Cambridge Assessment International Education

Curriculum Framework

Cambridge Lower Secondary Computing 0860

Published in September 2021 for first teaching in September 2022.

Version 2.0

Cambridge Pathway

Why choose Cambridge International?

Cambridge International prepares school students for life, helping them develop an informed curiosity and a lasting passion for learning. We are part of the University of Cambridge.

Our Cambridge Pathway gives students a clear path for educational success from age 5 to 19. Schools can shape the curriculum around how they want students to learn – with a wide range of subjects and flexible ways to offer them. It helps students discover new abilities and a wider world, and gives them skills they need for life so they can thrive throughout their schooling, work and life.

Our programmes and qualifications set the global standard for international education. They are created by subject experts, rooted in academic rigour and reflect the latest educational research. They provide a strong platform for students to progress from one stage to the next, and are well supported by teaching and learning resources.

Our mission is to provide educational benefit through provision of international programmes and qualifications for school education and to be the world leader in this field. Together with schools, we develop Cambridge learners who are confident, responsible, reflective, innovative and engaged – equipped for success in the modern world.

Every year, nearly a million Cambridge students from 10 000 schools in 160 countries prepare for their future with the Cambridge Pathway.

Quality management
Cambridge International is committed to providing exceptional quality. In line with this commitment, our quality management system for the provision of international qualifications and education programmes for students aged 5 to 19 is independently certified as meeting the internationally recognised standard, ISO 9001:2015. Learn more at **www.cambridgeinternational.org/ISO9001**

Contents

Changes to this curriculum framework

For information about changes to this curriculum framework, go to page 26.

The latest curriculum framework is version 2.0, published in April 2022.

1 Why choose this curriculum framework?

Key benefits

Cambridge Lower Secondary Computing supports learners to understand why the capability of computer technology is constantly changing and to discover how it is increasingly applied in a range of industries. Learners will explore how computers are programmed to replicate human behaviours, through the use of aritificial intelligence (AI) and machine learning. They will discover that these exciting developments are based on logical algorithms that can be broken down into simpler sub-processes and presented as a series of interconnected parts of a wider process.

Lower secondary learners will present algorithms using flowcharts and pseudocode and will understand how these help them to produce programs, and parts of programs, using text-based programming languages.

Lower secondary aged learners will already understand how to use, and be entertained by, a range of digital devices through their personal explorations and through other areas of their education, including Cambridge Lower Secondary Digital Literacy. The Cambridge Lower Secondary Computing curriculum takes this further by explaining how computers communicate with each other, including how data is broken down and transferred both wirelessly and through cables. Learners will discover how networks are constructed at both a local and global level and how the internet can be considered as being a ‘network of networks’.

Ever since computers were first introduced into classrooms and workplaces, improvements to technology have been continual and dynamic. Therefore, it is important that learners are supported to use, and understand, developments such as augmented reality and other simulators. They will also recognise how large data sets are an important factor in the sophisication of computer systems.

An important feature of the Cambridge Lower Secondary Computing curriculum is computational thinking. The activities that learners undertake while following this curriculum will make sure that they develop transferable thinking skills of logic, decomposition, abstraction, pattern recognition and precision.

Overall, this curriculum helps learners to understand:

- how computers work, including how their systems use logic and selection to process binary data
- how computers are programmed, using professional programming languages
- how computers and software are used to model and simulate data outcomes in a range of scenarios
- how data is transferred, effectively and securely, across local and global networks
- how computers are designed to suit many purposes, and the increasing role they play in local and global industries.

Supporting teachers

We provide a wide range of practical resources, detailed guidance, innovative training and professional development so that you can give your learners the best possible experience of Cambridge Lower Secondary Computing.

You will find most of these resources on the Cambridge Lower Secondary support site (**lowersecondary.cambridgeinternational.org**). Ask the Cambridge coordinator or exams officer in your school if you do not already have a log-in for this support site.

Cambridge Lower Secondary

Teaching resources

- Curriculum framework
- Teacher guide
- Schemes of work

Cambridge Lower Secondary assessments

- Assessment guidance (to support classroom assessment)
- CEM Baseline Tests

Training

- Face-to-face workshops around the world
- Online training
- Cambridge Professional Development Qualifications

Community

You can find useful information, as well as share your ideas and experiences with other teachers, on our social media channels and community forums. Find out more at **www.cambridgeinternational.org/social-media**

Progression through the Cambridge Pathway

Our lower secondary programme is part of the Cambridge Pathway. This pathway leads seamlessly from primary to secondary and pre-university years. Each step of the pathway builds on learners' development from the previous one or from other educational systems. This curriculum framework is typically for learners aged 11 to 14, but it may be appropriate to use it for slightly different ages to suit your context.

