Scheme of Work

Cambridge O Level
Computer Science

2210
For examination from 2016

Contents

Introduction

Developed from Cambridge O Level Computer Studies (7010) and now renamed Computer Science, this syllabus has been reviewed throughout to bring it up to date and to allow learners to begin the development of their computational thinking and programming skills. As 'Computer Science', this syllabus now shares the same name as the Cambridge IGCSE and the AS and A Level (formerly AS and A Level Computing) syllabuses, indicating the firm links and progression between these syllabuses.

This scheme of work provides ideas about how to construct and deliver the Cambridge O Level Computer Science course. The syllabus for Cambridge O Level Computer Science (2210) has been broken down and the scheme of work contains ten units with suggested teaching activities and learning resources to use in the classroom.

Outline
Whole class **(W)**, group work **(G),** pairwork **(P)** and individual activities **(I)** are indicated throughout this scheme of work. The activities in the scheme of work are only suggestions and there are many other useful activities to be found in the materials referred to in the learning resource list.

There is the potential for differentiation by resource, length, grouping, expected level of outcome, and degree of support by teacher, throughout the scheme of work. Timings for activities and feedback are left to the judgment of the teacher, according to the level of the learners and size of the class. Length of time allocated to a task is another possible area for differentiation.

The units within the scheme of work (with suggestions for time allocations, based on a total allocation of about 130 hours) are:

Unit 1:   **Introduction to computer systems** (15 hours)
Unit 2:   **Numbers, processors and operating systems** (10 hours)
Unit 3:   **Data communications and networking** (12–15 hours)
Unit 4:   **Data integrity and security** (10 hours)
Unit 5:   **Binary logic** (15 hours)
Unit 6:   **Practical problem solving – structure diagrams, algorithms and flowcharts** (12 hours)
Unit 7:   **Practical problem solving – pseudocode** (12 hours)
Unit 8:   **Programming concepts** (12 hours)
Unit 9:   **Databases** (9 hours)
Unit 10:   **Use of pre-release materials** (20–25 hours)

Teaching order
Units 1 to 5 address elements that will be tested in Paper 1. It is recommended that Unit 1 is taught first. Units 6 to 10 address elements that will be tested in Paper 2. These units are best taught in sequence, as concepts developed in one unit will be applied in the following units. Schools may choose to teach units addressing Paper 1 and Paper 2 in parallel, to balance theory with practical activity.

It is suggested that the course starts with an induction session, giving learners an overview of the contents of the course, the types of activities that they can expect to experience and the expectations expressed in the syllabus for learners of differing capabilities. Resources that will prove useful in support of this include the syllabus, the scheme of work and specimen papers.

In addition, it may be useful to give learners a tutorial on effective research using the internet. This could include the ability to focus a search and to identify the reliability and safety of sites. A useful resource to do this includes BBC Bite Size (www.bbc.co.uk/schools/gcsebitesize/dida/using_ict/webresearchrev5.shtml).

The time allocation is an approximate guide only, but will help to give some indication of the time that will be needed to be allocated to each of the units.

Approaches to teaching and learning

There is a great deal of evidence to show that the most effective teaching and learning takes place in structured lessons with a variety of different styles of active teaching and learning. A number of these are suggested within the units; a lesson structure comprising a short active starter (which may reflect on what has already been learned, e.g. a quiz or other oral activity), an explanation of what will be learned in the coming lesson with expectations, the core activity (with practical learner-centred activities where possible) and summarising the lesson with a plenary to review what has been learned.

Schools will have their own system for learners' note-keeping (for use in revision). It is important that the outcome of class-based and group-based activities can be recorded within this system.

Please note that learners do not gain marks for specific knowledge of brand names of software packages or hardware, for example, named viruses or anti-virus software such as Norton Internet Security or McAfee. The rubric on each question paper makes this clear.

Teacher support

Cambridge Teacher Support is a secure online resource bank and community forum for Cambridge teachers. Go to http://teachers.cie.org.uk for access to specimen and past question papers, mark schemes and other resources. We also offer online and face-to-face training; details of forthcoming training opportunities are posted online.

An editable version of this scheme of work is available on Teacher Support. Go to http://teachers.cie.org.uk. The scheme of work is in Word doc format and will open in most word processors in most operating systems. If your word processor or operating system cannot open it, you can download Open Office for free at www.openoffice.org

Resource list

An up-to-date resource list for the Cambridge O Level Computer Science (syllabus 2210) can be found at www.cie.org.uk

Textbooks

Leadbetter, C, Wainwright, S and Stinchcombe, A *Cambridge IGCSE Computer Studies Coursebook* (Cambridge University Press, UK) ISBN: 9780521169042
Watson, D and Williams, H *Cambridge IGCSE Computer Studies Revision Book* (Cambridge University Press, UK) ISBN: 9781107674196

Websites

The particular pages in the learning resources column for the units have been explored, but not other aspects of these sites so only the particular resources are recommended. There may be other useful materials on these websites but they have not been checked. However, the following websites have a range of useful materials beyond those identified in the learning resources column.

'BBC Bite Size' is a revision site containing notes, activities and tests across a range of contexts appropriate to this syllabus. Although it is titled ICT, much of the information refers to elements within this Computer Science syllabus.
www.bbc.co.uk/schools/gcsebitesize/ict/

Although titled 'Teach ICT', there is a comprehensive website for GCSE Computing subjects available. This includes notes, quizzes and lesson ideas. Much is free, although a small subscription gives access to additional useful resources such as a wide range of ideas for starter and plenary activities.
www.teach-ict.com/gcse_computing.html

'Computer Science Unplugged' is a collection of free learning activities that teach Computer Science through engaging games and puzzles.
http://csunplugged.org/

'How Stuff Works' is a wide-ranging website containing a wealth of information about computer systems.
http://computer.howstuffworks.com/

'Computer Science for Fun' is produced by staff in the School of Electronic Engineering and Computer Science of Queen Mary, University of London with the aim of 'sharing our passion about all things to do with Computer Science'. It is wide-ranging and interesting to read, with activities and magazine-type articles.
www.cs4fn.org/

