Accredited

# A LEVEL
*Specification*

# COMPUTER SCIENCE

H446
For first assessment in 2017

ocr.org.uk/alevelcomputerscience

Version 2.3 (October 2019)

OCR
Oxford Cambridge and RSA

*Registered office:*
*1 Hills Road*
*Cambridge*
*CB1 2EU*

*OCR is an exempt charity.*



*Disclaimer*

Specifications are updated over time. Whilst every effort is made to check all documents, there may be contradictions between published resources and the specification, therefore please use the information on the latest specification at all times. Where changes are made to specifications these will be indicated within the document, there will be a new version number indicated, and a summary of the changes. If you do notice a discrepancy between the specification and a resource please contact us at: **resources.feedback@ocr.org.uk**

We will inform centres about changes to specifications. We will also publish changes on our website. The latest version of our specifications will always be those on our website (**ocr.org.uk**) and these may differ from printed versions.

# Contents

## Introducing…
## A Level Computer Science (from September 2015)

Computer Science is a practical subject where students can apply the academic principles learned in the classroom to real-world systems. It's an intensely creative subject that combines invention and excitement, and can look at the natural world through a digital prism.

The aims of this qualification are to enable learners to develop:

- An understanding and ability to apply the fundamental principles and concepts of computer science, including: abstraction, decomposition, logic, algorithms and data representation
- The ability to analyse problems in computational terms through practical experience of solving such problems, including writing programs to do so
- The capacity to think creatively, innovatively, analytically, logically and critically
- The capacity to see relationships between different aspects of computer science
- Mathematical skills.

**Contact the team**

We have a dedicated team of people working on our A Level Computer Science qualifications.

If you need specialist advice, guidance or support, get in touch as follows:

- **01223 553998**
- **computerscience@ocr.org.uk**
- **@OCR_ict**

## Teaching and learning resources

We recognise that the introduction of a new specification can bring challenges for implementation and teaching. Our aim is to help you at every stage and we're working hard to provide a practical package of support in close consultation with teachers and other experts, so we can help you to make the change.

**Designed to support progression for all**

Our resources are designed to provide you with a range of teaching activities and suggestions so you can select the best approach for your particular students. You are the experts on how your students learn and our aim is to support you in the best way we can.

**We want to…**

- Support you with a body of knowledge that grows throughout the lifetime of the specification
- Provide you with a range of suggestions so you can select the best activity, approach or context for your particular students
- Make it easier for you to explore and interact with our resource materials, in particular to develop your own schemes of work
- Create an ongoing conversation so we can develop materials that work for you.

**Plenty of useful resources**

You'll have four main types of subject-specific teaching and learning resources at your fingertips:

- Delivery Guides
- Transition Guides
- Topic Exploration Packs
- Lesson Elements.

Along with subject-specific resources, you'll also have access to a selection of generic resources that focus on skills development and professional guidance for teachers.

**Skills Guides** – we've produced a set of Skills Guides that are not specific to Computer Science, but each covers a topic that could be relevant to a range of qualifications – for example, communication, legislation and research. Download the guides at **ocr.org.uk/skillsguides**.

**Active Results** – a free online results analysis service to help you review the performance of individual students or your whole school. It provides access to detailed results data, enabling more comprehensive analysis of results in order to give you a more accurate measurement of the achievements of your centre and individual students. For more details refer to **ocr.org.uk/activeresults**.

## Professional development

Take advantage of our improved Professional Development Programme, designed with you in mind. Whether you want to come to events, look at our new digital training or search for training materials, you can find what you're looking for all in one place at the CPD Hub.

**An introduction to the new specifications**

We'll be running events to help you get to grips with our A Level Computer Science qualification.

These events are designed to help prepare you for first teaching and to support your delivery at every stage.

Watch out for details at **cpdhub.ocr.org.uk**.

To receive the latest information about the training we'll be offering, please register for A Level email updates at **ocr.org.uk/updates**.

# 1 Why choose an OCR A Level in Computer Science?

## 1a. Why choose an OCR qualification?

Choose OCR and you've got the reassurance that you're working with one of the UK's leading exam board. Our new A Level in Computer Science course has been developed in consultation with teachers, employers and Higher Education to provide students with a qualification that's relevant to them and meets their needs.

We're part of the Cambridge Assessment Group, Europe's largest assessment agency and a department of the University of Cambridge. Cambridge Assessment plays a leading role in developing and delivering assessments throughout the world, operating in over 150 countries.

We work with a range of education providers, including schools, colleges, workplaces and other institutions in both the public and private sectors. Over 13,000 centres choose our A levels, GCSEs and vocational qualifications including Cambridge Nationals and Cambridge Technicals.

**Our Specifications**

We believe in developing specifications that help you bring the subject to life and inspire your students to achieve more.

We've created teacher-friendly specifications based on extensive research and engagement with the teaching community. They're designed to be straightforward and accessible so that you can tailor the delivery of the course to suit your needs. We aim to encourage learners to become responsible for their own learning, confident in discussing ideas, innovative and engaged.

We provide a range of support services designed to help you at every stage, from preparation through to the delivery of our specifications. This includes:

A wide range of high-quality creative resources including:

- Delivery Guides
- Transition Guides
- Topic Exploration Packs
- Lesson Elements
- …and much more.

- Access to Subject Advisors to support you through the transition and throughout the lifetimes of the specifications.
- CPD/Training for teachers to introduce the qualifications and prepare you for first teaching.
- Active Results – our free results analysis service to help you review the performance of individual students or whole schools.
- ExamBuilder – our new free online past papers service that enables you to build your own test papers from past OCR exam questions can be found at **http://www.ocr.org.uk/exambuilder**.

All A level qualifications offered by OCR are accredited by Ofqual, the Regulator for qualifications offered in England. The accreditation number for OCR A Level in Computer Science is QN: 601/4911/5.

## 1b. Why choose an OCR A Level in Computer Science?

1

*"At its heart lies the notion of computational thinking: a mode of thought that goes well beyond software and hardware, and that provides a framework within which to reason about systems and problems."*
(CAS-Computer Science a Curriculum for Schools).

This specification has been developed by the team that created the first GCSE Computing qualification in the UK. The experience of the past three years of assessment has clearly demonstrated that OCR has the knowledge and skills to develop reliable and valid qualifications in this area of study.

OCR Computer Science will above all else be relevant to the modern and changing world of computing. It enables teachers to tailor the qualification to meet the needs of their learners in their centre and has an open source ethos allowing any programming language that meets the needs of the course to be used.

Computer Science is a practical subject where learners can apply the academic principles learned in the classroom to real world systems.

It is an intensely creative subject that combines invention and excitement, and can look at the natural world through a digital prism. OCR's A Level in Computer Science will value computational thinking, helping learners to develop the skills to solve problems, design systems and understand the power and limits of human and machine intelligence.

Learners will develop an ability to analyse, critically evaluate and make decisions. The project approach is a vital component of 'post-school' life and is of particular relevance to Further Education, Higher Education and the workplace. Each learner is able to tailor their project to fit their individual needs, choices and aspirations. OCR offers a rigorous assessment structure that ensures the integrity of the project.

### Aims and learning outcomes

The aims of this qualification are to enable learners to develop:

- an understanding of and ability to apply the fundamental principles and concepts of computer science including; abstraction, decomposition, logic, algorithms and data representation
- the ability to analyse problems in computational terms through practical experience of solving such problems including writing programs to do so
- the capacity for thinking creatively, innovatively, analytically, logically and critically
- the capacity to see relationships between different aspects of computer science
- mathematical skills
- the ability to articulate the individual (moral), social (ethical), legal and cultural opportunities and risks of digital technology.

## 1c. What are the key features of this specification?

The OCR A Level in Computer Science will encourage learners to be inspired, motivated and challenged by following a broad, coherent, practical, satisfying and worthwhile course of study. It will provide insight into, and experience of how computer science works, stimulating learners' curiosity and encouraging them to engage with computer science in their everyday lives and to make informed choices about further study or career choices.

The key features of this specification encourage:

- emphasis on problem solving using computers
- emphasis on computer programming and algorithms
- emphasis on the mathematical skills used to express computational laws and processes, e.g. Boolean algebra/logic and comparison of the complexity of algorithms
- less emphasis on ICT.

Centres and learners have the opportunity to:

- produce a slimmed down programming project which is more refined and more focussed on coding
- choose the project title and problem to be solved
- choose any suitable programming language
- include agile methods.

## 1d. How do I find out more information?

If you are already using OCR specifications you can contact us at: **www.ocr.org.uk**

If you are not already a registered OCR centre then you can find out more information on the benefits of becoming one at: **www.ocr.org.uk**

If you are not yet an approved centre and would like to become one go to: **www.ocr.org.uk**

Find out more?