You can download more information on progression from the Cambridge Lower Secondary support site.

Teaching time

For guidance, this curriculum framework is based on learners having 2 hours of Computing per week (or about 60 hours per stage). Your actual number of teaching hours may vary according to your context.

2 Curriculum overview

Aims

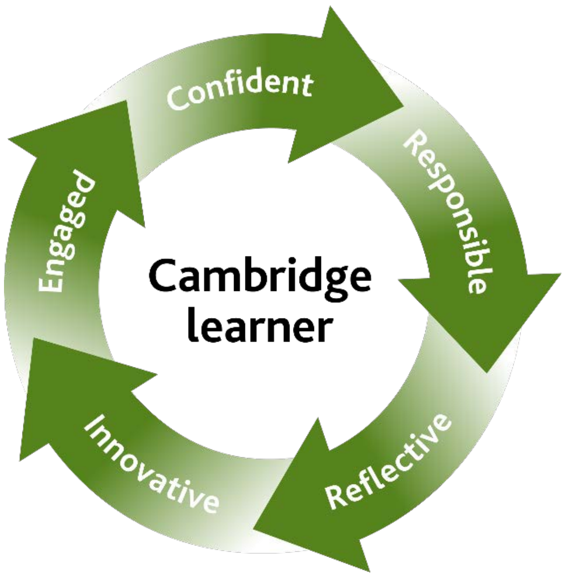
Following the Cambridge Lower Secondary programme helps learners to lay the foundations for lifelong learning, including:

- curiosity about the world around them and enthusiasm for learning
- knowledge, understanding and skills that can be applied in and across subjects
- effective and confident communication skills, including in English
- understanding of their personal and local context, as well as having global awareness.

In Cambridge Lower Secondary Computing, learners:

- become confident computational thinkers, who can abstract key information from a set of instructions, break down problems into smaller parts and recognise patterns within sequences of instruction. They can also present algorithms in formats that are increasingly relevant to the programs that they create.
- think logically, and identify and resolve errors in increasingly complex computing scenarios.
- see themselves as computer scientists, who are able to identify the increasing opportunities that exist for skills such as programming in a range of local and global industries.
- understand the role that data plays in the lives of individuals, businesses and in the wider world. They also understand how computers can be programmed to use large data sets and how these data sets can be used to model scenarios and, therefore, inform decision making.
- develop the vocabulary that is regularly associated with computers and with computational thinking.
- evaluate sequences of instructions and understand the value of working collaboratively so that a range of skills can be applied to the development of computer programs.
- understand how computers and other machines are interconnected and how they play a vital role in a range of industries.

The Cambridge approach encourages learners to be:



Cambridge Lower Secondary Computing supports learners to become:

Responsible – Learners take responsibility for their learning and fully engage with the devices that they use and the programs that they create or adapt. They understand that logical thinking is an important skill, both in the context of being a computer scientist and in their approach to their wider education.

Innovative – Learners recognise the opportunities that exist for those who can think innovatively and creatively within computer science. Lower secondary learners understand that the hardware and software that they will use when they are adults will be far more powerful than what they use today. They understand that the ways that computers communicate with each other is constantly changing. However, they also understand that future products will be created using the same core principles of logic, precision, computational thinking and creativity that are used in today's devices.

Confident – Learners develop the confidence to learn from their mistakes when programming and creating sequences of instruction. When they identify errors, they have the confidence to persevere to find ways to resolve them.

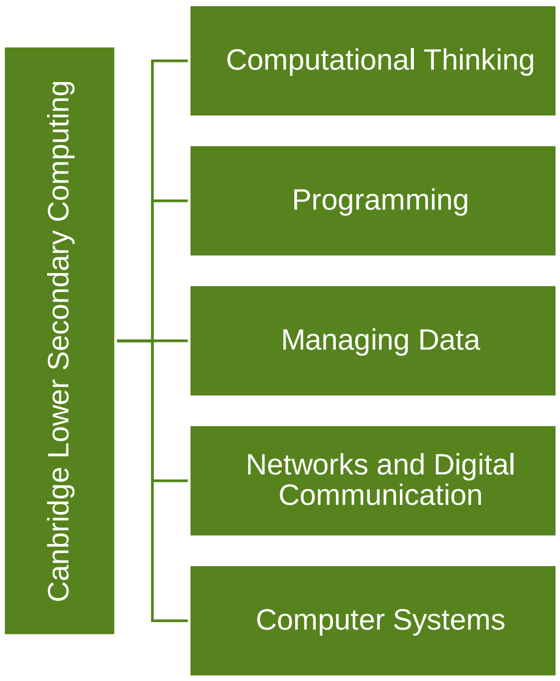
Engaged – Learners are engaged with all aspects of computing. They understand the strong relationship that exists between computers and data, and how data are a key component that helps computers to make effective decisions. Through engagement with a range of data sets, learners understand how computers collect, store, sort and analyse data through a range of platforms including databases, spreadsheets and the World Wide Web.