'Computing at School (CAS)' is 'a grass roots organisation that aims to promote the teaching of computing at school. CAS is a collaborative partner with the British Computer Society (BCS) through the BCS Academy of Computing, and has formal support from other industry partners'. It is possible to join the CAS website (free of charge) and share ideas for teaching and learning.
www.computingatschool.org.uk/

'Computer Science Teachers Association' is an American institution that promotes the teaching of computer science. It is free to join.
www.csta.acm.org/

'Quizlet' contains lots of short quizzes on different aspects of computing, contributed by teachers.
http://quizlet.com/subject/computing/

A website containing a wide range of notes, presentations, quizzes, etc. for GCSE and other certifications; Teachers and learners may register (free) with a school email address.
www.mrfraser.org/resources/

A website designed to support the Cambridge IGCSE ICT syllabus (0417), but contains some useful information for the 2210 syllabus too.
www.igcseict.info/

Royal Institution Christmas Lectures 2008 – Hi-Tech Trek (5 lecture/demonstrations, each lasting about 40 minutes, covering computer technologies). It is recommended that teachers watch these and identify extracts (or possibly whole videos) that may be of interest to their learners at various points through the course.
www.richannel.org/christmas-lectures/2008/2008-chris-bishop

Associated activities can be found at:
www.rigb.org/christmaslectures08/

A source of definitions of computing terms, with links to associated concepts.
www.webopedia.com/

A website that will help with learning various programming languages.
www.codecademy.com

Unit 1: Introduction to computer systems

Recommended prior knowledge
Learners beginning this course are not expected to have studied computer science previously. Often, different learners will have varying levels of knowledge of computer science concepts; therefore this unit starts with an introduction to the basic concepts.

Context
The basic parts of a computer and the functions of a computer system are introduced here, followed by an appreciation of the diversity of the range of computer systems. This leads into a more detailed consideration of the major components of a computer system – input devices, memory and output devices. This unit provides underpinning knowledge for later units (e.g. Units 2 and 3).

Outline
This unit begins with a consideration of the basic parts of a computer system and what a computer system does. A more detailed exploration of input and output devices follows, together with commonly encountered types of computer memory.

It is suggested that much of this work is carried out by active learning, facilitated by the teacher. Paired work is suggested, as evidence shows that the discussion generated whilst working in pairs helps to develop and embed learning; oral presentation and the opportunity to answer questions from peers also helps in this process. Activities such as quizzes (both authoring and answering), brainstorming and short-answer questions can be used to break up the lesson into smaller sub-sections.

In order for learners to have a set of notes for revision purposes, a common structure for writing up notes would be useful. The teacher may decide to produce a template which will focus the notes on key points and make them more consistent between groups of learners. For purposes of accessibility in the future, a printed format or an intranet page may be most convenient. If a paper-based approach is used, copies of reports/posters/leaflets can be distributed to each learner after the presentations and, taken together, the reports from each group will provide a class set of notes to cover the topic under discussion.

Suggested teaching time
Based on a total time allocation of 130 contact hours for this Cambridge O Level Computer Science course, it is recommended that this unit should take about 15 hours.

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
1.3.3	Describe the principles of operation (how each device works) of these input devices: 2D and 3D scanners, barcode readers, Quick Response (QR) code readers, digital cameras, keyboards, mice, touch screens, interactive whiteboard, microphones Describe how these principles are applied to real-life scenarios, for example: scanning of passports at airports, barcode readers at supermarket checkouts, and touch screens on mobile devices	Start with a class discussion about what learners already know. What computer systems do they know about and where are they used? Write ideas on the whiteboard and expand with more examples. **(W)** Show a video of computer systems from the early days to modern-day devices so that learners can see the difference e.g. in size, application of computer science and portability (History of computers video lasts about five minutes). **(W)** Ask learners to work in pairs to identify as many parts of a computer system that they can **(P)** Write up on board and then add to that list. **(W)** Develop this list into categories by brainstorming: • input devices (keyboard, mouse, touch screens, scanners, etc.) • processing devices (from large and power-hungry in supercomputers to small, low power consumption in smart phones and microcontrollers) • storage devices (internal memory (RAM), backing storage such as HDD and DVD, etc.) • output devices (printers, screens, plotters, etc.). **(W)** Use a quiz or match the definitions activity to match the basic parts and functions of a computer system. **(W)/(G)** Discuss the need to communicate with a computer and physical ways of doing this. Ask learners to identify common features of input, processing, data storage and output by completing a gapped handout. **(G)/(P)**	History of computing website: www.computerhistory.org/timeline/?category=cmptr History of computers video: www.history.com/shows/modern-marvels/videos/who-invented-the-computer#who-invented-the-computer Outline of computer systems and their components at: www.teach-ict.com/gcse_computing/ocr/211_hardware_software/computer_system/home_computer_system.htm BBC Bite size contains notes, activities, tests, etc.: www.bbc.co.uk/schools/gcsebitesize/ict/system/0ictsystemsrev1.shtml Teacher provides quiz or card sets, each set comprising the range of devices listed in the syllabus and their associated functions. Input devices, processing and output devices: www.bbc.co.uk/schools/gcsebitesize/ict/hardware/0inputandoutputdevices_act.shtml Websites such as http://computer.howstuffworks.com/ *Cambridge IGCSE Computer Studies Coursebook* pp. 49–66 *Cambridge IGCSE Computer Studies Revision Book* Chp 12.3

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
		Specialist devices such as sensors and actuators used in monitoring and control systems and those designed for disabled people, devices for interfacing with virtual reality systems, etc. this needs to be explained by the teacher. **(W)** Learners are divided into small groups; each group investigates one specific input device and produces a report/poster/leaflet on: • how it works • typical applications • why it is used in these applications; its advantages and limitations. Reports/posters/leaflets to be prepared using software such as DTP, presentation software, intranet/internet pages etc. Each group gives a five-minute presentation on their device. **(G)** Each group prints enough copies of their reports to provide a copy for each learner. If learner work is stored on an intranet, copy files into each learner account.	
1.3.3	Describe how a range of sensors can be used to input data into a computer system, including light, temperature, magnetic field, gas, pressure, moisture, humidity, pH and motion Describe how these sensors are used in real-life scenarios, for example: street lights, security devices, pollution control, games, and household and industrial	Teacher introduction explaining the distinction between: • monitoring, in which system acquires data at intervals from sensors and, where necessary, analogue-to-digital converters (ADCs), and how software processes the input data to provide the user with information for monitoring physical or chemical quantities (such as temperature, flow rate or oxygen concentration), and warning signals if stored limits are exceeded • control, in which input data may also be used as	*Cambridge IGCSE Computer Studies Coursebook* pp. 66–71 *Cambridge IGCSE Computer Studies Revision Book* Chp 12.3 Illustrated notes on sensors: www.igcseict.info/theory/2/sensor/ Notes on sensors: www.bbc.co.uk/schools/gcsebitesize/ict/measurecontrol/0computercontrolrev2.shtml