Ask a Subject Advisor:

Email: **computerscience@ocr.org.uk**

Customer Contact Centre: 01223 553998

Teacher support: **www.ocr.org.uk**

News: **www.ocr.org.uk**

# 2 The specification overview

## 2a. Overview of A Level in Computer Science (H446)

Learners must take three components (01, 02 and 03 or 01, 02 and 04) to be awarded the OCR A Level in Computer Science.



| Content Overview | Assessment Overview | |
|---|---|---|
| • The characteristics of contemporary processors, input, output and storage devices<br>• Software and software development<br>• Exchanging data<br>• Data types, data structures and algorithms<br>• Legal, moral, cultural and ethical issues<br><br>• Elements of computational thinking<br>• Problem solving and programming<br>• Algorithms to solve problems and standard algorithms<br><br>*The learner will choose a computing problem to work through according to the guidance in the specification.*<br>• Analysis of the problem<br>• Design of the solution<br>• Developing the solution<br>• Evaluation | Computer systems (01)<br>140 marks<br>2 hours and 30 minutes<br>written paper<br>(no calculators allowed) | **40%** of total A level |
| | Algorithms and programming (02*)<br>140 marks<br>2 hours and 30 minutes<br>written paper<br>(no calculators allowed) | **40%** of total A level |
| | Programming project<br>03* – Repository<br>or<br>04* – Postal<br>or<br>80 – Carry forward (2018 onwards)*<br>70 marks<br>Non-exam assessment | **20%** of total A level |

* Indicates synoptic assessment

## 2b. Content of A Level in Computer Science (H446)

The content of this A Level in Computer Science is divided into three components:

- Computer systems component (01) contains the majority of the content of the specification and is assessed in a written paper recalling knowledge and understanding.
- Algorithms and programming component (02) relates principally to problem solving skills needed by learners to apply the knowledge and understanding encountered in Component 01.
- Programming project component (03 or 04) is a practical portfolio based assessment with a task that is chosen by the teacher or learner and is produced in an appropriate programming language of the learner's or teacher's choice. Appendix 5d of this specification gives a list of programming languages which OCR will accept. If the task demands another choice of language that does not appear in the list, the task outline, the details of the programming language and the reasons for the choice of this language should be submitted to OCR for consideration. Please contact **computerscience@ocr.org.uk** for further advice and guidance.

This specification has been designed to be co-teachable with the stand alone AS Level in Computer Science.

Mathematical skills are embedded throughout the content of the three components. They will be assessed in the written papers and through the non examined assessment where appropriate. The quality of extended responses are assessed in the written papers where indicated by an asterisk. It is marked using levels of response style mark schemes and in the Evaluation section of the Programming project component.

## 2c. Content of Computer systems (Component 01)

This component will introduce learners to the internal workings of the Central Processing Unit (CPU), the exchange of data and will also look at software development, data types and legal and ethical issues. It is expected that learners will draw on this underpinning content when studying computational thinking, developing programming techniques and devising their own programming approach in the Programming project component (03 or 04).

Learners will be expected to apply the criteria below in different contexts including current and future uses of the technologies.



| **1.1 The characteristics of contemporary processors, input, output and storage devices** | |
|---|---|
| **Components of a computer and their uses** | |
| 1.1.1 Structure and function of the processor | (a) The Arithmetic and Logic Unit; ALU, Control Unit and Registers (Program Counter; PC, Accumulator; ACC, Memory Address Register; MAR, Memory Data Register; MDR, Current Instruction Register; CIR). Buses: data, address and control: how this relates to assembly language programs.<br>(b) The Fetch-Decode-Execute Cycle; including its effects on registers.<br>(c) The factors affecting the performance of the CPU: clock speed, number of cores, cache.<br>(d) The use of pipelining in a processor to improve efficiency.<br>(e) Von Neumann, Harvard and contemporary processor architecture. |
| 1.1.2 Types of processor | (a) The differences between and uses of CISC and RISC processors.<br>(b) GPUs and their uses (including those not related to graphics).<br>(c) Multicore and Parallel systems. |
| 1.1.3 Input, output and storage | (a) How different input, output and storage devices can be applied to the solution of different problems.<br>(b) The uses of magnetic, flash and optical storage devices.<br>(c) RAM and ROM.<br>(d) Virtual storage. |

| 1.2 Software and software development | |
|---|---|
| **Types of software and the different methodologies used to develop software** | |
| 1.2.1 Systems Software | (a) The need for, function and purpose of operating systems.<br>(b) Memory Management (paging, segmentation and virtual memory).<br>(c) Interrupts, the role of interrupts and Interrupt Service Routines (ISR), role within the Fetch-Decode-Execute Cycle.<br>(d) Scheduling: round robin, first come first served, multi-level feedback queues, shortest job first and shortest remaining time.<br>(e) Distributed, embedded, multi-tasking, multi-user and Real Time operating systems.<br>(f) BIOS.<br>(g) Device drivers.<br>(h) Virtual machines, any instance where software is used to take on the function of a machine, including executing intermediate code or running an operating system within another. |
| 1.2.2 Applications Generation | (a) The nature of applications, justifying suitable applications for a specific purpose.<br>(b) Utilities.<br>(c) Open source vs closed source.<br>(d) Translators:Interpreters, compilers and assemblers.<br>(e) Stages of compilation (lexical analysis, syntax analysis, code generation and optimisation).<br>(f) Linkers and loaders and use of libraries. |
| 1.2.3 Software Development | (a) Understand the waterfall lifecycle, agile methodologies, extreme programming, the spiral model and rapid application development.<br>(b) The relative merits and drawbacks of different methodologies and when they might be used.<br>(c) Writing and following algorithms. |
| 1.2.4 Types of Programming Language | (a) Need for and characteristics of a variety of programming paradigms.<br>(b) Procedural languages.<br>(c) Assembly language (including following and writing simple programs with the Little Man Computer instruction set). See appendix 5d.<br>(d) Modes of addressing memory (immediate, direct, indirect and indexed).<br>(e) Object-oriented languages (see appendix 5d for pseudocode style) with an understanding of classes, objects, methods, attributes, inheritance, encapsulation and polymorphism. |

| 1.3 Exchanging data | |
|---|---|
| **How data is exchanged between different systems** | |
| 1.3.1 Compression, Encryption and Hashing | (a) Lossy vs Lossless compression.<br>(b) Run length encoding and dictionary coding for lossless compression.<br>(c) Symmetric and asymmetric encryption.<br>(d) Different uses of hashing. |
| 1.3.2 Databases | (a) Relational database, flat file, primary key, foreign key, secondary key, entity relationship modelling, normalisation and indexing. See appendix 5f.<br>(b) Methods of capturing, selecting, managing and exchanging data.<br>(c) Normalisation to 3NF.<br>(d) SQL – Interpret and modify. See appendix 5d.<br>(e) Referential integrity.<br>(f) Transaction processing, ACID (Atomicity, Consistency, Isolation, Durability), record locking and redundancy. |
| 1.3.3 Networks | (a) Characteristics of networks and the importance of protocols and standards.<br>(b) The internet structure:<br>• The TCP/IP Stack.<br>• DNS<br>• Protocol layering.<br>• LANs and WANs.<br>• Packet and circuit switching.<br>(c) Network security and threats, use of firewalls, proxies and encryption.<br>(d) Network hardware.<br>(e) Client-server and peer to peer. |
| 1.3.4 Web Technologies | (a) HTML, CSS and JavaScript. See appendix 5d.<br>(b) Search engine indexing.<br>(c) PageRank algorithm.<br>(d) Server and client side processing. |

| 1.4 Data types, data structures and algorithms | |
|---|---|
| **How data is represented and stored within different structures. Different algorithms that can be applied to these structures** | |
| 1.4.1 Data Types | (a) Primitive data types, integer, real/floating point, character, string and Boolean.<br>(b) Represent positive integers in binary.<br>(c) Use of sign and magnitude and two's complement to represent negative numbers in binary.<br>(d) Addition and subtraction of binary integers.<br>(e) Represent positive integers in hexadecimal.<br>(f) Convert positive integers between binary hexadecimal and denary.<br>(g) Representation and normalisation of floating point numbers in binary.<br>(h) Floating point arithmetic, positive and negative numbers, addition and subtraction.<br>(i) Bitwise manipulation and masks: shifts, combining with AND, OR, and XOR.<br>(j) How character sets (ASCII and UNICODE) are used to represent text. |
| 1.4.2 Data Structures | (a) Arrays (of up to 3 dimensions), records, lists, tuples.<br>(b) The following structures to store data: linked-list, graph (directed and undirected), stack, queue, tree, binary search tree, hash table.<br>(c) How to create, traverse, add data to and remove data from the data structures mentioned above. *(NB this can be **either** using arrays and procedural programming **or** an object-oriented approach).* |
| 1.4.3 Boolean Algebra | (a) Define problems using Boolean logic. See appendix 5d.<br>(b) Manipulate Boolean expressions, including the use of Karnaugh maps to simplify Boolean expressions.<br>(c) Use the following rules to derive or simplify statements in Boolean algebra: De Morgan's Laws, distribution, association, commutation, double negation.<br>(d) Using logic gate diagrams and truth tables. See appendix 5d.<br>(e) The logic associated with D type flip flops, half and full adders. |