Reflective – Learners reflect on the role that computers play in society and also in manufacturing and service industries. They recognise how computers and computational thinking can be applied to support them as they continue their education and as they begin to consider their future academic and vocational pathways.

Overview of the strands

This curriculum framework provides a comprehensive set of learning objectives for Cambridge Lower Secondary Computing. These give a structure for teaching and learning and a reference against which learners' attainment and skills development can be checked.

We have divided the learning objectives into five main areas called 'strands' which run through every lower secondary stage. The strands work together to create a holistic approach to the teaching and learning of Computing skills. The strands are listed in the following diagram:


Canbridge Lower Secondary Computing
Computational Thinking
Programming
Managing Data
Networks and Digital Communication
Computer Systems

Below is a brief description of each strand:

Computational Thinking

In this strand, learners begin to present their algorithms professionally. They follow, understand, edit and correct algorithms that are presented as flowcharts and pseudocode and consider the different applications of both. They apply logic and precision, and become increasingly aware of how different computational constructs can be combined within an algorithm.

Learners develop their understanding of sub-routines, including how they can use routines that they import from elsewhere, including from other algorithms and from program libraries.

Programming

During the lower secondary stages, learners use text-based programming languages to create programs that make decisions based upon selection and Boolean logic. They also create programs that use different data types, program libraries and arrays.

Learners consider the different development phases of their programs. They explore the role of test plans and the need to identify a range of data to make sure their programs run as expected. They use systematic thinking skills to identify and correct any programming errors.

As well as on-screen programming, learners create code that enables physical devices to respond to inputs, interact with each other and use data to solve problems.

Managing Data

In this strand, learners consider the role of simulations and data models for a range of purposes. They consider the data requirements of those models and plan the collection of that data. They evaluate the suitability of data models and of the associated data, and explore the role that data plays within specific industries both locally and globally.

Learners use functions within spreadsheets and database software to validate, analyse and select data based upon real world scenarios.

Networks and Digital Communication

This strand complements Cambridge Lower Secondary Digital Literacy, by enabling learners to understand the technical aspects behind the connectivity of computers and related hardware, both locally and globally. Learners discover that the internet is a network of local networks and understand the basic topologies of those local networks. They consider the differences between wired and wireless connections, including their relative benefits and drawbacks, and how new technologies are improving both.

Learners understand how data is transmitted across networks, including how it is broken down into manageable parts to assist the transfer of information. They investigate the tools and protocols that are used to keep data secure while it is in transit.

Learners design networks of their own, and demonstrate which hardware and connectors are needed to make sure communication is secure and scalable.

Computer Systems

This strand considers the role of the different components within a computer system, such as operating systems and application software. Learners apply their learning from familiar experiences to bigger systems, such as automated operations within manufacturing processes.

Learners understand how a computer processes data, to produce onscreen characters and images, and to output sounds. They learn about logic gates and the speed that computers process instructions.

New technologies, such as Artificial Intelligence (AI), are considered in a range of contexts. Learners are supported to understand that systems, and their software, will continue to evolve and that this will continue to change the domestic, educational, industrial and economic landscape around the world.

Overview of teaching approaches

Cambridge Lower Secondary Computing is designed to be taught using a broad range of activities that promote experience, reflection and improvement. Wherever possible, computational thinking should be taught in an integrated way, rather than focusing each lesson on a single element, such as sequencing or debugging. Lessons should include regular opportunities for learners to follow flowcharts and pseudocode to find errors and to predict an output based upon one or more inputs. Through engaging with existing examples of flowchart and pseudocode algorithms, learners will become better equipped to professionally present algorithms of their own.

Computational thinking, programming and data-related content should be taught in context to provide learners with an authentic and meaningful learning experience. Learners should enjoy the programs that they create, and they can do this through activities such as programming quizzes and simple games, or by setting programming challenges for each other.

Many of the programming activities are completed using a text-based programming language, such as Python. In these activities, learners:

- read prepared examples of code, to understand what each line does
- investigate particular sections of code, such as those involving selection, to understand how they are constructed and how they can be copied and adapted for other coding scenarios
- make changes to particular sections of the prepared code to understand the effects of those changes
- understand the importance of labelling sub-routines and variables so that they can be identified within a program and can be re-used elsewhere
- work with others to discuss their observations and to agree corrections and improvements
- understand that collaborative working is an important part of computer science and that it provides opportunities for sharing and extending ideas, and for sharing knowledge.

When exploring new programming contexts, we always recommend that learners investigate, discuss and amend prepared code before they create new code of their own.

