 1, it is marked out of 80 and has a weighting of 40 per cent.

All questions are compulsory.

Calculators are not allowed.

A clean copy of the pseudo-code booklet (pages 23-30 or 71-78 in the Sample Assessment Materials[11]) should be printed and made available to each student.

The first examination will take place in June 2018.

The paper assesses students' understanding of:

- what algorithms are, what they are used for and how they work
- how to develop program code, using programming constructs, data types and structures, input/output, operators and subprograms

and their ability to:

- interpret, amend and create algorithms.

A real-world context is used throughout the paper. As well as more traditional question styles, there are tables to complete and algorithms to interpret, amend, trace and design.

This paper too is 'ramped'.

Longer answers in this paper are likely to involve creation of an algorithm expressed either in pseudo-code or as flowchart. See page 17 for more details of the question styles used in the written paper.

Computer-related mathematics

The use of computer-related mathematics is assessed in context in both written papers, such as in questions that require students to:

- convert between binary and denary (3.1.3)
- perform binary arithmetic (3.1.4)
- convert between hexadecimal and binary (3.1.5)
- convert between units of measurement (3.3.1)
- calculate file sizes (3.3.4)
- construct expressions that use arithmetic, relational or logical operators (2.5.1, 2.5.2, 2.5.3)
- construct truth tables and logic statements (4.3.1, 4.3.2).

5.3 Non-Examined Assessment

The Non-Examined Assessment component targets Assessment Objectives 2 and 3. It is marked out of 60 and has a weighting of 20 per cent.

Students must undertake a single project that requires them to analyse a problem and then design, implement, test, refine and evaluate a computer program to solve it.

Students may complete the assignment over multiple sessions, up to a combined duration of 20 hours.

The program must be written in a high-level textual language. The languages approved for use by Edexcel are:

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

Access to the internet is not allowed.

11 http://qualifications.pearson.com/content/dam/pdf/GCSE/Computer%20Science/2016/Specification%20and%20sample%20assessments/computer-science-sam.pdf

A clean copy of the pseudo-code booklet (available electronically on the Pearson website) should be made available to each student.

A printed or digital syntax guide for the high-level programming language being used can also be made available to students. An example is provided in *Appendix 3* of the specification.

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

A new project brief will be published on the Pearson website each September, from September 2017.

There is no choice of project brief. Students must attempt the project that is published in the final year of their course.

First assessment will take place in 2018.

Students must submit their program along with a report evidencing the development process, including an evaluation.

The NEA assesses students' ability to:

- analyse and decompose problems
- design solutions and create algorithms
- design and use test plans and test data
- develop program code, using programming constructs, data types and structures, input/output, operators and subprograms
- evaluate a program and suggest improvements.

It is the centre's responsibility to ensure that assignments are valid for the series in which they are submitted.

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 is assessed in the NEA.

Project stages

The project is divided into four stages.

1. Analysis
2. Design
   - Solution design
   - Test strategy and initial test plan
3. Implementation
   - Implementing the design
   - Building the solution
4. Testing, refining and evaluation

The **analysis** is worth 6 marks. Students must analyse the given problem and produce a list of the requirements; decompose the problem into manageable sub-problems and produce a brief written description of each; and explain why the chosen decomposition is appropriate. They are not expected to produce flowcharts or pseudo-code at this stage.

More information about the analysis stage of the assignment can be found on *page 20* of the specification.

The **design** is worth 18 marks – 12 for designing a solution to the problem and 6 for devising a test strategy and initial test plan.

Students must design an algorithm (or algorithms) to meet the requirements of the problem. They can use pseudo-code, flowcharts or written descriptions to do so. The use of program code at this stage is not permitted. The algorithm should be fully decomposed into sub-programs and show the links between sub-programs.

The test strategy should outline the approach to testing the student intends to take and should be reflected in the initial test plan. A test plan template will be provided. At this stage only the first four columns should be completed. Where appropriate, test data should cover normal, erroneous and boundary values. Further tests can be added to the plan at a later stage if required.

More information about the design stage of the assignment can be found on *pages 22–24* of the specification.

The **implementation** is worth 24 marks – 6 for implementing the design and 18 for building the solution.

Students must use a high-level programming language to convert their design into executable code. They will have to make decisions about the data types of variables, how best to implement data structures and subprograms in their chosen language, etc.

The final program should be fully functional and meet all the identified requirements. It should have been fully decomposed into subprograms and be clear and easy to understand.

The marks in this strand are for producing a working solution to the problem. A program that demonstrates great coding skills but fails to solve the problem will achieve fewer marks.

The NEA is intended to mirror a 'real world' approach to problem solving, where a solution is refined as it progresses. Refinements should be added to the design and the program code (and commented on).