| 1.5 Legal, moral, cultural and ethical issues | |
|---|---|
| **The individual moral, social, ethical and cultural opportunities and risks of digital technology. Legislation surrounding the use of computers and ethical issues that can or may in the future arise from the use of computers** | |
| 1.5.1 Computing related legislation | (a) The Data Protection Act 1998.<br>(b) The Computer Misuse Act 1990.<br>(c) The Copyright Design and Patents Act 1988.<br>(d) The Regulation of Investigatory Powers Act 2000. |
| 1.5.2 Moral and ethical Issues | The individual moral, social, ethical and cultural opportunities and risks of digital technology:<br>• Computers in the workforce.<br>• Automated decision making.<br>• Artificial intelligence.<br>• Environmental effects.<br>• Censorship and the Internet.<br>• Monitor behaviour.<br>• Analyse personal information.<br>• Piracy and offensive communications.<br>• Layout, colour paradigms and character sets. |

## 2c. Content of Algorithms and programming (Component 02)

This component will incorporate and build on the knowledge and understanding gained in the Computer systems component (01).

In addition, learners should:

- understand what is meant by computational thinking
- understand the benefits of applying computational thinking to solving a wide variety of problems
- understand the principles of solving problems by computational methods
- be able to use algorithms to describe problems
- be able to analyse a problem by identifying its component parts.



| **2.1 Elements of computational thinking** | |
|---|---|
| **Understand what is meant by computational thinking** | |
| 2.1.1 Thinking abstractly | (a) The nature of abstraction.<br>(b) The need for abstraction.<br>(c) The differences between an abstraction and reality.<br>(d) Devise an abstract model for a variety of situations. |
| 2.1.2 Thinking ahead | (a) Identify the inputs and outputs for a given situation.<br>(b) Determine the preconditions for devising a solution to a problem.<br>(c) The nature, benefits and drawbacks of caching.<br>(d) The need for reusable program components. |
| 2.1.3 Thinking procedurally | (a) Identify the components of a problem.<br>(b) Identify the components of a solution to a problem.<br>(c) Determine the order of the steps needed to solve a problem.<br>(d) Identify sub-procedures necessary to solve a problem. |
| 2.1.4 Thinking logically | (a) Identify the points in a solution where a decision has to be taken.<br>(b) Determine the logical conditions that affect the outcome of a decision.<br>(c) Determine how decisions affect flow through a program. |
| 2.1.5 Thinking concurrently | (a) Determine the parts of a problem that can be tackled at the same time.<br>(b) Outline the benefits and trade offs that might result from concurrent processing in a particular situation. |

| 2.2 Problem solving and programming | |
|---|---|
| **How computers can be used to solve problems and programs can be written to solve them** ***(Learners will benefit from being able to program in a procedure/imperative language and object oriented language.)*** | |
| 2.2.1 Programming techniques | (a) Programming constructs: sequence, iteration, branching.<br>(b) Recursion, how it can be used and compares to an iterative approach.<br>(c) Global and local variables.<br>(d) Modularity, functions and procedures, parameter passing by value and by reference.<br>(e) Use of an IDE to develop/debug a program.<br>(f) Use of object oriented techniques. |
| 2.2.2 Computational methods | (a) Features that make a problem solvable by computational methods.<br>(b) Problem recognition.<br>(c) Problem decomposition.<br>(d) Use of divide and conquer.<br>(e) Use of abstraction.<br>(f) Learners should apply their knowledge of:<br>• backtracking<br>• data mining<br>• heuristics<br>• performance modelling<br>• pipelining<br>• visualisation to solve problems. |

| 2.3 Algorithms | |
|---|---|
| **The use of algorithms to describe problems and standard algorithms** | |
| 2.3.1 Algorithms | (a) Analysis and design of algorithms for a given situation.<br>(b) The suitability of different algorithms for a given task and data set, in terms of execution time and space.<br>(c) Measures and methods to determine the efficiency of different algorithms, Big O notation (constant, linear, polynomial, exponential and logarithmic complexity).<br>(d) Comparison of the complexity of algorithms.<br>(e) Algorithms for the main data structures, (stacks, queues, trees, linked lists, depth-first (post-order) and breadth-first traversal of trees).<br>(f) Standard algorithms (bubble sort, insertion sort, merge sort, quick sort, Dijkstra's shortest path algorithm, A* algorithm, binary search and linear search). |

## 2c. Content of non exam assessment Programming project (Component 03 or 04)

Learners will be expected to analyse, design, develop, test, evaluate and document a program written in a suitable programming language. The underlying approach to the project is to apply the principles of computational thinking to a practical coding problem. Learners are expected to apply appropriate principles from an agile development approach to the project development.

While the project assessment criteria are organised into specific categories, it is anticipated the final report will document the agile development process and elements for each of the assessment categories will appear throughout the report.



| **3.1. Analysis of the problem (10 marks)** | |
|---|---|
| 3.1.1 Problem identification | (a) Describe and justify the features that make the problem solvable by computational methods.<br>(b) Explain why the problem is amenable to a computational approach. |
| 3.1.2 Stakeholders | (a) Identify and describe those who will have an interest in the solution explaining how the solution is appropriate to their needs (this may be named individuals, groups or persona that describes the target end user). |
| 3.1.3 Research the problem | (a) Research the problem and solutions to similar problems to identify and justify suitable approaches to a solution.<br>(b) Describe the essential features of a computational solution explaining these choices.<br>(c) Explain the limitations of the proposed solution. |
| 3.1.4 Specify the proposed solution | (a) Specify and justify the solution requirements including hardware and software configuration (if appropriate).<br>(b) Identify and justify measurable success criteria for the proposed solution. |

| 3.2 Design of the solution (15 marks) | |
|---|---|
| 3.2.1 Decompose the problem | (a) Break down the problem into smaller parts suitable for computational solutions justifying any decisions made. |
| 3.2.2 Describe the solution | (a) Explain and justify the structure of the solution.<br>(b) Describe the parts of the solution using algorithms justifying how these algorithms form a complete solution to the problem.<br>(c) Describe usability features to be included in the solution.<br>(d) Identify key variables / data structures / classes justifying choices and any necessary validation. |
| 3.2.3 Describe the approach to testing | (a) Identify the test data to be used during the iterative development and post development phases and justify the choice of this test data. |

| 3.3 Developing the solution (25 marks) | |
|---|---|
| 3.3.1 Iterative development process | (a) Provide annotated evidence of each stage of the iterative development process justifying any decision made.<br>(b) Provide annotated evidence of prototype solutions justifying any decision made. |
| 3.3.2 Testing to inform development | (a) Provide annotated evidence for testing at each stage justifying the reason for the test.<br>(b) Provide annotated evidence of any remedial actions taken justifying the decision made. |

| 3.4 Evaluation (20 marks) | |
|---|---|
| 3.4.1 Testing to inform evaluation | (a) Provide annotated evidence of testing the solution of robustness at the end of the development process.<br>(b) Provide annotated evidence of usability testing (user feedback). |
| 3.4.2 Success of the solution | (a) Use the test evidence from the development and post development process to evaluate the solution against the success criteria from the analysis. |
| 3.4.3 Describe the final product | (a) Provide annotated evidence of the usability features from the design, commenting on their effectiveness. |
| 3.4.4 Maintenance and development | (a) Discuss the maintainability of the solution.<br>(b) Discuss potential further development of the solution. |

## 2d. Prior learning and progression

There are no prior qualification requirements for this specification.

Learners in England who are beginning an A level course are likely to have followed a Key Stage 4 programme of study. This course will enable learners to progress to higher study or to progress directly to employment.

This qualification is suitable for learners intending to pursue any career in which an understanding of technology is needed. The qualification is also suitable for any further study as part of a course of general education.

It will provide learners with a range of transferable skills which will facilitate personal growth and foster cross curriculum links in areas such as maths, science and design and technology. Computer Science is a very creative subject and skills such as problem solving and analytical thinking will all be refined and explored as learners progress through the learning and assessment programme.

There are a number of Computer Science specifications at OCR. Find out more at **www.ocr.org.uk**

# 3 Assessment of OCR A Level in Computer Science

## 3a. Forms of assessment

The A Level in Computer Science is a linear qualification with 100% terminal external assessment.