Data investigation activities are engaging when they focus on familiar contexts. For example, if learners need to evaluate a data model or simulation for a large set of business data, the products and services used should be of interest to the learners. Fictional data is likely to be most effective here.

Also, when discussing networks and computing systems, learners should consider familiar contexts first. For example, discussing how a film or song moves across a network will be more engaging than discussions or models that focus on an unspecified piece of ‘data’. Lower Secondary learners tend to enjoy the opportunity to understand how computers enable them to access the songs, films, games and other creative outputs that they enjoy.

In contrast, opportunities for learners to see computers working within unfamiliar contexts helps them to understand applications that are beyond their personal use of digital devices for entertainment or educational purposes. For example, site visits or videos of automated manufacturing processes, that show how computing devices and other technology combine to produce outputs in the form of physical products, help learners to understand broader economic benefits of computers.

You can find more information and ideas for teaching and learning activities in the *Cambridge Lower Secondary Computing Teacher Guide* and schemes of work available on the Lower Secondary support site (**lowersecondary.cambridgeinternational.org**).

The teacher guide will support you to plan and deliver lessons using effective teaching and learning approaches.

The scheme of work for each stage of Cambridge Lower Secondary Computing contains:

- suggested units showing how the learning objectives in the curriculum framework can be grouped and ordered
- at least one suggested teaching activity for each learning objective
- a list of subject-specific language that will be useful for your learners
- additional support for aspects which may be unfamiliar to the teacher and the learner, such as example code
- a series of optional end of stage projects that enable learners to apply and consolidate their learning from that stage
- sample lesson plans.

You do not need to use the ideas in the schemes of work to teach Cambridge Lower Secondary Computing. They are designed to indicate the types of activities you might use, and the intended depth and breadth of each learning objective. These activities are not designed to fill all the teaching time for each lower secondary stage. You should use other activities with a similar level of difficulty, for example, those from endorsed resources.

We work with a range of publishers to provide high-quality endorsed resources to support our curriculum frameworks. In order to provide choice for Cambridge International Schools, we encourage publishers to develop resources with varying approaches. There is no requirement for endorsed textbooks to follow the teaching order suggested in the Cambridge Lower Secondary schemes of work. If a resource is endorsed, you can be confident that all the learning objectives are covered.

Teaching tools

The curiculum is designed to be used with a range of teaching tools that are both easy and affordable to access. Many activities do not require the use of any technology so learners can benefit from using traditional tools, such as pen and paper, when following, designing and presenting algorithms. In computing, activities that do not require the use of technology are known as ‘unplugged’.

Pre-prepared sets of cards are useful when teaching Cambridge Lower Secondary Computing. Learners can use these cards to create flowcharts, to create representations of networks or logic circuits, or to catergorise different types of software. Also, as vocabulary is an important part of this curriculum, key vocabulary cards should be displayed which contain computing words. These cards should be used to give learners regular opportunities to explain their understanding of the words.

When learners are creating text-based programs, we recommend that they use Python. Python is one of many text-based programming languages that are used professionally but we recommend it due to its accessibility for learners of this age. An integrated development environment (IDE) is needed for creating, sharing and running Python code but many of these are freely available to download, including IDLE: **www.python.org/downloads** or to use online, for example: **www.online-python.com**.

Learners are also required to use programmable devices such as the micro:bit, **microbit.org**. The micro:bit contains a number of input and output features including lights, buttons and sensors and can be programmed using block-based languages such as Scratch or Microsoft MakeCode.

We recommend that enough computers and other programmable devices are available for learners to work in pairs or small groups.

Learners are likely to use a range of software packages in other subjects, for example for text processing or for creating presentations. The only specific software package that is required for Cambridge Lower Secondary Computing is one that allows learners to input, store and interact with data. We recommend that Microsoft Excel and Microsoft Access are used for teaching the data content, as they contain the functionality that learners will need.

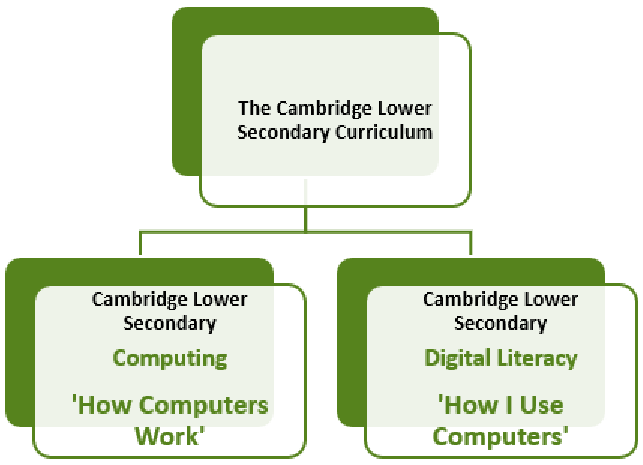
3 Computing and Digital Literacy

Content explanation

Cambridge Lower Secondary Computing supports learners to see inside computer systems. It supports them to understand how computers work and how they communicate with each other and with other technological devices. It also explains how networks are created, from a technical perspective, and supports learners to understand how large scale economic, industrial and service related systems can be created through a combination of hardware, software, data and specific instruction.