More information about the implementation stage of the assignment can be found on *pages 25–27* of the specification.

Testing, refining and evaluation is worth 12 marks. Students must carry out the test strategy they designed in stage 2 and complete the 'Final Test Plan'. Additional tests can be added here, including refinements.

Students should correct any errors they encounter during testing and then re-test. A record of any additional tests that are carried out should be added to the test plan.

More detail about the testing, refining and evaluation stage of the assignment can be found on *pages 28–29* of the specification.

What evidence is required?

1. An executable version of the program.
2. A written report documenting the work that has been carried out and including an evaluation of the program. This table details what needs to be included in the report.

Project stage	Report content
Analysis	• A brief introduction to the problem. • A list of the problem requirements. • Decomposition of the problem into sub-problems, including: – a brief description of the purpose of each sub-problem. – a short explanation of the reasoning behind the decomposition.
Design – solution design	• The algorithm(s). • Any refinements to the design identified during implementation, with reasons.
Design – test strategy and initial test plan	• The test strategy. • The initial test plan with the first four columns completed.
Implementation	• A copy of the program code with any refinements noted as comments. • One or more screenshots demonstrating effective use of debugging skills to correct errors.
Testing, refining and evaluation	• The updated and completed test plan. • The evaluation.

Carrying out the project

Work on the NEA must be carried out in a supervised environment, such that students' work can be authenticated. This requires centres to:

- set up secure user accounts for each student, accessible only in the controlled environment and only by that student and their supervisor.
- ensure that no work is taken out of and no reference material or notes are brought into the controlled environment.
- prevent students from accessing the internet or intranet.
- restrict the software available to students to just the high-level programming language they are using and a word processing package.

Students must be supervised and must not work with others on the project. Centres are recommended to install monitoring software do that they can monitor students at individual workstations.

Supervisors may provide support to a student who is not able to carry out sufficient work at one stage to enable them to progress to the next stage. Any help given should be noted on the centre assessor record sheet (available from our website) and marks awarded accordingly.

Students may use precompiled library units with some provisos and can have access to a printed or digital syntax guide. An example is provided in *Appendix 3* of the specification.

Further information about carrying out the project can be found on *pages 14–17* of the specification.

Marking guidance

Teachers should mark students' finished assignments using the levels-based marking grids provided on *pages 19–29* of the specification. The same marking grids are used to assess every NEA project.

A two-stage process should be used to mark the NEA. First decide which level best describes the student's response and then decide the 'best fit' mark within that level. Teachers should be prepared to use the full range of marks available in a level.

Moderation process

Edexcel will moderate the work. As of December 2015 discussion is ongoing between all the Awarding Organisations and Ofqual about moderation procedures

Further information about marking, standardisation and moderation can be found on *page 17* of the specification.

Teachers' guide to the NEA

Further information about the NEA can be found in the Teachers' Guide to the NEA, due to be published in Spring 2016

6. Mark schemes and question styles

6.1 Question types

A range of question types is used across the two written assessments.

- **Short response and multiple-choice –** Usually 1 mark.
- **Open response –** Usually 2–4 marks.
- **Extended open response (written) –** 6 marks; marked using a levels-based mark scheme.
- **Extended open response (algorithm) –** 6 marks; marked using a levels-based mark scheme.
- **Calculation –** Usually 1–2 marks.

6.2 Command words

Amend – Alter a given algorithm so that it performs differently or to correct an error.

Assess – Judge or decide the value, quality or importance of something in a particular context.

For example: 'Assess how appropriate a 1-dimensional array is as opposed to using separate variables when storing this data.'

Calculate – Work out a numerical answer, showing relevant stages in the working.

For example: 'Calculate how many bits are being transmitted per second for a network described as **three** Mbps.'

Compare – Look for the similarities and differences between two (or more) items/situations. The answer must relate to both (or all) things mentioned in the question and should include at least one similarity and one difference.

For example: 'Compare storing data in the 'cloud' with storing data on hard discs connected to the school's servers.'

Complete – Fill in/write all the details asked for.

For example: 'Complete the flowchart to show this process.'

Construct – Display information in a diagrammatic or tabular form.

For example: 'Construct test data to meet the requirements set out in the table.'

Convert – Express a quantity in alternative units.

For example: 'Convert the hexadecimal number 3F to 8-bit binary.'

Derive – Use the information provided in the question to work out some new information.

For example: 'The ASCII code for the character 'D' is 68 in denary. Derive the ASCII code for the character 'J' in denary.'

Describe – Give an account of something. Statements in the response are often linked.

For example: 'Describe how binary digits are used to represent bitmap images.'

Discuss – Explore the issue/situation/problem presented in the question. Conclusions should be presented clearly and supported by appropriate evidence.