This qualification consists of two examined components (01 and 02), externally assessed by OCR and one internally assessed and moderated non exam assessment component (03 or 04).

Both examinations are of 2 hours and 30 minutes duration, each with a 40% weighting. The non-exam assessment component weighted at 20%.



**Computer systems (Component 01)**

Learners answer **all** the questions. There will be a mix of questions including short answer, longer answer and some higher tariff questions that will test the quality of extended responses. Marks for these responses are integrated into the marking criteria.

The whole of the Computer systems content will be covered over the life of the specification.

Questions may contain, for example, following and correcting algorithms and programs, software development and legal and moral issues.

**Algorithms and programming (Component 02)**

Learners answer **all** the questions in Section A and **all** questions in Section B. There will be a mix of questions including short answer, longer answer and some higher tariff questions that will test the quality of written responses via a level of response mark scheme.

The whole of the Algorithms and programming content will be covered over the life of the specification.

Section A will contain questions which may cover writing algorithms and computational methods, programming and programming techniques and problem solving. These questions may contain some shorter answer questions.

Section B will have a scenario set at the start of the section; this will contain information that will be used for the questions that follow. The questions will be largely of a higher tariff with problem solving algorithms and programming again forming the basis.

**Programming project (Component 03 or 04)**

The programming project will be submitted in the form of a report that will contain the solution to a problem, selected by the learner or centre, written in a suitable programming language. Appendix 5e reproduced in this document gives a list of programming languages which OCR will accept. If the task demands another choice of language that does not appear in the list, the task outline, the details of the programming language and the reasons for the choice of this language should be submitted to OCR for consideration. Please contact computerscience@ocr.org.uk for further advice and guidance. Within the report the learner must demonstrate their ability to analyse, design, develop, test and document using the principles learnt in computational thinking.

The report will be marked internally by the centre using the Programming project content in section 2c, in conjunction with the marking criteria in section 3e of this specification. It should then be sent to OCR for moderation.

Learners who are retaking the qualification may carry forward their mark for the non exam assessment component (see Section 4a for the relevant entry code (H446 C)).

## 3b. Assessment objectives (AO)

There are three assessment objectives in OCR's A Level in Computer Science. These are detailed in the table below.

Learners are expected to demonstrate their ability to:

| | Assessment Objectives |
|---|---|
| **AO1** | Demonstrate knowledge and understanding of the principles and concepts of computer science, including abstraction, logic, algorithms and data representation |
| **AO2** | Apply knowledge and understanding of the principles and concepts of computer science including to analyse problems in computational terms |
| **AO3** | Design, program and evaluate computer systems that solve problems, making reasoned judgements about these and presenting conclusions |



### AO weightings in A Level in Computer Science

The relationship between the assessment objectives and the components is shown in the following table:

| Component | % of A Level Computer Science (H446) | | | |
|---|---|---|---|---|
| | AO1 | AO2 | AO3 | Total |
| Computer systems (H446/01) | 21* | 9* | 10* | 40 |
| Algorithms and programming (H446/02) | 15* | 18* | 7* | 40 |
| Programming project (H446/03 or H446/04) | 0* | 3* | 17* | 20 |
| **Total** | **35*** | **30*** | **35*** | **100** |

* values rounded to the nearest whole %

## 3c. Assessment availability

There will be one examination series available each year in May/June to **all** learners.

This specification will be certificated from the June 2017 examination series onwards.

## 3d. Retaking the qualification

Learners can retake the qualification as many times as they wish. Learners must retake all examined components but they can choose to either retake the non-exam assessment (NEA) or carry forward (re-use) their most recent result (see Section 4d).

## 3e. Assessment of extended responses

The assessment materials for this qualification provide learners with the opportunity to demonstrate their ability to construct and develop a sustained line of reasoning and marks for extended responses are integrated into the marking criteria.

## 3f. Non exam assessment

**Programming project (Component 03 or 04) Guidance**

This qualification has one non exam assessment which takes the form of the Programming project (Component 03 or 04).

The project is a substantial piece of work which assesses a variety of different skills including the development and demonstration of computational thought processes. The following guidance should be considered before learners embark on this particular assessment.

**Acceptable programming languages**

Appendix 5e in this specification gives a list of programming languages which OCR will accept. If the task demands another choice of language that does not appear in the list, the task outline, the details of the programming language and the reasons for the choice of this language should be submitted to OCR for consideration. Please contact computerscience@ocr.org.uk for further advice and guidance.

**Context Choice**

Learners will need to choose in liaison with their teacher a well defined user driven problem. Whilst OCR will not be quality assuring the contexts chosen by learners the following criteria should be considered. The choice of project must enable learners to:

- meet all assessment requirements as contained in the specification
- use an appropriate programming language which is non-trivial and has a substantial coded element (see appendix 5e)
- solve a problem sensibly within the constraints of resources available
- facilitate the successful completion of a whole task from its definition to its acceptance and evaluation by that user
- involve all elements of the skills of analysis, design, development and evaluation.

**Evidence Generation**

It is important that learners establish thorough and robust working practices at an early stage. The projects should contain as standard the following:

| | |
|---|---|
| Title | ✓ |
| Contents list | ✓ |
| Description of investigation | ✓ |
| Justification of investigation | ✓ |
| Analysis, design and methods used | ✓ |
| Evaluation | ✓ |
| Bibliography | ✓ |
| Pages numbered | ✓ |

Appropriate annotated evidence should be used to support the report – e.g. screen dumps or photographs taken of screen layouts, for example. It is important that each learner generates their own individual, authentic evidence to show that they have met key assessment criteria. Any evidence submitted to demonstrate the development of solution must be able to be assessed without the use of any specific hardware or software.

Whilst there is no word count for this particular assessment learners need to focus on the command words used within the assessment criteria. These command words indicate the depth of coverage that is required at each stage of the process and should drive the evidencing approach taken by learners.

The teacher has an important role in supporting learners through the evidence generation process. Whilst it is not appropriate for teachers to over direct learners in their generation of evidence it is acceptable for them to reinforce concepts and assist with the development of knowledge and understanding at various stages of the process. Acceptable forms of intervention include:

- offering learners advice about how best to approach each of the tasks
- exercising continuous supervision of work in order to monitor progress and to prevent plagiarism
- ensuring work is completed in accordance with the specification requirements and can be assessed in accordance with the specified marking criteria and procedures.

**Assessing Evidence**

It is important that centre staff responsible for marking learner projects are familiar with the principles of best fit assessment to ensure that a consistent marking approach is taken.

For each of the marking criteria assessors select the band descriptor provided in the marking grid that most closely describes the quality of the work being marked.

Marking should be positive, rewarding achievement rather than penalising failure or omissions.

The award of marks must be directly related to the marking criteria.

- Each band descriptor covers all of the relevant criteria for a particular section of the specification content related to analysis, design, development and evaluation.
- The descriptors for each section should be read and applied as a whole.
- An answer does not have to meet all of the requirements of a band descriptor before being placed in that band. It will be placed in a particular band when it meets more of the requirements of that band than it meets the requirements of other bands.

When deciding the mark within a band, the following criteria should be applied:

- the extent to which the statements within the band have been achieved.

For example:

- an answer that convincingly meets nearly all of the requirements of a band descriptor should be placed at or near the top of that band. Where the learner's work convincingly meets the statement, the highest mark should be awarded
- an answer that meets many of the requirements of the band descriptor should be placed in the middle of the band. Where the learner's work adequately meets the statement, the most appropriate mark in the middle range should be awarded
- if an answer is on the border-line between two bands but it is decided that it fits better the descriptors for the lower of these two bands, then it should be placed near the top of that band. Where the learner's work just meets the statement, the lowest mark should be awarded.

Assessors should use the full range of marks available to them and award full marks in any band of work that fully meets the descriptor. This is work that is 'the best one could expect from learners working at that level'.

Learners or centres must declare that the work is the learner's own (see section 4d).

**Authentication**

It is important that centres put in place the necessary measures to both guard against and detect malpractice when it is has occurred. As well as the use of the authentication process it is important that learners take responsibility for ensuring the work they produce is their own, is appropriately referenced and is a true reflection of their ability (see Section 4d).

**Ongoing Support**

OCR will support teachers in the teaching and learning of this component by providing an extensive range of support material including sample assessment materials and schemes of work to provide exemplar ideas. These resources will be available online to support the launch of the qualification **www.ocr.org.uk**

## Assessment criteria

3

Learners are expected to demonstrate their ability to analyse, design, develop, test, evaluate and document a program written in a suitable programming language. The problem must be solvable effectively in the chosen language so it is important there is a good match between the requirements of the problem and the language that is chosen.