This is different to the knowledge and skills related to the creation of digital outputs, such as typed documents, videos and multimedia, that learners will develop in another of our lower secondary subjects, Digital Literacy. Cambridge Lower Secondary Digital Literacy supports learners to develop responsibility for their personal safety and behaviour when online. It enables them to understand the impact that evolving technology has upon both individuals and global communities at a human level. The Digital Literacy curriculum documents are available on the Cambridge Lower Secondary support site.

The relationship between these two Cambridge Lower Secondary curricula can be explained as follows:


The Cambridge Lower Secondary Curriculum
Cambridge Lower Secondary Computing 'How Computers Work'
Cambridge Lower Secondary Digital Literacy 'How I Use Computers'

Lower secondary learners do not need to follow both curricula but there are many benefits if they do. As Cambridge Lower Secondary Digital Literacy is an enabling curriculum that teaches important skills that can be applied across a range of other subjects, it has been designed so that it can be delivered flexibly in order to suit your school timetable.

4 Learning objectives by stage

Overview of learning objectives

There are learning objectives for each of Stages 7 to 9.

To enable effective progression in your teaching, you need to be familiar with the progression of skills across stages. This will help you to build on prior learning in every stage. The progression of learning objectives across Stages 7 to 9 is available on the Lower Secondary support site (**lowersecondary.cambridgeinternational.org**).

Learning objective codes

Each learning objective has a unique code, e.g. **7CT.05**. These codes appear in the schemes of work, teacher guide and other Cambridge Lower Secondary resources. Each learning objective code includes:

- the stage number, e.g. **7**
- a reporting code that reflects the strand titles, e.g. **CT** is **C**omputational **T**hinking
- a number reflecting the order of the learning objectives in the strand for the stage, e.g. **05** is the fifth learning objective.

Stage 7

Computational Thinking

- **7CT.01** Follow, understand, edit and correct algorithms that are presented as flowcharts.
- **7CT.02** Know how to create algorithms using flowchart symbols.
- **7CT.03** Follow and understand the logic of AND, OR, NOT.
- **7CT.04** Understand and use selection statements, limited to IF, THEN, ELSE, presented as flowcharts.
- **7CT.05** Predict the outcome of flowcharts that use selection.
- **7CT.06** Explain the importance of pattern recognition when designing solutions to tasks.
- **7CT.07** Follow, understand, edit and correct algorithms that use sub-routines.
- **7CT.08** Select and use appropriate constructs in algorithms written as flowcharts, limited to sequence and selection.
- **7CT.09** Select and use appropriate comparison operators in algorithms, limited to <,>, <=, >=, == (equal to) and != (not equal to).

Programming

- **7P.01** Identify and describe data types in text-based programs, including Integer, Real and String.
- **7P.02** Know how to develop text-based programs that use input and output.
- **7P.03** Know how to develop text-based programs using data types, including Integer, Real, and String.
- **7P.04** Know how to use variables in text-based programs.
- **7P.05** Know how to develop text-based programs that use different arithmetic operators, including +, −, *, /.
- **7P.06** Evaluate prototypes for software development projects.
- **7P.07** Explain the purpose of project plans for software development projects.
- **7P.08** Know how to apply test plans.
- **7P.09** Understand how errors can be introduced into programs.
- **7P.10** Know how to systematically identify and debug errors in text-based programs.
- **7P.11** Know how to develop programs for a physical computing device to generate multiple outputs, based on multiple inputs.

Managing Data

- **7MD.01** Know that there are many systems that can be used to model real-life scenarios, such as simulators.
- **7MD.02** Evaluate the effectiveness of data capture forms.
- **7MD.03** Know how to write rules to apply conditional formating to cells.
- **7MD.04** Understand the purpose of a primary key.
- **7MD.05** Select appropriate fields to be the primary key.
- **7MD.06** Know how to search pre-existing databases using a single criterion, such as <,>.
- **7MD.07** Know that data is used to model scenarios within a range of industries, including health, manufacture and retail.

Networks and Digital Communication

- **7DC.01** Explain the use of IP addresses and URLs.
- **7DC.02** Explain how DNS enables users to access websites.
- **7DC.03** Know the differences between Bluetooth®, wi-fi and cellular networks, including the different generations of cellular networks (4G, 5G).
- **7DC.04** Outline why errors occur in data transmission.
- **7DC.05** Explain the use of encryption to keep data secure during data transmission.
- **7DC.06** Explain how to check whether a website is secure.