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
	applications	feedback from a system being controlled so that software can compare feedback with stored set-points or upper and lower limits to decisions about the outputs required to, where necessary, digital-to-analogue converters (DACs) and actuators, such as heaters or motorised valves. **(W)** Learners work in groups; each group investigates one specific sensor and produces a report/poster/leaflet on: • how it works • typical applications • why it is used in these applications; its advantages and limitations. Reports/posters/leaflets to be prepared using software such as DTP, presentation software, intranet/internet pages etc. Each group gives a five-minute presentation on their device. **(G)** Each group prints enough copies of their reports to provide a copy for each learner. If learner work is stored on an intranet, copy files into each learner account.	
1.3.4	Describe the principles of operation of the following output devices: inkjet, laser and 3D printers; 2D and 3D cutters; speakers and headphones; actuators; flat-panel display screens, including Liquid Crystal Display (LCD) and Light-Emitting Diodes (LED) display; LCD projectors and Digital Light Projectors (DLP)	Learners work in groups to investigate one specific output device and produce a report/poster/leaflet on: • how it works • typical applications • why it is used in these applications; its advantages and limitations. Reports/posters/leaflets to be prepared using software such as DTP, presentation software, intranet/internet pages etc. Each group gives a five-minute presentation on their device. **(G)**	Websites such as http://computer.howstuffworks.com/ *Cambridge IGCSE Computer Studies Coursebook* pp. 71-9 *Cambridge IGCSE Computer Studies Revision Book* Chp 12.3

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
	Describe how these principles are applied to real-life scenarios, for example: printing single items on demand or in large volumes; use of small screens on mobile devices	Each group prints enough copies of their reports to provide a copy for each learner. If learner work is stored on an intranet, copy files into each learner account.	
1.3.5	Show understanding of the difference between: primary, secondary and off-line storage and provide examples of each, such as: primary: Read Only Memory (ROM), and Random Access Memory (RAM) secondary: hard disk drive (HDD) and Solid State Drive (SSD); off-line: Digital Versatile Disc (DVD), Compact Disc (CD), Blu-ray disc, USB flash memory and removable HDD Describe the principles of operation of a range of types of storage device and media including magnetic, optical and solid state Describe how these principles are applied to currently available storage solutions, such as SSDs, HDDs, USB flash memory, DVDs, CDs and Blu-ray discs Calculate the storage requirement of a file	Class brainstorms the difference between primary (e.g. RAM, ROM), secondary (e.g. hard disks, SSD) and offline (e.g. removable storage media such as CD, USB flash memory) memory, storage devices and media. **(W)** Learners complete a gapped handout, requiring them to insert the correct class of memory device to match a definition or application. **(I)** Teacher gives a short presentation to explain and clarify the principles and differences between magnetic, optical and solid state memory. **(W)** Learners work in groups to describe the ways in which different types of memory are used in a typical day in their life (e.g. DVD/ Blu-ray to watch a film, USB flash memory to carry data to and from school, CD to play music, use of mobile, MP3 player etc.). Use an appropriate way to keep notes on their work. **(G)**	Data storage: www.bbc.co.uk/schools/gcsebitesize/ict/hardware/1datastoragerev2.shtml Websites such as: http://computer.howstuffworks.com/computer-ram-memory-channel.htm *Cambridge IGCSE Computer Studies Coursebook* pp. 79–88 *Cambridge IGCSE Computer Studies Revision Book* Chp 12.4
1.3.3 1.3.4	Formative assessment activities	Learner progress could be assessed using specimen/past paper examination questions/multiple-	Example questions given in specimen papers for 2210, past O Level papers for the previous syllabus

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
1.3.5		choice test/short case study and questions.	7010 are available at: http://teachers.cie.org.uk Online tests within BBC Bite Size (details earlier) Tests/quizzes at: www.teach-ict.com/gcse_computing/gcse_computing_quizzes.htm

Unit 2: Numbers, processors and operating systems

Recommended prior knowledge
In order to understand the role of an operating system, learners should have had practical experience of using at least one operating system with a Graphical User Interface (GUI). It is also recommended that learners should have studied Unit 1 before starting this unit.

Context
This unit looks at the way in which numbers are represented within a computer system, the structure of the central processing unit and its functions, and the role of the operating system in managing the components of a computer system and interactions with the user.

Outline
This unit starts with binary and hexadecimal representation of numbers, leading to the von Neumann model of a computer system and the concept of a computer. This is illustrated practically by learner use of the Little Man Computer (LMC). The role of operating systems is then considered, including control of peripherals and the user interface. Learners will not be expected to know detail of any specific operating system.

Suggested teaching time
Based on a total time allocation of 130 contact hours for this Cambridge O Level Computer Science course, it is recommended that this unit should take about 10 hours.