The nature of the problem chosen is significant in that learners should be able to demonstrate, through their solutions, an appropriate range of skills identified by the specification; trivial problems, no matter how well they are executed will not be able to provide this evidence.

Learners are expected to apply the computational thinking approach identified in Algorithms and programming (Component 02) to a practical coding problem and are expected to apply principles from an agile development approach to the project development.

The project should be assessed holistically and, while the assessment criterion are organised into discrete sections, it is unlikely evidence can be similarly organised in the learner's work. Evidence to support the assessment may be found throughout the project report and should be noted, by reference to page number, or otherwise, on the assessment grid. The assessment criterion are defined in mark bands and teachers should use a best-fit approach, selecting the most appropriate description and quality of match to that description, in order to place the work at the right level in the right mark band.

## Programming project (Component 03 or 04) marking criteria – 70 marks

| AO 2.2 Analysis (maximum 10 marks) | | | |
|---|---|---|---|
| **1–2 marks** | **3–5 marks** | **6–8 marks** | **9–10 marks** |
| **The candidate will have:** | | | |
| • Identified some features that make the problem solvable by computational methods.<br>• Identified suitable stakeholders for the project and described them and some of their requirements.<br>• Identified some appropriate features to incorporate into their solution.<br>• Identified some features of the proposed computational solution.<br>• Identified some limitations of the proposed solution.<br>• Identified some requirements for the solution.<br>• Identified some success criteria for the proposed solution. | • Described the features that make the problem solvable by computational methods.<br>• Identified suitable stakeholders for the project and described how they will make use of the proposed solution.<br>• Researched the problem looking at existing solutions to similar problems identifying some appropriate features to incorporate into their solution.<br>• Identified the essential features of the proposed computational solution.<br>• Identified and described some limitations of the proposed solution.<br>• Identified most requirements for the solution.<br>• Identified some measurable success criteria for the proposed solution. | • Described the features that make the problem solvable by computational methods and why it is amenable to a computational approach.<br>• Identified suitable stakeholders for the project and described them and how they will make use of the proposed solution and why it is appropriate to their needs.<br>• Researched the problem in depth looking at existing solutions to similar problems identifying and describing suitable approaches based on this research.<br>• Identified and described the essential features of the proposed computational solution.<br>• Identified and explained any limitations of the proposed solution.<br>• Specified the requirements for the solution including (as appropriate) any hardware and software requirements.<br>• Identified measurable success criteria for the proposed solution. | • Described and justified the features that make the problem solvable by computational methods, explaining why it is amenable to a computational approach.<br>• Identified suitable stakeholders for the project and described them explaining how they will make use of the proposed solution and why it is appropriate to their needs.<br>• Researched the problem in depth looking at existing solutions to similar problems, identifying and justifying suitable approaches based on this research.<br>• Identified the essential features of the proposed computational solution explaining these choices.<br>• Identified and explained with justification any limitations of the proposed solution.<br>• Specified and justified the requirements for the solution including (as appropriate) any hardware and software requirements.<br>• Identified and justified measurable success criteria for the proposed solution. |

0 marks = no response or no response worthy of credit.

| AO 3.1 Design (maximum 15 marks) | | | |
|---|---|---|---|
| **1–4 marks** | **5–8 marks** | **9–12 marks** | **13–15 marks** |
| **The candidate will have:** | | | |
| • Described elements of the solution using algorithms.<br>• Described some usability features to be included in the solution.<br>• Identified the key variables / data structures / classes (as appropriate to the proposed solution).<br>• Identified some test data to be used during the iterative or post development phase of the process.<br>• | • Broken the problem down systematically into a series of smaller problems suitable for computational solutions describing the process.<br>• Defined the structure of the solution to be developed.<br>• Described the solution fully using appropriate and accurate algorithms.<br>• Described the usability features to be included in the solution.<br>• Identified the key variables / data structures / classes (as appropriate to the proposed solution) and any necessary validation.<br>• Identified the test data to be used during the iterative development of the solution.<br>• Identified any further data to be used in the post development phase. | • Broken the problem down systematically into a series of smaller problems suitable for computational solutions explaining the process.<br>• Defined in detail the structure of the solution to be developed.<br>• Described the solution fully using appropriate and accurate algorithms explaining how these algorithms form a complete solution to the problem.<br>• Described, explaining choices made, the usability features to be included in the solution.<br>• Identified and justified the key variables / data structures / classes (as appropriate to the proposed solution) explaining any necessary validation.<br>• Identified and justified the test data to be used during the iterative development of the solution.<br>• Identified and justified any further data to be used in the post development phase. | • Broken the problem down systematically into a series of smaller problems suitable for computational solutions, explaining and justifying the process.<br>• Defined in detail the structure of the solution to be developed.<br>• Described the solution fully using appropriate and accurate algorithms justifying how these algorithms form a complete solution to the problem.<br>• Described, justifying choices made, the usability features to be included in the solution.<br>• Identified and justified the key variables / data structures / classes (as appropriate to the proposed solution) justifying and explaining any necessary validation.<br>• Identified and justified the test data to be used during the iterative development of the solution.<br>• Identified and justified any further data to be used in the post development phase. |

0 marks = no response or no response worthy of credit.

| AO 3.2 Developing the coded solution (maximum 25 marks) | | | |
|---|---|---|---|
| **Iterative development of a coded solution (maximum 15 marks)** | | | |
| **1–4 marks** | **5–8 marks** | **9–12 marks** | **13–15 marks** |
| **The candidate will have:** | | | |
| • Provided evidence of some iterative development for a coded solution.<br>• Solution may be linear.<br>• Code may be inefficient.<br>• Code may not be annotated appropriately.<br>• Variable names may be inappropriate.<br>• There will be little or no evidence of validation.<br>• There will be little evidence of review during the development. | • Provided evidence for most stages of the iterative development process for a coded solution describing what they did at each stage.<br>• Solution will have some structure.<br>• Code will be briefly annotated to explain key components.<br>• Some variable and/or structure names will be largely appropriate.<br>• There will be evidence of some basic validation.<br>• There will be evidence that the development was reviewed at some stage during the process. | • Provided evidence of each stage of the iterative development process for a coded solution relating this to the break down of the problem from the analysis stage and explaining what they did at each stage.<br>• Provided evidence of some prototype versions of their solution.<br>• The solution will be modular in nature.<br>• Code will be annotated to explain all key components.<br>• Most variables and structures will be appropriately named.<br>• There will be evidence of validation for most key elements of the solution.<br>• The development will show review at most key stages in the process. | • Provided evidence of each stage of the iterative development process for a coded solution relating this to the break down of the problem from the analysis stage and explaining what they did and justifying why.<br>• Provided evidence of prototype versions of their solution for each stage of the process.<br>• The solution will be well structured and modular in nature.<br>• Code will be annotated to aid future maintenance of the system.<br>• All variables and structures will be appropriately named.<br>• There will be evidence of validation for all key elements of the solution.<br>• The development will show review at all key stages in the process. |
| **Testing to inform development (maximum 10 marks)** | | | |
| **1–2 marks** | **3–5 marks** | **6–8 marks** | **9–10 marks** |
| **The candidate will have:** | | | |
| • Provided some evidence of testing during the iterative development process. | • Provided some evidence of testing during the iterative development process.<br>• Provided evidence of some failed tests and the remedial actions taken. | • Provided evidence of testing at most stages of the iterative development process.<br>• Provided evidence of some failed tests and the remedial actions taken with some explanation of the actions taken. | • Provided evidence of testing at each stage of the iterative development process.<br>• Provided evidence of any failed tests and the remedial actions taken with full justification for any actions taken. |

0 marks = no response or no response worthy of credit.

| AO 3.3 Evaluation (maximum 20 marks) | | | |
|---|---|---|---|
| **Testing to inform evaluation (maximum 5 marks)** | | | |
| **1 mark** | **2 marks** | **3–4 marks** | **5 marks** |
| **The candidate will have:** | | | |
| • Provided evidence of some post development testing. | • Provided evidence of final product testing for function. | • Provided annotated evidence of post development testing for function.<br>• Provided annotated evidence for usability testing. | • Provided annotated evidence of post development testing for function and robustness.<br>• Provided annotated evidence for usability testing. |
| **Evaluation of solution (maximum 15 marks)** | | | |
| **1–4 marks** | **5–8 marks** | **9–12 marks** | **13–15 marks** |
| **The candidate will have:** | | | |
| • Commented on the success or failure of the solution with some reference to test data.<br>• The information is basic and communicated in an unstructured way. The information is supported by limited evidence and the relationship to the evidence may not be clear. | • Cross referenced some of the test evidence with the success criteria and commented on the success or otherwise of the solution.<br>• Provided evidence of usability features.<br>• Identified some limitations on the solution.<br>• The information has some relevance and is presented with limited structure. The information is supported by limited evidence. | • Used the test evidence to cross reference with the success criteria to evaluate the solution identifying whether the criteria have been met, partially met or unmet.<br>• Provided comments on how any partially or not met criteria could be addressed in further development.<br>• Provided evidence of the usability features.<br>• Considered maintenance issues and limitations of the solution.<br>• There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence. | • Used the test evidence to cross reference with the success criteria to evaluate the solution explain how the evidence shows that the criteria has been fully, partially or not met in each case.<br>• Provided comments on how any partially or unmet criteria could be addressed in further development.<br>• Provided evidence of the usability features justifying their success, partial success or failure as effective usability features.<br>• Provided comments on how any issues with partially or unmet usability features could be addressed in further development.<br>• Considered maintenance issues and limitations of the solution.<br>• Described how the program could be developed to deal with limitations and potential improvements / changes.<br>• There is a well developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated. |

0 marks = no response or no response worthy of credit.