Computer Systems

- **7CS.01** Evaluate the design of digital devices and systems.
- **7CS.02** Understand the differences between application software and system software.
- **7CS.03** Describe how analogue images are digitised.
- **7CS.04** Understand that a binary number can represent different data, such as numbers, characters, images, and sounds.
- **7CS.05** Know that computers are made up of logic gates that are represented by Boolean logic.
- **7CS.06** Understand the role of logic gates in circuits, including AND, OR and NOT.
- **7CS.07** Know that Artificial Intelligence (AI) allows computers to take information from their surroundings to produce outputs based on how they are able to process that information.
- **7CS.08** Explain the use of automation in at least two industries, such as health, manufacture or advertising.
- **7CS.09** Explain a range of applications of AI, including in image recognition and in computer games.

Stage 8

Computational Thinking

- **8CT.01** Follow and understand algorithms that are presented as pseudocode.
- **8CT.02** Follow flowcharts and pseudocode algorithms that use conditional statements.
- **8CT.03** Identify the important characteristics of pseudocode, including that it should be short, clear and precise and should have the start and end clearly shown.
- **8CT.04** Explain the need for searching algorithms.
- **8CT.05** Describe and use linear searches.
- **8CT.06** Understand and use rules using AND, OR and NOT to create logic within algorithms.
- **8CT.07** Predict the outcome of algorithms and test that they meet those outcomes.
- **8CT.08** Know how to decompose problems into their sub-problems.
- **8CT.09** Know how to develop algorithms that use at least one constant.

Programming

- **8P.01** Outline the purpose of program libraries.
- **8P.02** Identify and describe data types in text-based programs, including Integer, Real and Boolean.
- **8P.03** Identify and know how to use library functions in text-based programs.
- **8P.04** Know how to develop text-based programs with conditional (selection) statements.
- **8P.05** Know how to develop text-based programs using data types, including Integer, Real, String and Boolean.
- **8P.06** Know how to develop text-based programs which use rules involving AND, OR and NOT.
- **8P.07** Use an iterative process to develop programs.
- **8P.08** Know how to develop and apply test plans.
- **8P.09** Explain the need for using a range of test data.
- **8P.10** Know how to test algorithms using suitable data.
- **8P.11** Know how to develop programs that allow two or more physical devices to interact.

Managing Data

- **8MD.01** Identify key features of models such as simulators, including their data requirements.
- **8MD.02** Design appropriate forms, including data validation, to collect data for given purposes.
- **8MD.03** Know how to use spreadsheets that are models of real-life systems, using what-if analysis to compare alternative scenarios.
- **8MD.04** Evaluate the suitability of data that have been collected for particular purposes.
- **8MD.05** Know how to add validation rules to a database structure.
- **8MD.06** Evaluate the suitability of pre-existing databases for given purposes.
- **8MD.07** Identify the data that are required for specific applications.

Networks and Digital Communication

- **8DC.01** Identify types of network, including PAN, LAN, WAN.
- **8DC.02** Describe the uses and characteristics of copper cables and fibre optic cables to transmit data.
- **8DC.03** Describe the advantages and disadvantages of wired and wireless networks, including performance and security aspects.
- **8DC.04** Describe how echo checks are used to detect errors in transmission.
- **8DC.05** Explain the role and the importance of firewalls in networks.
- **8DC.06** Explain the use of antivirus and antispyware to keep data secure on a network.

Computer Systems

- **8CS.01** Describe the purpose of operation systems.
- **8CS.02** Describe the purpose of utility programs.
- **8CS.03** Describe how ASCII is used to represent characters.
- **8CS.04** Define the term ‘compression’ and describe why it is required.
- **8CS.05** Know how to convert binary to denary, and denary to binary.
- **8CS.06** Complete truth tables for AND, OR and NOT gates.
- **8CS.07** Understand the roles of primary memory, RAM and ROM.
- **8CS.08** Define the term ‘machine learning’.
- **8CS.09** Identify the use of augmented reality in familiar contexts, including education and entertainment.
- **8CS.10** Describe how autonomous programming and AI is used in robotics.

Stage 9

Computational Thinking

- **9CT.01** Follow, understand, edit and correct algorithms that are presented as pseudocode.
- **9CT.02** Follow flowchart or pseudocode algorithms that use loops.
- **9CT.03** Know how to create algorithms using flowcharts and pseudocode.
- **9CT.04** Know how to use predefined sub-routines in flowcharts or pseudocode.
- **9CT.05** Describe and use binary searches.
- **9CT.06** Understand and use iteration statements, limited to count-controlled loops, presented as either flowcharts or pseudocode.
- **9CT.07** Predict the outcome of algorithms that use iteration.
- **9CT.08** Compare and contrast algorithms designed for the same tasks to determine which is best suited to the purpose.
- **9CT.09** Combine multiple constructs (sequence, selection, count-controlled iteration) to write algorithms as flowcharts or pseudocode.