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
1.1.1	Convert positive denary integers into binary and positive binary integers into denary	Teacher presentation to introduce the concepts of binary notation, e.g. using an automatic binary counter; binary number cards available (with worksheets etc.). **(W)** Learners convert denary numbers into binary and binary numbers into denary. **(I)** Reinforce with a game such as the Cisco binary game. This provides formative assessment of understanding. **(G)** Learners answer previous exam/textbook questions on binary representation. **(I)**	Binary counter – for example: www.mathsisfun.com/binary-decimal-hexadecimal-converter.html Binary numbers at Computer Studies Unplugged: http://csunplugged.org/binary-numbers Cisco binary game: http://forums.cisco.com/CertCom/game/binary_game_page.htm *Cambridge IGCSE Computer Studies Revision Book* Chp 11.4 – sample questions in Chp 11.6
1.1.1	Recognise the use of binary numbers in computer systems	Teacher presents the concept of the byte; class discussion about how the byte is used to measure	Simple comparisons at www.bbc.co.uk/schools/gcsebitesize/ict/hardware/1

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
	Show understanding of the concept of a byte and how the byte is used to measure memory size	memory size by introducing the concept of kb, Mb, Gb, Tb. **(W)** Class brainstorm to reflect on capacity of commonly found elements of computer systems such as hard disk drives, RAM, DVD, USB flash drives etc. (refer back to Unit 1 – types of memory). **(G)**	datastoragerev2.shtml Useful reinforcement material: http://computer.howstuffworks.com/bytes.htm
1.1.2	Represent integers as hexadecimal numbers Show understanding of the reasons for choosing hexadecimal to represent numbers Convert positive hexadecimal integers to and from denary Convert positive hexadecimal integers to and from binary Represent numbers stored in registers and main memory as hexadecimal	Teacher presentation on hexadecimal notation and its relationship to binary notation. Demonstration of the conversion of binary and denary to hexadecimal. **(W)** Learners convert positive hexadecimal integers to and from binary and to and from denary. **(I)** Class brainstorm to show understanding of the reasons for choosing hexadecimal to represent numbers, e.g. those stored in registers and main memory. **(W)** Learners answer previous exam/textbook questions on hexadecimal representation. **(I)**	Hexadecimal counter – for example: www.mathsisfun.com/binary-decimal-hexadecimal-converter.html *Cambridge IGCSE Computer Studies Revision Book* Chp 11.5 – sample questions in Chp 11.6
1.3.2 1.1.1	Show understanding of the basic Von Neumann model for a computer system and the stored program concept (program instructions and data are stored in main memory and instructions are fetched and executed one after another) Describe the stages of the fetch-execute cycle, including the use	Teacher presents basic concepts of computer architecture, including registers, and the fetch-execute cycle followed by demonstration via projector of the LMC. **(W)** Learners carry out simple low level tasks using LMC software – paired work is probably most effective. **(P)** Differentiation can be achieved by giving able learners more challenging tasks (examples available in quoted	Notes/presentation on computer architecture: http://web.eecs.utk.edu/research/cs100modules/module1/index.html Notes and animations of fetch-execute cycle: www.eastaughs.fsnet.co.uk/cpu/execution-cycle.htm Little Man Computer download: http://gcsecomputing.org.uk/lmc/lmc.html

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
	of registers Use binary in computer registers for a given application (such as in robotics, digital instruments and counting systems)	resources and by searching for the LMC tasks using e.g. Google).	Little Man Computer web based: http://www.yorku.ca/sychen/research/LMC/
1.3.6	Describe the purpose of an operating system Show understanding of the need for interrupts	Teacher presentation to include: the idea of system software as different from applications software general tasks and facilities of an operating system – for processor management, it is helpful to demonstrate Windows Task Manager the role of the operating system (OS) in file management how peripheral devices, such as keyboards and printers, must be controlled and responded to by the operating system how communication between the computer and peripherals must be controlled and errors detected. **(W)** Learners (paired/grouped) to research: • buffer • polling • interrupts • handshaking • checksum. Learners use their findings to create a short role play activity that demonstrates how each of these works **(G/P)**. Learners need to make their own notes on each of these after they have been acted out. **(I)** Class brainstorm to review learners' previous experience of operating systems with graphical user	*Cambridge IGCSE Computer Studies Coursebook* pp. 92–4 *Cambridge IGCSE Computer Studies Revision Book* Chp 13 Introduction to operating systems: http://gcsecomputing.net/wp-content/uploads/2012/01/OCR%20A451%202.1.2%20CPU%20-%20Summary.pdf Several pages describing operating systems and their functions: www.howstuffworks.com/operating-system1.htm Windows, Linux, Android could be used as examples *Cambridge IGCSE Computer Studies Coursebook* pp. 102–5 *Cambridge IGCSE Computer Studies Coursebook* pp. 98–100 Theory notes, activities and quizzes on user interfaces: www.teachict.com/gcse_computing/ocr/213_software/user_interface/home_user_interface.htm Notes on user interfaces: www.igcseict.info/theory/1/uis/index.html

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
		interfaces (GUI), and introduce the idea of a command line interface. **(W)**	Quizzes to test understanding at: www.teach-ict.com/gcse_computing/gcse_computing_quizzes.htm
		Discuss the main differences between command line interfaces and GUIs and their respective advantages and disadvantages. **(G/P)**	
		Pairs of learners devise their own quiz questions (and answers) on this unit **(P).**	
		Teacher selects one or two quizzes to test understanding of operating systems and their function. **(W)**	

Unit 3: Data communications and networking

Recommended prior knowledge
Learners should have had some experience of using the internet and of using a local area network (LAN) in school.

Context
Data transmission between a processor and its peripherals, and between computers in a network, is a central element in everyday life; the internet has become an unconscious way of life for many people today. This unit looks at the principles underpinning data transmission in these contexts.

Outline
Learners find out about data transmission and the differences between serial and parallel data transmission. The concepts, advantages and problems associated with computer networks are introduced. Learners design a simple web page using HTML, and consider ways in which the internet can be misused for criminal purposes. Strategies for addressing these issues are then considered, including a practical approach to cryptography.

Suggested teaching time
Based on a total time allocation of 130 contact hours for this Cambridge O Level Computer Science course, it is recommended that this unit should take about 12–15 hours.