## 3g. Synoptic assessment

Synoptic assessment draws together the knowledge, understanding and skills learnt in different aspects of the A level computing course.

The emphasis of synoptic assessment is to encourage the understanding of Computer Science as a discipline.

It is envisaged that the Computer systems component (01) would be taught first but there are elements of the Algorithms and programming component (02) that should also be considered as computer systems are taught in order that learners are fully aware of the way problem solving through computational thinking is achieved.

The Algorithms and programming component (02) builds on the knowledge and understanding gained in computer systems (01). Algorithms and programming will contain synoptic assessment.

The programming project component (03) or (04) draws on all the elements of the other components in order to enable learners to develop synoptic thought processes in their approach to the solving of a real life scenario. As part of this process, learners will draw on knowledge, understanding and skills that have been embedded in components 01 and 02.



## 3h. Calculating qualification results

A learner's overall qualification grade for an A Level in Computer Science will be calculated by adding together their marks from the three components taken to give their total weighted mark.

This mark will then be compared to the qualification level grade boundaries for the entry option taken by the learner and for the relevant exam series to determine the learner's overall qualification grade.

# 4 Admin: what you need to know

The information in this section is designed to give an overview of the processes involved in administering this qualification so that you can speak to your exams officer. All of the following processes require you to submit something to OCR by a specific deadline.

More information about the processes and deadlines involved at each stage of the assessment cycle can be found in the Administration area of the OCR website. OCR's *Admin overview* is available on the OCR website: http://www.ocr.org.uk/administration

## 4a. Pre-assessment

### Estimated entries

Estimated entries are your best projection of the number of learners who will be entered for a qualification in a particular series.

Estimated entries should be submitted to OCR by the specified deadline. They are free and do not commit your centre in any way.



### Final entries

Final entries provide OCR with detailed data for each learner, showing each assessment to be taken. It is essential that you use the correct entry code, considering the relevant entry rules and ensuring that you choose the entry option for the moderation method you intend to use.

Final entries must be submitted to OCR by the published deadlines or late entry fees will apply.

All learners taking A Level in Computer Science must be entered for one of the following entry options:

| Entry option | | Components | | |
|---|---|---|---|---|
| Entry code | Title | Code | Title | Assessment type |
| H446 A | Computer Science (OCR Repository) | 01 | Computer systems | External Assessment |
| | | 02 | Algorithms and programming | External Assessment |
| | | 03 | Programming project (OCR Repository) | Non exam assessment (OCR Repository) |
| H446 B | Computer Science (Postal moderation) | 01 | Computer systems | External Assessment |
| | | 02 | Algorithms and programming | External Assessment |
| | | 04 | Programming project (Postal moderation) | Non exam assessment (Postal moderation) |
| H446 C* | Computer Science Non exam assessment (carried forward) | 01 | Computer systems | External Assessment |
| | | 02 | Algorithms and programming | External Assessment |
| | | 80 | Programming project (carried forward) | Non exam assessment (carried forward) |

*Entry option H446 C should only be selected for learners who are retaking the qualification and who want to carry forward their mark for the non exam assessment.

## 4b. Accessibility and special consideration

Reasonable adjustments and access arrangements allow learners with special educational needs, disabilities or temporary injuries to access the assessment and show what they know and can do, without changing the demands of the assessment. Applications for these should be made before the examination series. Detailed information about eligibility for access arrangements can be found in the JCQ *Access Arrangements and Reasonable Adjustments*.

Special consideration is a post-assessment adjustment to marks or grades to reflect temporary injury, illness or other indisposition at the time the assessment was taken.

Detailed information about eligibility for special consideration can be found in the JCQ *A guide to the special consideration process*.

## 4c. External assessment arrangements

Regulations governing examination arrangements are contained in the JCQ *Instructions for conducting examinations*.

Learners are not permitted to use a scientific or graphical calculator for both components. Calculators are subject to the rules in the document *Instructions for Conducting Examinations* published annually by JCQ (**www.jcq.org.uk**).

### Head of Centre Annual Declaration

The Head of Centre is required to provide a declaration to the JCQ as part of the annual NCN update, conducted in the autumn term, to confirm that the centre is meeting all of the requirements detailed in the specification.

Any failure by a centre to provide the Head of Centre Annual Declaration will result in your centre status being suspended and could lead to the withdrawal of our approval for you to operate as a centre.

### Private Candidates

Private candidates may enter for OCR assessments.

A private candidate is someone who pursues a course of study independently but takes an examination or assessment at an approved examination centre. A private candidate may be a part-time student, someone taking a distance learning course, or someone being tutored privately. They must be based in the UK.

Private candidates need to contact OCR approved centres to establish whether they are prepared to host them as a private candidate. The centre may charge for this facility and OCR recommends that the arrangement is made early in the course.

Further guidance for private candidates may be found on the OCR website: **http://www.ocr.org.uk**

## 4d. Non exam assessment

Regulations governing arrangements for internal assessments are contained in the JCQ *Instructions for conducting non-examination assessment*.

### Authentication of learners' work

Candidates and centres must declare that the work submitted for assessment is the candidate's own by completing a centre authentication form (CCS160) for the course requirements. This information must be retained at the centre and be available on request to either OCR or the JCQ centre inspection service. It must be kept until the deadline has passed for centres to submit an enquiry about results (EAR). Once this deadline has passed and centres have not requested an EAR, this evidence can be destroyed.



### Internal standardisation

Centres must carry out internal standardisation to ensure that marks awarded by different teachers are accurate and consistent across all learners entered for the component from that centre.

### Moderation

The purpose of moderation is to bring the marking of internally-assessed components in all participating centres to an agreed standard. This is achieved by checking a sample of each centre's marking of learners' work.

Following internal standardisation, centres submit marks to OCR and the moderator. If there are fewer than 10 learners, all the work should be submitted for moderation at the same time as marks are submitted.

Once marks have been submitted to OCR and your moderator, centres will receive a moderation sample request. Samples will include work from across the range of attainment of the learners' work.

There are two ways to submit a sample:

**Moderation via the OCR Repository** – Where you upload electronic copies of the work included in the sample to the OCR Repository and your moderator accesses the work from there.

**Postal moderation** – Where you post the sample of work to the moderator.

The method that will be used to submit the moderation sample must be specified when making entries. The relevant entry codes are given in Section 4a.

All learners' work must be submitted using the same entry option. It is not possible for centres to offer both options within the same series.

Centres will receive the outcome of moderation when the provisional results are issued. This will include:

**Moderation Adjustments Report** – Listing any scaling that has been applied to internally-assessed components.

**Moderator Report to Centres** – A brief report by the moderator on the internal assessment of learners' work.

### Carrying forward non-exam assessment (NEA)

Learners who are retaking the qualification can choose to either retake the non-exam assessment – Programming project (03/04), or carry forward their most recent result for that component.

To carry forward the NEA component result, you must use the correct carry forward entry option (see table in Section 4a).

Learners must decide at the point of entry whether they are going to carry forward the NEA result or not.

The result for the NEA component may be carried forward for the lifetime of the specification and there is no restriction on the number of times the result may be carried forward. However, only the most recent non-absent result may be carried forward.

When the result is carried forward, the grade boundaries from the previous year of entry will be used to calculate a new weighted mark for the carried forward component, so the value of the original mark is preserved.

## 4e. Results and certificates



### Grade Scale

A level qualifications are graded on the scale: A*, A, B, C, D, E, where A* is the highest. Learners who fail to reach the minimum standard for E will be Unclassified (U). Only subjects in which grades A* to E are attained will be recorded on certificates.

### Results

Results are released to centres and learners for information and allow any queries to be resolved **before** certificates are issued.