For example: 'A student is learning to program. Discuss the suitability of compiled and interpreted programming languages for the student.'

Draw – Produce an output, either by freehand or using a ruler.

For example: 'Draw a flowchart or a diagram of a data structure.'

Evaluate – Review information then bring it together to form a conclusion, drawing on evidence including strengths, weaknesses, alternative actions, relevant data or information. Come to a supported judgement of a subject's qualities and relation to its context.

For example: 'Evaluate the use of a 'divide and conquer' strategy for sorting and searching data.'

Explain – An explanation that requires a justification/exemplification of a point. The answer must contain some element of reasoning/justification.

For example: 'Explain how a run-length encoding algorithm works.'

Fill in – Synonymous with 'Complete'.

For example: 'Fill in the table to show an input, a process and an output, using the following information:'

Give – Recall a piece of information. When used in relation to a context, it is used to determine a student's grasp of the factual information presented.

For example: 'Give **one** reason why programmers use subprograms.'

Identify – This requires some key information to be selected from a given stimulus/resource.

For example: 'Identify **two** ethical issues associated with the use of computing technology.'

List – Give a sequence of brief answers with no explanation.

For example: 'Two fields in a network packet are source and destination. List **three** additional fields found in a network packet.'

Name – Synonymous with 'Give'.

For example: 'Name the component that holds instructions and data for programs waiting to be run by the CPU.'

Select – Select the correct answer from a set of four possible answers to a multiple-choice question.

Show – Give the steps involved in deriving a result.

For example: 'Show the result of applying an algorithm to some given data.'

State – Synonymous with 'Give'.

For example: 'State what is meant by the term overflow.'

Suggest – Propose a solution in written form.

For example: 'Sam is concerned about the environmental impact of using computers. Suggest **one** possible action he could take to reduce the environmental impact.'

Write – Compose an expression, a line of pseudo-code or an algorithm.

For example: 'Write an expression to calculate how many different numbers can be represented by an 8-bit binary number.'

6.3 Mark schemes

The mark schemes used to assess students' responses in the written examination show a model solution, but alternative and valid solutions will also be rewarded.

Answers to extended open-response questions are marked using a levels-based mark scheme.

Clear wording makes the expectations clear for teachers and for markers.

The application of the new mark schemes will be demonstrated with marked student answers to the SAMs questions accompanied by examiner commentary. These will be available on the GCSE 2016 Computer Science pages of our website.

7. Grading structure

Ofqual has published the following guidance regarding the grading of new GCSEs.

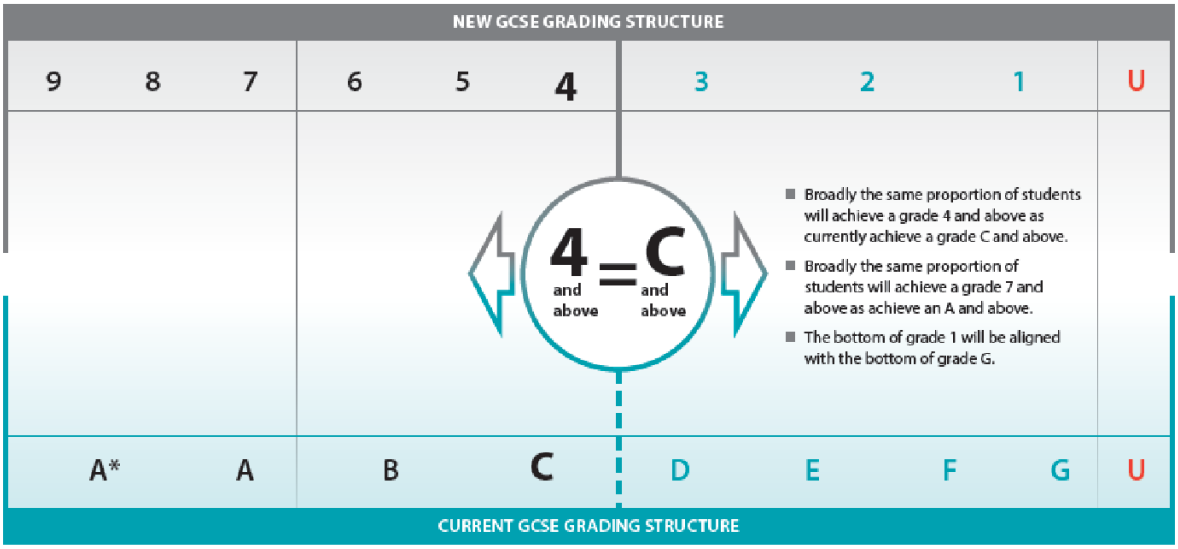
Broadly the same proportion of students will achieve a **grade 4** and above as currently achieve a grade C and above.