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
1.2.1	Show an understanding of what is meant by transmission of data Distinguish between serial and parallel data transmission Distinguish between simplex, duplex and half-duplex data transmission Show understanding of the reasons for choosing serial or parallel data transmission Show understanding of the need to check for errors Explain how parity bits are used	Teacher leads discussion of what is meant by transmission of data; the need to check for errors; differences between serial and parallel data transmission. **(W)** Learners complete short case study scenario questions on the need to check for errors. **(I)** In pairs, learners identify current uses of serial and	Notes on bandwidth: www.teach-ict.com/as_a2_ict_new/ocr/A2_G063/333_networks_coms/bandwidth/home_bandwidth.html Pages on serial, parallel, USB, etc.: http://computer.howstuffworks.com/computer-buses-channel.htm Example of simplex, duplex and half duplex: http://www.iec-usa.com/Browse05/DTHFDUP.html Pages on error checking (checksum, CRC): http://computer.howstuffworks.com/encryption7.htm

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
	for error detection Show understanding of the use of serial and parallel data transmission, in Universal Serial Bus (USB) and Integrated Circuit (IC)	parallel data transmission e.g. Integrated Circuits (IC), Universal Serial Bus (USB); reasons for choosing serial or parallel data transmission. **(P)**	
1.2.3 1.1.2	Show understanding of the role of the browser and internet server Show understanding of the concepts of MAC address, Internet Protocol (IP) address, Uniform Resource Locator (URL) and cookies Show understanding of what is meant by hypertext transfer protocol (http and https) and HTML Distinguish between HTML structure and presentation Identify current uses of hexadecimal numbers in computing, such as defining colours in Hypertext Markup Language (HTML), Media Access Control (MAC) addresses, assembly languages and machine code, debugging	What is a network? Class brainstorms the concepts of local area network (LAN), wide area network (WAN), shared resources, communications. **(W)** Learners construct a virtual network. **(G)** Teacher presentation of the Royal Institution Christmas Lecture 'Untangling the Web' – about 35 minutes **(W)** plus time for questions during pauses. Class brainstorm/ teacher input to enable learners to: • describe the nature of the internet as a worldwide collection of computer networks. • describe the hardware needed to connect to the internet including modems, routers etc. • explain the need for IP addressing of resources on the Internet and how this can be facilitated by the role of Domain Name System (DNS) servers; Media Access Control (MAC) address; cookies. • explain the importance of HTML and its derivatives as a standard for the creation of web pages. **(W)**	What is a network?: www.bbc.co.uk/schools/gcsebitesize/ict/datacomm/ Simulated network builder: www.gcsecomputing.org.uk/support/network/NWB_SIM.swf R I lecture video – Untangling the web: www.richannel.org/christmas-lectures/2008/2008-chris-bishop#/christmas-lectures-2008-chris-bishop-untangling-the-web Video – history of the internet (8 mins, animation): http://vimeo.com/2696386?pg=embed&sec=2696386 What is the internet? Activity: www.bbc.co.uk/schools/gcsebitesize/ict/datacomm/0internet_act.shtml Notes and activities on networks and network OS: www.teach-ict.com/gcse_computing/ocr/213_software/os_types/miniweb/pg4.htm www.teach-ict.com/gcse_computing/ocr/215_communications_networking/networks/home_networks.htm http://gcsecomputing.net/wp-

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
			content/uploads/2012/01/OCR-A451-2.1.6-Networks-Summary.pdf http, html and associated terms: www.webopedia.com/TERM/H/HTTP.html
1.1.3	Show understanding that sound (music), pictures, video, text and numbers are stored in different Formats Identify and describe methods of error detection and correction, such as parity checks, check digits, checksums and Automatic Repeat reQuests (ARQ) Show understanding of the concept of Musical Instrument Digital Interface (MIDI) files, JPEG files, MP3 and MP4 files Show understanding of the principles of data compression (lossless and lossy compression algorithms) applied to music/video, photos and text files	Brainstorm to identify common file standards associated with the internet such as JPG, GIF, PDF, MP3, MPEG. Also address other file types mentioned in syllabus such as Musical Instrument Digital Interface (MIDI). **(W)** Teacher summarises these, then looks at issues of error detection and correction. Teacher presentation to explain the importance of compressing files that are transmitted via the internet: • describe the differences between lossy and lossless compression • describe methods of error detection and correction e.g. parity checks, check digits, checksums, Automatic Repeat reQuests (ARQ). **(W)** Learners complete a multiple-choice quiz. **(I)**	Lossy and lossless compression notes: http://computer.howstuffworks.com/file-compression3.htm Notes on ARQ: http://en.wikipedia.org/wiki/Automatic_repeat_request
1.2.3	Show understanding of the role of the browser and internet server Distinguish between HTML structure and presentation	Examination of browser screen to identify key components; comparison of two or more browsers. **(G)** Learner individual work – design a simple web page/website to distinguish between HTML structure and presentation. **(I)** **Extension work:** Learners could create a mini website with several linked pages, including hyperlinks to external websites.	Browsers could include Internet Explorer, Firefox, Chrome, Opera, Safari, etc. www.w3schools.com/ Flash website: http://helpx.adobe.com/flash.html

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
		Learners could also write Flash animations or write Java scripts. **(G)/(I)**	
1.2.2 1.4.2	Show understanding of the security aspects of using the Internet and understand what methods are available to help minimise the risks Show understanding of the Internet risks associated with malware, including viruses, spyware and Hacking Show understanding of how data are kept safe when stored and transmitted, including: Use of passwords, both entered at a keyboard and biometric Use of firewalls, both software and hardware, including proxy servers Use of security protocols such as Secure Socket Layer (SSL) and Transport Layer Security (TLS) Use of symmetric encryption (plain text, cypher text and use of a key) showing understanding that increasing the length of a key increases the strength of the encryption	Learners work in groups to research and create a leaflet/web pages on internet safety, addressing: • viruses, spy-ware and hacking and report the internet risks associated with these • what methods are available to help minimise the risks • how anti-virus and other protection software help to protect the user from security risks • the use of passwords, both entered at a keyboard and biometric. **(G)** Teacher input to describe technical/practical issues around use of firewalls both software and hardware, including proxy servers; use of Secure Socket Layer (SSL). **(W)** Learners carry out coding and decoding activities in pairs **(P)**: use of symmetric encryption (plain text, cypher text, use of a key) – differentiation by task will make this more challenging where necessary.	Lots of information on computer security: http://computer.howstuffworks.com/computer-internet-security-channel.htm Information on SSL: http://en.wikipedia.org/wiki/Secure_Sockets_Layer *Cambridge IGCSE Computer Studies Coursebook* pp. 207–9 – internet security pp. 222–3 – firewalls, etc. Information on symmetric encryption: http://en.wikipedia.org/wiki/Symmetric_encryption Notes on encryption: www.igcseict.info/theory/4/secure/index.html