Centres will have access to the following results information for each learner:

- the grade for the qualification
- the raw mark for each component
- the total weighted mark for the qualification.

The following supporting information will be available:

- raw mark grade boundaries for each component
- weighted mark grade boundaries for the qualification.

Until certificates are issued, results are deemed to be provisional and may be subject to amendment. A learner's final results will be recorded on an OCR certificate.

The qualification title will be shown on the certificate as 'OCR Level 3 Advanced GCE in Computer Science'.

## 4f. Post-results services

A number of post-results services are available:

- **Enquiries about results** – If you are not happy with the outcome of a learner's results, centres may submit an enquiry about results.
- **Missing and incomplete results** – This service should be used if an individual subject result for a learner is missing, or the learner has been omitted entirely from the results supplied.
- **Access to scripts** – Centres can request access to marked scripts.

## 4g. Malpractice

Any breach of the regulation for the conduct of examinations and coursework may constitute malpractice (which includes maladministration) and must be reported to OCR as soon as it is detected.

Detailed information on malpractice can be found in the JCQ publication *Suspected Malpractice in Examinations and Assessments: Policies and Procedures*.

# 5 Appendices

## 5a. Overlap with other qualifications

The knowledge, understanding and skills that are developed throughout this qualification are distinct and have very little overlap with other qualifications.

This overlap may occur only at level 5 in the hardware and software elements of ICT Cambridge Technicals and GCE Applied ICT.

## 5b. Avoidance of bias

The A level qualification and subject criteria have been reviewed in order to identify any feature which could disadvantage candidates who share a protected characteristic as defined by the Equality Act 2010. All reasonable steps have been taken to minimise any such disadvantage.

## 5c. Mathematical skills

Computer Science uses mathematics to express its computational laws and processes.

All AS level and A level Computer Science qualifications must contain a minimum of 10% mathematical skills. Candidates may be asked to demonstrate their knowledge, understanding and skills of computational processes and problem solving in both theoretical and practical ways. The following list of topics will be counted as Level 2 (or higher) mathematics.

Topic:

- Boolean algebra
- comparison of complexity of algorithms
- number representation and bases

Whilst the concept for each topic is Level 2 (though it may not appear in GCSE mathematics specifications) candidates will, however be expected to apply the skills in a Level 3 context.

## 5d. Languages and Boolean logic guide for use in external assessments

The tables below show languages and logic that will be used in the external assessments and indicates the limits and scope of each.

Centres are free to go beyond these parameters.

### Pseudocode

The following guide shows the format pseudocode will appear in the examined components. It is provided to allow you to give learners familiarity before the exam. Learners are not expected to memorise the syntax of this pseudocode and when asked may provide answers in any style of pseudocode they choose providing its meaning could be reasonably inferred by a competent programmer.

**Variables**
Variables are assigned using the = operator.

```
x=3
name="Bob"
```

A variable is declared the first time a value is assigned. It assumes the data type of the value it is given.

Variables declared inside a function or procedure are local to that subroutine.

Variables in the main program can be made global with the keyword `global`.

```
global userid = 123
```

**Casting**
Variables can be typecast using the `int str` and `float` functions.

`str(3)` returns "3"
`int ("3")` returns 3
`float ("3.14")` returns `3.14`

**Outputting to Screen**
`print(`*string*`)`

5

Example

```
print("hello")
```

Taking Input from User
*variable*`=input(`*prompt to user*`)`

Example

```
name=input("Please enter your name")
```

**Iteration – Count Controlled**

```
for i=0 to 7
    print("Hello")
next i
```

Will print hello 8 times (0–7 inclusive).

**Iteration – Condition Controlled**

```
while answer!="computer"
    answer=input("What is the password?")
endwhile
```

```
do
    answer=input("What is the password?")
until answer=="computer"
```

### Logical Operators

```
AND OR NOT
```

```
e.g.
while x<=5 AND flag==false
```

Comparison Operators

| | |
|---|---|
| == | Equal to |
| != | Not equal to |
| < | Less than |
| <= | Less than or equal to |
| > | Greater than |
| >= | Greater than or equal to |

Arithmetic Operators

| | |
|---|---|
| + | Addition e.g. x=6+5 gives 11 |
| – | Subtraction e.g. x=6–5 gives 1 |
| * | Multiplication e.g. x=12*2 gives 24 |
| / | Division e.g. x=12/2 gives 6 |
| MOD | Modulus e.g. 12MOD5 gives 2 |
| DIV | Quotient e.g. 17DIV5 gives 3 |
| ^ | Exponentiation e.g. 3^4 gives 81 |



### Selection

Selection will be carried out with if/else and switch/case

if/else

```
if entry==”a” then
     print(“You selected A”)
elseif entry==”b” then
     print(“You selected B”)
else
     print(“Unrecognised selection”)
endif
```

switch/case

```
switch entry:
      case “A”:
         print(“You selected A”)
      case “B”:1
         print(“You selected B”)
      default:
         print(“Unrecognised selection”)

endswitch
```

### String Handling

To get the length of a string:

```
stringname.length
```

To get a substring:

```
stringname.subString(startingPosition, numberOfCharacters)
```

NB The string will start with the 0th character.

Example

```
someText="Computer Science"

print(someText.length)
print(someText.substring(3,3))
```

Will display

```
16
put
```

5

### Subroutines

```
function triple(number)
    return number*3
endfunction
```

Called from main program

```
y=triple(7)
```

```
procedure greeting(name)
    print("hello"+name)
endprocedure
```

Called from main program

```
greeting("Hamish")
```

**Unless stated values passed to subroutines can be assumed to be passed by value.**
**If this is relevant to the question byVal and byRef will be used. In the case below x is passed by value and y is passed by reference.**

```
procedure foobar(x:byVal, y:byRef)
    …
    …
endprocedure
```

### Arrays

Arrays will be 0 based and declared with the keyword *array*.

```
array names[5]
names[0]="Ahmad"
names[1]="Ben"
names[2]="Catherine"
names[3]="Dana"
names[4]="Elijah"

print(names[3])
```

Example of 2D array:

```
Array board[8,8]
board[0,0]="rook"
```

### Reading to and Writing from Files

To open a file to read from openRead is used and readLine to return a line of text from the file.

The following program makes x the first line of sample.txt

```
myFile = openRead("sample.txt")
x = myFile.readLine()
myFile.close()
```

`endOfFile()` is used to determine the end of the file. The following program will print out the contents of sample.txt

```
myFile = openRead("sample.txt")
while NOT myFile.endOfFile()
    print(myFile.readLine())
endwhile
myFile.close()
```

To open a file to write to openWrite is used and writeLine to add a line of text to the file. In the program below hello world is made the contents of sample.txt (any previous contents are overwritten).

```
myFile = openWrite("sample.txt")
myFile.writeLine("Hello World")
myFile.close()
```

### Comments

Comments are denoted by //

```
print("Hello World") //This is a comment
```

## Object-Oriented

Object oriented code will match the pseudocode listed above with the following extensions:

### Methods and Attributes:

Methods and attributes can be assumed to be public unless otherwise stated. Where the access level is relevant to the question it will always be explicit in the code denoted by the keywords.

```
public and private.

private attempts = 3

public procedure setAttempts(number)
    attempts=number
endprocedure

public function getAttempts()
    return attempts
endfunction
```

Methods will always be instance methods, learners aren't expected to be aware of static methods. They will be called using object.method so

```
player.setAttempts(5)

print(player.getAttempts())
```

Constructors and Inheritance
Inheritance is denoted by the `inherits` keyword, superclass methods will be called with the keyword super. i.e. `super.methodName(`*parameters*`)` in the case of the constructor this would be `super.new()`
Constructors will be procedures with the name new.

```
class Pet

    private name
     public procedure new(givenName)
        name=givenName

     endprocedure

endclass


class Dog inherits Pet

    private breed

     public procedure new(givenName, givenBreed)
        super.new(givenName)
        breed=givenBreed
     endprocedure

endclass
```

**Constructors and Inheritance**

Constructors will be procedures with the name new.

```
class Pet

    private name
    public procedure new(givenName)
        name=givenName

    endprocedure

endclass
```

Inheritance is denoted by the `inherits` keyword, superclass methods will be called with the keyword super. i.e. `super.methodName(`*parameters*`)` in the case of the constructor this would be `super.new()`

```
class Dog inherits Pet

    private breed

    public procedure new(givenName, givenBreed)
       super.new(givenName)
       breed=givenBreed
    endprocedure

endclass
```

To create an instance of an object the following format is used

*objectName* = new *className*(*parameters*)

e.g.

```
myDog = new Dog("Fido", "Scottish Terrier")
```

## HTML

Learners are expected to have an awareness of the following tags. Any other tags used will be introduced in the question.