Programming

- **9P.01** Explain the purpose of a one-dimensional array.
- **9P.02** Identify and describe data types in text-based programs, including Integer, Real, Character, String and Boolean.
- **9P.03** Know how to develop text-based programs with count-controlled loops.
- **9P.04** Know how to access data from an array using a text-based language.
- **9P.05** Know how to develop text-based programs using string manipulation, including length, upper case, and lower case.
- **9P.06** Use iterative development on software prototypes to produce solutions to problems.
- **9P.07** Evaluate the processes that are followed to develop programs.
- **9P.08** Know how to develop and apply test plans that include normal, extreme and invalid data.
- **9P.09** Identify test data that covers normal, extreme and invalid.
- **9P.10** Identify a range of errors, including syntax, logic, and runtime errors.
- **9P.11** Use trace tables to systematically debug text-based programs.
- **9P.12** Know how to program physical devices to use data to solve problems.

Managing Data

- **9MD.01** Evaluate the use of models that represent real-life systems.
- **9MD.02** Know how to use functions in spreadsheets to analyse data, including IF, MIN, MAX, COUNT.
- **9MD.03** Create spreadsheets that model real-life systems.
- **9MD.04** Evaluate the suitability of pre-existing spreadsheets for given purposes.
- **9MD.05** Know how to create relational databases with two or more linked tables.
- **9MD.06** Know how to create complex searches for data in databases using two or more criteria.
- **9MD.07** Create complex searches in relational databases.
- **9MD.08** Define the term ‘Big Data’ and describe its applications.

Networks and Digital Communication

- **9DC.01** Know that there are different network topologies, including bus, ring and star.
- **9DC.02** Explain the role of protocols in transmitting data, including TCP/IP and HTTP.
- **9DC.03** Explain the scalability factors that should be considered when designing networks.
- **9DC.04** Understand the role of parity bits in error detection.
- **9DC.05** Explain the choices that should be made when implementing network security, including accessibility, cost and the relative security requirements of different data sets.

Computer Systems

- **9CS.01** Identify improvements to the design of digital devices, based on prototypes and a range of factors including user experience, accessibility, ergonomics and emerging technologies.
- **9CS.02** Understand which tasks are carried out by an operating system.
- **9CS.03** Describe examples of utility programs including drivers, security software and defragmentation.
- **9CS.04** Understand that there are different types of translator, including the main characteristics of compilers and interpreters.
- **9CS.05** Describe how analogue sound is digitised.
- **9CS.06** Know how to convert between storage units.
- **9CS.07** Know how to draw logic circuits for Boolean expressions.
- **9CS.08** Understand that computers store lists of instructions to be run one at a time.
- **9CS.09** Understand the Fetch-Decode-Execute cycle.
- **9CS.10** Describe a range of scenarios where machine learning is used.
- **9CS.11** Describe the benefits and risks of the computerisation of traditional manufacturing and industrial practices, for example Industry 4.0.

5 Glossary

This glossary is provided to support the content of this curriculum framework. The definitions are intended to be sufficient to guide an informed reader.

Algorithm – a set of precise instructions for solving a problem.

Application software – programs that are written for a specific task or set of tasks, such as text processing software, media players or games.

Array – a series of memory locations that each hold a single item of related data.

ASCII – code that represents letters, numbers and punctuation within a computer system. ASCII is typically converted into 8-bit code, therefore allowing 256 items to be represented.

Automation – the application of machines that are programmed to carry out repeated industrial procedures.

Autonomous programming – programming that allows devices to function independently, depending on the inputs received from their sensors.

Big Data – data that contains variety and are available in ever increasing volumes. Specific data are extracted from big data based on the requirement.

Bluetooth® – a wireless technology used for exchanging data over short distances.

Cellular network – a network that allows mobile phones to connect. The last link is wireless but allows devices to connect to a transmitter within the local area, or local cell.

Conditional format – a feature in many data applications that allows rules to be placed on cells so that they meet certain criteria.

Conditional statement – the instruction in a program that instructs it to perform different actions depending on whether the condition is ‘true’ or ‘false’. Also known as a Selection statement.

Constant – a data value that stays the same each time a program is run, for example the starting values within a game.

Construct – the common building blocks that are used when designing programs.

Data type – the content of data within a variable, for example whether it is text or numeric.

Decomposition – the taking of a complex problem and breaking it down into smaller parts that are easier to solve.

Defragmentation – the process that organises the content of mass storage devices into the smallest number of separate parts.