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
		Learners attempt to decode some encrypted messages from the Central Intelligence Agency (CIA). **(G)** Class brainstorm code-breaking and relative ease of breaking each method. **(W)**	CIA code-breaking for kids: https://www.cia.gov/kids-page/games/break-the-code/code-1.html
1.4.3	Show understanding of the need to keep online systems safe from attacks including denial of service attacks, phishing, pharming	In pairs or individually, learners research and create posters to explain safety from phishing and pharming; other security issues that are down to user carelessness. **(P)/(I)**	Notes on phishing and pharming: www.igcseict.info/theory/6/internet/index.html http://quizlet.com/10713330/phishing-and-pharming-flash-cards/
1.2.1 1.2.3 1.1.2 1.1.3 1.2.3 1.2.2 1.4.2 1.4.3	Formative assessment activities	Pairs of learners devise their own quiz questions (and answers) on this unit **(P)** Teacher selects one or two quizzes to test understanding. **(W)**	

Unit 4: Data integrity and security

Recommended prior knowledge
This unit builds upon the concepts addressed in Unit 3 (Data communications and networking).

Context
With increased use of, and dependence on, computer systems in almost every aspect of modern life, the importance of ensuring accuracy and integrity of information has become paramount. It is possible for data to be corrupted as it is transferred from one system to another – whether between computer systems or when entering data into a computer system. As the value of information held in and transferred between computer systems has increased, there has been a tendency for criminal activity such as hacking to increase at the same time.

Concepts covered in this unit will be used in practical activities in Units 6, 7 and 8.

Outline
The possible causes of loss of data integrity and security are considered, followed by a discussion of preventative measures. The ethical and legal issues surrounding the use of computer systems are also addressed. Note that learners are not required to know details of specific viruses and will not gain credit for doing so.

Suggested teaching time
Based on a total time allocation of 130 contact hours for this Cambridge O Level Computer Science course, it is recommended that this unit should take about 10 hours.

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
1.4.1 2.1.1	Show understanding of the need to keep data safe from accidental damage, including corruption and human errors Understand the need for validation and verification checks to be made on input data (validation could include range checks, length checks, type checks and check digits)	Brainstorm ideas such as corruption of data (reflect back on data transmission) and human error (verification) to raise a range of issues. **(W)** Learners then follow this with internet-based research to produce their own notes. **(I)** Teacher provides list of common applications (e.g. car registrations, test marks, learner names, temperatures, salaries) and learners identify possible validation rules **(P)**. Teacher adds those that the class has not identified (need to cover range checks, length checks, type checks and check digits). **(W)** Calculation of check digits using ISBN (for example) by teacher demonstration and learner completion of a worksheet with a selection of graded problems – both calculating check digit and checking given ISBN codes for validity. **(I)** Class brainstorm and reflect the importance of verification when data is transferred between media and discuss possible strategies for verifying input. **(W)**	Notes from BBC Bite Size: www.bbc.co.uk/schools/gcsebitesize/ict/databases/6datasecurityrev1.shtml and from 'Teach ICT': www.teach-ict.com/gcse_new/protecting_systems/protecting_systems/home_protecting_systems.htm *Cambridge IGCSE Computer Studies Coursebook* pp. 33–7 *Cambridge IGCSE Computer Studies Revision Book* Chp 11.2 Notes, quizzes and activities for data validation: www.teach-ict.com/gcse_computing/ocr/databases/validating/home_validating.htm www.bbc.co.uk/schools/gcsebitesize/ict/databases/3datavalidationrev1.shtml
1.4.1 1.5	Show understanding of the need to keep data safe from malicious actions, including unauthorised viewing, deleting, copying and corruption Show understanding of the ethical issues raised by the spread of electronic communication and computer systems, including hacking,	Brainstorm possible malicious actions (including unauthorised viewing, deleting, copying and corruption). **(W)** Each group of learners researches one of the issues and does a short presentation to explain the implications and ways of preventing each issue. No knowledge of any specific virus (name, action, etc.) is needed. **(G)**	How to prevent computer misuse: www.bbc.co.uk/schools/gcsebitesize/ict/legal/1dataandcomputermisuserev2.shtml The internet and crime: www.bbc.co.uk/schools/gcsebitesize/ict/implications/0moralandsocialissuesrev3.shtml Hacking and data theft: www.teach-ict.com/gcse_new/protecting_systems/hackers/mini

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
	cracking and production of malware		web/index.htm
1.4.3 1.4.4	Show understanding of the need to keep online systems safe from attacks including denial of service attacks, phishing, pharming Describe how the knowledge from 1.4.1, 1.4.2 and 1.4.3 can be applied to real-life scenarios including, for example, online banking, shopping	If learners have already worked through Unit 3, this will be revision in pairs or individually. **(P)/(I)** Learners research and create posters to explain safety from phishing and pharming; other security issues that are down to user carelessness. **(G)** Teacher sets real-life scenarios to allow each group of learners to investigate a unique scenario. This leads to the production of a poster/ leaflet/ presentation covering ways in which real-life scenarios can be compromised and how the issues can be addressed. **(G)** If Unit 3 has not been covered yet, the activity above (phishing and pharming) can be incorporated here. **(P)/(I)**	Notes on phishing and pharming: www.igcseict.info/theory/6/internet/index.html Phishing and pharming: http://quizlet.com/10713330/phishing-and-pharming-flash-cards/ Notes on computer crime: www.teach-ict.com/gcse_new/protecting_systems/crime/home_ictcrime.htm
1.5	Show understanding of computer ethics, including copyright issues and plagiarism Distinguish between free software, freeware and shareware	Teacher introduces issues of copyright and plagiarism. Class brainstorms these issues leading to short notes. **(W)** Quizzes/tests to assess understanding of the issues raised in Unit.4. **(W)/(I)**	A range of quizzes are available at http://quizlet.com/subject/computer-ethics/ They can be used online, as matching cards or for revision.

Unit 5: Binary logic

Recommended prior knowledge
No prior knowledge is needed to start this unit.