<html>
<link> to link to a CSS file
<head>

<title>

<body>

<h1> <h2> <h3>

<img> including the src, alt, height and width attributes.

<a> including the href attribute.

<form>

<input> where the input is a textbox (i.e. has the attribute type="text" and another attribute name to identify it) or a submit button (i.e. has the attribute type="submit")

<p>

<li>

<ol>

<ul>

<script>

Any other elements used will be explained in the question.

## CSS

Learners are expected to be able to use CSS directly inside elements using the style attribute

```
<h1 style="color:blue;">
```

and external style sheets. In the style sheets they should be able to use CSS to define the styling of elements:

```
h1{
    color:blue;
}
```

classes

```
.infoBox{
    background-color: green;
}
```

and Identifiers

```
#menu{
    background-color: #A2441B;
}
```

They are expected to be familiar with the following properties.

```
background-color
border-color
border-style
border-width
color with named and hex colours
font-family
font-size
height
width
```

Any other properties used will be explained in the question.

## JavaScript

Learners are expected to be able to follow and write basic JavaScript code. It is hoped they will get practical experience of JavaScript in their study of the course. They will not be expected to commit exact details of syntax to memory. Questions in the exam will not penalise learners for minor inaccuracies in syntax. Learners *will* be expected to be familiar with the JavaScript equivalents of the structures listed in the pseudocode section (with the exception of input and output (see below). They will not be expected to use JavaScript for Object Oriented programming or file handling. Questions will not be asked in JavaScript where something is passed to a subroutine by value or reference is relevant.

### Input

Input will be taken in by reading values from a form. *NB learners will not be expected to memorise the method for doing this as focus will be on what they do with that input once it is received.*

### Output

By changing the contents of an HTML element

```
chosenElement = document.getElementById("example");
chosenElement.innerHTML = "Hello World";
```

By writing directly to the document

```
document.write("Hello World");
```

By using an alert box

```
alert("Hello World");
```

Any other JavaScript used will be explained in the question.

## Little Man Computer Instruction Set

In questions mnemonics will always be given according to the left hand column below. Different implementations of LMC have slight variations in mnemonics used and to take this into account the alternative mnemonics in the right hand column will be accepted in learners' answers.

| Mnemonic | Instruction | Alternative mnemonics accepted |
|---|---|---|
| ADD | Add | |
| SUB | Subtract | |
| STA | Store | STO |
| LDA | Load | LOAD |
| BRA | Branch always | BR |
| BRZ | Branch if zero | BZ |
| BRP | Branch if positive | BP |
| INP | Input | IN, INPUT |
| OUT | Output | |
| HLT | End program | COB, END |
| DAT | Data location | |

## Structured Query Language (SQL)

Learners will be expected to be familiar with the structures below. Should any other aspects of SQL be used they will be introduced and explained in the question.

`SELECT` (including nested SELECTs)

`FROM`

`WHERE`

`LIKE`

`AND`

`OR`

`DELETE`

`INSERT`

`DROP`

5

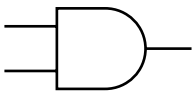
`JOIN` (Which is equivalent to INNER JOIN, there is no expectation to know about outer, left and right joins)

`WILDCARDS` (Learners should be familiar in the use of '*' and '%' as a wildcard to facilitate searching and matching where appropriate)

## Boolean Algebra

When Boolean algebra is used in questions the notation described below will be used. Other forms of notation exist and below are also a list of accepted notation we will accept from learners.

### Conjunction



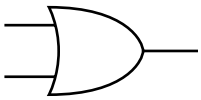
Notation used:
$\wedge$ e.g. A $\wedge$ B

| A | B | A $\wedge$ B |
|---|---|---|
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | F |

Alternatives accepted:

AND e.g. A AND B
e.g. A.B

### Disjunction



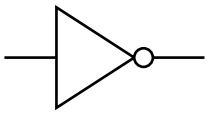
Notation used:
$\vee$ e.g. A $\vee$ B

| A | B | A $\vee$ B |
|---|---|---|
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |

Alternatives accepted:

OR e.g. A OR B
+ e.g. A+B

### Negation

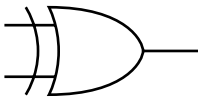


Notation used:
¬ e.g. ¬A

| A | ¬A |
|---|---|
| T | F |
| F | T |

Alternatives Accepted:
*bar* e.g. $\overline{A}$
~ e.g. ~A
NOT e.g. NOT A

### Exclusive Disjunction



Notation used:
$\veebar$ e.g. A$\veebar$B

| A | B | A$\veebar$B |
|---|---|---|
| T | T | F |
| T | F | T |
| F | T | T |
| F | F | F |

Alternatives accepted:

XOR e.g. A XOR B
$\oplus$ e.g. A $\oplus$B

5

### Equivalence / Iff

Notation used:
$\equiv$ e.g. $(A \barwedge B) \equiv \neg(\neg A \barvee \neg B)$

Alternatives accepted:
$\leftrightarrow$

## 5e. Acceptable programming languages for the Programming project (03)

The language chosen for the Programming project (03) should be appropriate to the task chosen. If the task demands another choice of language that does not appear in the list below, the task outline, the details of the programming language and the reasons for the choice of this language should be submitted to OCR for consideration computerscience@ocr.org.uk. All tasks completed in all languages need to have a suitable graphical interface.

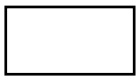
Programming languages which OCR will accept:

- Python
- C family of languages (for example C# C+ etc.)
- Java
- Visual Basic
- PHP
- Delphi

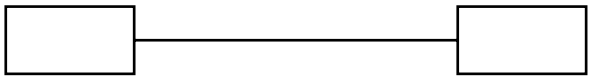
## 5f. Entity relationship diagrams

The following symbols are used for entities and their relationships.

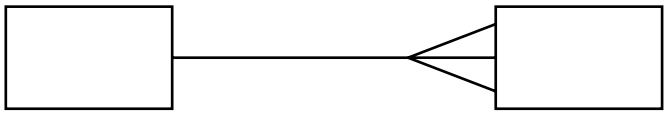
Entity



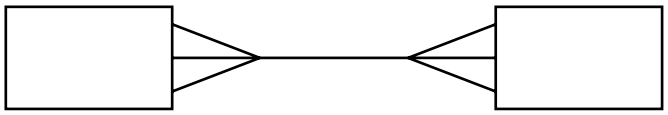
One-To-One relationship



One-To-Many relationship



Many-To-Many relationship



## Summary of updates

| Date | Version | Section | Title of section | Change |
|---|---|---|---|---|
| December 2017 | 2 | Multiple | – | Changes to generic wording and OCR website links throughout the specification. No changes have been made to any assessment requirements. |
| May 2018 | 2.1 | Front cover | Disclaimer | Addition of Disclaimer |
| August 2018 | 2.2 | 3d<br><br>4d | Retaking the qualification<br><br>Non exam assessment | Update to wording for carry forward rules |
| October 2019 | 2.3 | 5d | Languages and Boolean logic guide for use in external assessments | Error corrected in coding example on page 36. |

# Your checklist

## Our aim is to provide you with all the information and support you need to deliver our specifications.

- ☐ Bookmark ocr.org.uk/alevelcomputerscience for all the latest resources, information and news on AS and A Level Computer Science A
- ☐ Be among the first o hear about support materials and resources as they become available – register for Computer Science updates at ocr.org.uk/updates
- ☐ Find out about our professional development at cpdhub.ocr.org.uk
- ☐ View our range of skills guides for use across subjects and qualific tions at ocr.org.uk/skillsguides
- ☐ Discover our new online past paper service at ocr.org.uk/exambuilder
- ☐ Learn more about Active Results at ocr.org.uk/activeresults
- ☐ Join our Computer Science social network community for teachers at social.ocr.org.uk

## Download high-quality, exciting and innovative AS and A Level Computer Science resources from ocr.org.uk/alevelcomputerscience

Free resources and support for our A Level Computer Science qualification, developed through collaboration between our Computer Science Subject Advisors, teachers and other subject experts, are available from our website. You can also contact our Computer Science Subject Advisors for specialist advice, guidance and support, giving you individual service and assistance whenever you need it.

Contact the team at:
01223 553998
computerscience@ocr.org.uk
@OCR_ict

To stay up to date with all the relevant news about our qualifications, register for email updates at ocr.org.uk/updates

### Science community

The social network is a free platform where teachers can engage with each other – and with us – to find and offer guidance, discover and share ideas, best practice and a range of Science support materials. To sign up, go to social.ocr.org.uk

follow us on

facebook.com/ocrexams
linkedin.com/company/ocr
@OCR_ict
youtube.com/ocrexams



ocr.org.uk/alevelcomputerscience

4742794683