DNS – the domain name system, or address book, of the internet. It allows web browsers to connect with websites.

Emerging technology – refers to new technology and to the continued development of existing technology.

Flowchart – a diagram that uses a standard set of symbols, including arrows, to represent each step of an algorithm. The symbols are used to represent different types of instruction.

Industry 4.0 – the combination of physical production and smart technology, machine learning and Big Data to automate production processes. Also known as the fourth industrial revolution.

Integer – the data type for a whole number within a variable in a program.

IP address – a unique address that identifies a device on the internet or on a local network.

Iterative process – the process of writing part of a program, testing it, then editing or adding to the program continually until a final, working, version is produced.

Learning objectives – statements from the curriculum framework of the expectations of knowledge, understanding and skills that learners will develop; they provide a structure for teaching and learning, and a reference against which to check learners' attainment and skills development.

Logic gates – the building blocks of digital circuits that have one or two inputs that can be turned on or off.

Loop – a section of code that is repeated within a program.

Machine learning - an application of AI that enables systems to automatically learn, and alter their behaviours, based on their experiences.

Model (data) – a computer application that replicates a real-life environment, where data is changed to see what happens. Simple models can be built in spreadsheet software.

Model (teaching) – the demonstration of an activity for learners to follow. This could be walking through an algorithm or program to find and debug an error, where the teacher describes each step that they take and verbally asks themselves the questions that will inform their next steps.

Operation, or operating, system – software that supports a computer's basic functions.

Parity bit – a check that is added to a block of data for error detection purposes.

Physical (computing) device – devices that learners program and interact with through a range of inputs, including sensors, and outputs, including LEDs.

Primary key – a key in a relational database that is unique for each data record and allows the data for each record to be identified across different data sheets.

Prototype – an early example of a product, including a device or a piece of software. Prototypes allow for early feedback and help to define the requirements of the final product.

Program library – a collection of programs that are made available for importing into other programs to perform a particular function or sub-routine.

Protocol – established rules that determine how data is transmitted across a network.

Pseudocode – a way of planning the structure and function of a program in a short, clear and precise way. Pseudocode is not specific to any programming language.

Real – a programming data type that stores numbers that contain decimal places.

Relational database – data that is organised into tables but allows data from one table to be identified based upon its relationship to data within another table.

Scheme of work – support materials for each stage of Cambridge Lower Secondary Computing. Each scheme of work contains a suggested long-term plan, a medium-term plan with suggested teaching and learning activities and sample short-term (lesson) plans.

Searching algorithm – an algorithm used for finding a particular item within a large dataset.

Selection – a programming construct where a decision is made within a program, such as where the program decides to move in a particular direction based upon the outcome of a prior event or upon an input from a user.

Selection statement – the instruction in a program that tells it to perform different actions depending on whether the condition is 'true' or 'false'. Also known as a Conditional statement.

Simulator – a model that takes a large amount of data to either predict a future instance, determine the best course of action or validate a model.

Strand – a collection of learning objectives in the curriculum framework that forms an area of learning.

String – a programming data type that stores alphanumeric combinations and text. A string should not include numbers that are used for calculations but can include sequences of numbers in contexts such as addresses or telephone numbers.

Sub-routine – a discrete section of code that performs a particular operation within a program. Sub-routines can be used many times within a program and can be re-used in other programs.

Syntax – the rules that define the structure of a programming language.

System software – software that controls the hardware within a computer system and provides the user interface.

Teacher guide – a document providing support in using the curriculum framework to plan and deliver lessons using effective teaching and learning approaches.

Topology – the arrangement of the components within a network.

Unplugged – computational thinking activities that take place without digital devices, such as following an algorithm through physical movement.

Utility program – a program that works alongside the operating system to support a computer's infrastructure.

URL (Uniform Resource Locator) – the address of a given unique resource on the Web.

Variable – a memory location within a program that stores particular values, such as the name, age or score of a user. The values can change each time the program is run or while the program is running.

Vocabulary – words and phrases.

Wi-fi – network technology that allows devices to connect to the internet without wires.

Wired / Wireless – describes whether devices are connected to a network with or without cables.

World Wide Web – a service provided by computers that are connected to the internet, consisting of pages of information that are transmitted to users upon request.

6 Changes to this curriculum framework

This curriculum framework has been amended. The latest curriculum framework is version 2.0, published in April 2022.

- Learning objective **7CT.09** has been changed to: Select and use appropriate comparison operators in algorithms, limited to <,>, <=, >=, == (equal to) and != (not equal to).

Cambridge Assessment International Education
The Triangle Building, Shaftesbury Road,
Cambridge, CB2 8EA, United Kingdom

t: +44 (0)1223 553 554

www.cambridgeinternational.org