Context
This unit introduces logic gates, which are the building blocks for the relatively complex memory and processor circuits found in computers. At this level, to achieve a more concrete grasp of their fundamental properties, they are treated only as components in relatively simple, stand-alone logic circuits. For this unit, together with Units 6 and 7, *Computer Studies Support Booklet – Part 3* (http://teachers.cie.org.uk/docs/dynamic/31798.pdf) provides notes and practice problems (with answers in *Computer Studies Support Booklet – Answers* (http://teachers.cie.org.uk/docs/dynamic/31801.pdf).
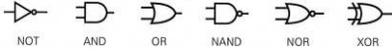
Outline
This unit introduces truth tables and symbols for one-input NOT, and for two-input AND, OR, NAND, NOR and XOR (EOR) logic gates. This is developed to look at truth tables for more complex given logic circuits (with a maximum of 3 inputs and 6 gates). Learners will produce a simple logic circuit from a written design brief.

WARNING: Practical work with logic gate chips can be relatively cheap to perform and very rewarding, but is hazardous if teachers and learners are not properly aware of the risks from connecting modules or electronic components in ways for which they are not intended. For example, if a LED is connected directly across a power supply without a current-limiting series resistor, it is liable to explode in a way that could cause permanent damage to an unprotected eye. See, for example, www.youtube.com/watch?v=L85UNTW4lgU. Practical work can be performed quite adequately with free logic simulation software.

Suggested teaching time
Based on a total time allocation of 130 contact hours for this Cambridge O Level Computer Science course, it is recommended that this unit should take about 15 hours.

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
1.3.1	Understand and define the functions of NOT, AND, OR, NAND, NOR and XOR (EOR) gates, including the binary output produced from all the possible binary inputs (all gates, except the NOT gate, will have 2 inputs only) Draw truth tables and recognise a logic gate from its truth table Recognise and use the following	Teacher introduces the concepts of OR and AND by careful use of English in logical statement; linking TRUE and FALSE to binary states (1 and 0). **(W)** Teacher demonstrates OR gate and AND gate by use of electrical model, projector presentation of simulation, or similar. Teacher develops the concept of truth table for OR gate; learners develop truth table for AND gate. Teacher introduces the standard symbols for both gates. **(W)**	Notes on binary logic (includes extension materials): http://gcsecomputing.net/wp-content/uploads/2012/01/OCR-A451-2.1.2-Binary-Logic-Summary.pdf Introduction, notes and activities at: http://nrich.maths.org/6023 http://nrich.maths.org/5974 A useful presentation on binary logic is: www.mrfraser.org/resources/gcse/presentations/A4

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
	standard symbols used to represent logic gates: NOT AND OR NAND NOR XOR	Learners use appropriate hardware or simulation software to develop understanding of the functions of the NOT, NAND, NOR and XOR (EOR) logic gates. Learners can observe the output produced from all possible combinations of inputs to construct each gate's truth table. **(P)/(I)** **Extension work:** Learners work out the simple logic circuits required to create NAND and NOR gates using AND, OR and NOT gates and test them. **(P)/(I)**	51_presentations.php?topic=2_2 (need to register) Simple logic simulator using standard symbols: http://logic.ly/demo/ More functions available, but uses square boxes for gates: www.neuroproductions.be/logic-lab/ Downloadable logic gate simulator: www.softpedia.com/get/Others/Home-Education/Logic-Gate-Simulator.shtml and www.logiccircuit.org/ *Cambridge IGCSE Computer Studies Coursebook* pp. 274–7 *Cambridge IGCSE Computer Studies Revision Book* Chp 9
1.3.1	Produce truth tables for given logic circuits Produce a logic circuit to solve a given problem or to implement a given written logic statement, such as IF (switch A is NOT on) OR (switch B is on AND switch C is NOT on) then alarm, X, sounds Use logic gates to create electronic circuits (please see warning in the outline for this unit)	Learners should perform a range of graded practical exercises, using additional columns for intermediate outputs, to produce truth tables for given logic circuits (maximum of three inputs and 6 gates). **(G)/(I)** Learners should perform practical exercises to design, build and test a simple logic circuit from a given written statement (e.g. if A OR B are on AND if C is on, then the lights will be on). **(G)/(I)** **Extension work:** Some learners may be able to work algebraically, as would be necessary for circuit simplification beyond the scope of this syllabus.	Various websites with practical exercises can be found at: http://docsfiles.com/pdf_logic_gates_exercises.html Some interesting challenges (and solutions): http://nrich.maths.org/5973 Some more simple examples: www.mrfraser.org/resources/gcse/misc/A451_22_BinaryLogic_Exercises.pdf *Cambridge IGCSE Computer Studies Coursebook* pp. 277–281 *Cambridge IGCSE Computer Studies Revision*

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
		• Work out how to create NOT, AND and OR gates using *only* NAND gates and test the solutions. • Work out how to create NOT, AND and OR gates using *only* NOR gates and test the solutions. **(P)/(I)**	*Book* Chp 9 Some solutions here: http://nrich.maths.org/5967/solution

Unit 6: Practical problem solving – structure diagrams, algorithms and flowcharts

Recommended prior knowledge
No prior knowledge is needed to start this unit.

Context
This unit starts the teaching and learning associated with Paper 2 of the examination. The skills, knowledge and understanding associated with practical problem solving begins with an introduction to algorithms, and to flowcharts as a visual method of representing these. Flowcharts may be considered a more effective way of introducing problem solving rather than beginning with pseudocode.

Computer Studies Support Booklet – Part 3 (http://teachers.cie.org.uk/docs/dynamic/31798.pdf) includes notes on Section 2 of the syllabus and practice problems (with answers in *Computer Studies Support Booklet – Answers* (http://teachers.cie.org.uk/docs/dynamic/31801.pdf). It is therefore relevant to this unit and to Unit 7.

Outline
The unit starts with use of structure diagrams, leading to the definition of algorithms and their representation as flowcharts. This is followed by the use of dry runs and trace tables to work out the purpose of an algorithm, suggesting and using suitable test data and identifying and correcting errors in algorithms. Study of these topics can be illustrated by case studies of existing solutions to problems, and reinforced through practical work.

Suggested teaching time
Based on a total time allocation of 130 contact hours for this Cambridge O Level Computer Science course, it is recommended that this unit should take about 12 hours.