Broadly the same proportion of students will achieve a **grade 7** and above as currently achieve an A and above.

For each exam, the top 20 per cent of those who get **grade 7** or above will get a **grade 9** – the very highest performers.

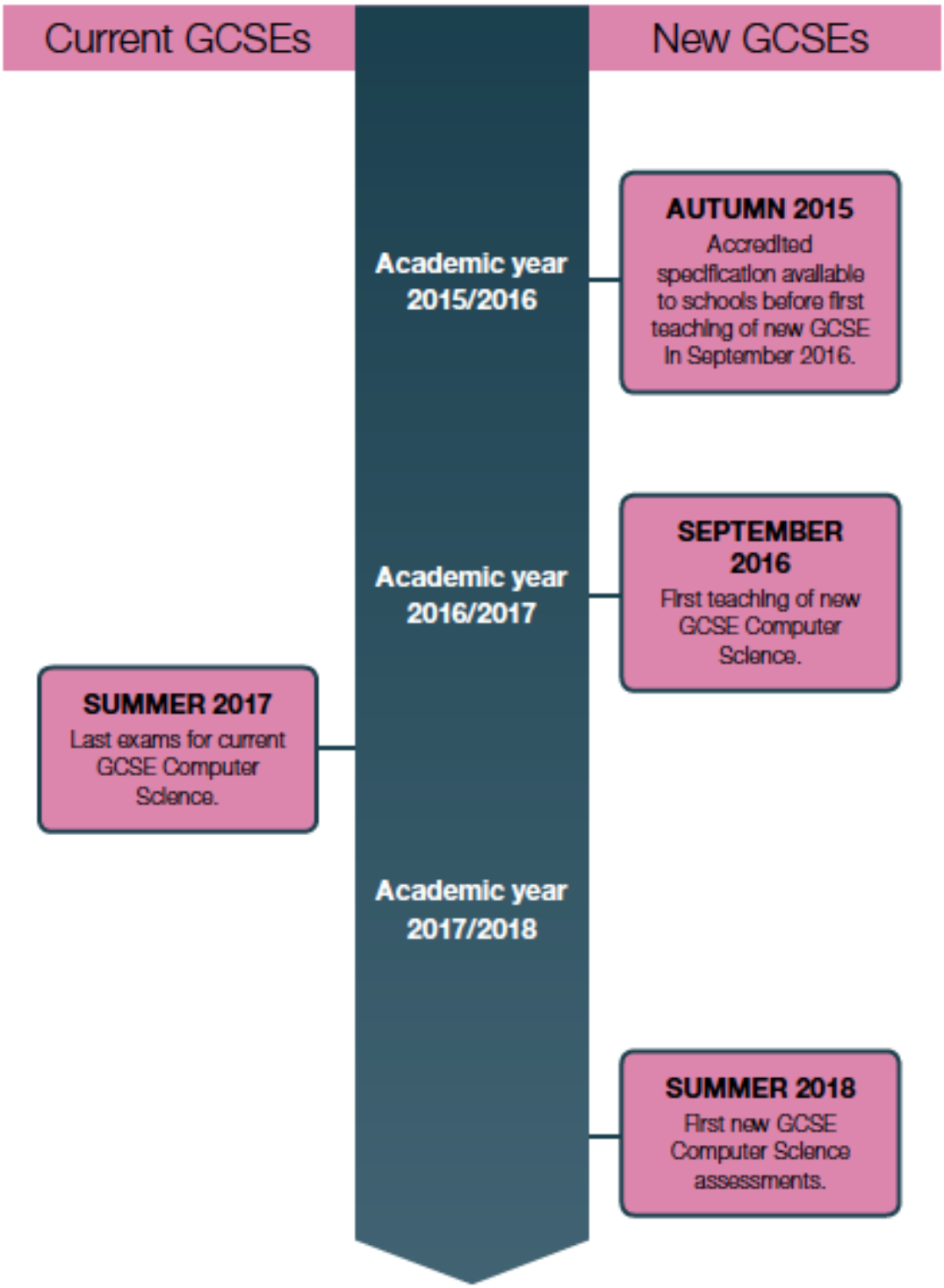
The bottom of **grade 1** will be aligned with the bottom of grade G.

Grade 5 will be positioned in the top third of the marks for a current grade C and in the bottom third of the marks for a current grade B. This will mean it will be more demanding than the present grade C and broadly in line with what the best available evidence tells us is the average PISA performance in countries such as Finland, Canada, the Netherlands and Switzerland. [12]



[12] Ofqual diagram September 2014

8. Timeline



This diagram shows the timeline for phasing out the current GCSE and replacing it with the new one. The last cohort of students taking the old GCSE will finish in summer 2017. First teaching of the new GCSE starts in September 2016, with first assessments taking place in June 2018.