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
2.1.1	Show understanding that every computer system is made up of sub-systems, which in turn are made up of further sub-systems	Class brainstorms a non-computer system to show that it is comprised of sub-systems. **(W)** Learners analyse a relevant and appropriate system to identify sub-systems, and to sub-divide these. This could be the academic or personnel structure of a school/college, a department store, a large company, etc. Structure diagrams can be used to document this. **(G)**	*Cambridge IGCSE Computer Studies Coursebook* pp. 242–7 *Cambridge IGCSE Computer Studies Revision Book* Chp 7.1 A tutorial on how to draw structure diagrams using SmartDraw, of which a free, trial version is available at: www.smartdraw.com/resources/tutorials/
2.1.1	Use top-down design, structure diagrams, flowcharts, pseudocode, library routines and	Learners are introduced to the need for algorithms in developing software solutions. In pairs they can identify the sequence of operations required to carry	An introduction to algorithmic thinking: http://raptor.martincarlisle.com/Introduction%20to%20Algorithmic%20Thinking.doc

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
	subroutines Work out the purpose of a given algorithm Explain standard methods of solution	out a simple multi-stage everyday process such as making a cup of tea/coffee, preparing a meal, etc. **(P)/(I)** Teacher presents a simple flowchart to show: • flow of control/ data • explanation of terminator, input/output, process and decision boxes. Use of formal or informal variable names and mapping of values (e.g. $x \leftarrow 3$ means the value 3 is written as the new value stored in the memory location labelled x, $x \leftarrow y$ means the value stored in the memory location labelled y is copied to the memory location labelled x) • use of conventional mathematical operators (+, -, *, /). **(W)** Learners carry out analysis of prepared flowcharts to work out purpose. The difficulty/ complexity of the flowchart can be increased to make it more challenging where necessary. **(G)/(I)**	PowerPoint presentation on flowcharts and program design: http://staffweb.itsligo.ie/staff/bmulligan/Lectures/CS French/French08.ppt Some self-checking flowchart exercises, with outline structure and available operations: www.cimt.plymouth.ac.uk/projects/mepres/book8/bk 8i1/bk8_1i2.htm *Cambridge IGCSE Computer Studies Coursebook* pp. 247–51 *Cambridge IGCSE Computer Studies Revision Book* Chp 7.2
2.1.1	Suggest and apply suitable test data (The text in italics may have already been taught in Unit 4 – this provides an opportunity for revision and practical application) *Understand the need for validation and verification checks to be made on input data (validation could include range checks, length checks, type*	Brainstorm to consider the limits for input data in any system; identify possible different types of input data (normal, boundary/extreme and abnormal/erroneous). **(W)** Learners identify examples of each type for a range of given situations, as test data. **(P)** *Teacher explains need for validation checks to prevent input of incorrect data; teacher provides list of common applications (e.g. car registrations, test marks, learner names, temperatures, salaries) and learners identify possible validation rules.***(G)/(I)** *Teacher adds those that the class has not identified (need to cover range checks, length checks, type checks and check digits).* **(W)**	*Cambridge IGCSE Computer Studies Coursebook* pp.33–37 *Cambridge IGCSE Computer Studies Revision Book* Chp 11 Notes, quizzes and activities for data validation: www.teach-ict.com/gcse_computing/ocr/databases/validating/home_validating.htm www.bbc.co.uk/schools/gcsebitesize/ict/databases/3datavalidationrev1.shtml www.klbict.co.uk/gcse/theory/5_3/5_3_3_valid_verif.htm

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
	checks and check digits)	Teacher demonstrates introduction of validation checks with decision boxes in a sample flowchart; learners add validation to existing flowcharts. **(W)**	
		Calculation of check digits using ISBN (for example) by teacher demonstration and learner completion of a selection provided. **(I)**	
		Brainstorm the importance of verification when data is transferred between media (design flowchart for double entry of e.g. password). **(W)**	
	Use trace tables to find the value of variables at each step in an algorithm	Teacher demonstrates design and completion of trace table for dry runs of a simple flowchart. **(W)**	
	Identify errors in given algorithms and suggest ways of removing these errors	Learners carry out exercise on flowcharts (opportunity for differentiation by complexity of flowchart). **(P)/(I)**	
		Learners use a trace table to analyse a flowchart for an incorrect algorithm and identify the source of the error. **(P)/(I)**	
2.1.1	Produce an algorithm for a given problem (either in the form of pseudocode or flowchart) Comment on the effectiveness of a given solution	Learners should perform practical exercises to demonstrate solution design such as: • finding the average of a set of input numbers • finding largest and smallest numbers in a set of input numbers calculating the frequency distribution of ranges of numbers in a set of input numbers (e.g. when a series of temperatures T are input, how many are in each of the ranges $-20 \le T < 0$, $0 \le T < 20$ and $20 \le T < 40$?) These could be followed by case studies and questions. For example, an automatic supermarket stock control system for calculating stock levels and	*RAPTOR*, free program flowchart interpreter software that allows learners to draw a flowchart and check its functioning by executing it: http://raptor.martincarlisle.com/ *Cambridge IGCSE Computer Studies Coursebook* pp. 240–2 *Cambridge IGCSE Computer Studies Revision Book* 7.3 – example questions in Chp 7.5

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
		automatically re-ordering items. **(G)/(I)** **Extension work:** Learners answer more complex questions on the case study. Learners' capability in generating an algorithm and presenting it as a flowchart, and their analysis of prepared flowcharts, can be assessed by completion of some test questions before moving on to pseudocode in Unit 7. **(I)**	

Unit 7: Practical problem solving – pseudocode

Recommended prior knowledge
Learners need to have studied Unit 6 before starting this unit.

Context
This unit develops the methodologies of problem solving, from the visual representations of algorithms as flowcharts, towards the final computer program by generating pseudocode representations. Suggested basic syntax for pseudocode is given in the syllabus. Further notes and practice problems can be found in the *Computer Studies Support Booklet – Part 3* (http://teachers.cie.org.uk/docs/dynamic/31798.pdf) (with answers in *Computer Studies Support Booklet – Answers* (http://teachers.cie.org.uk/docs/dynamic/31801.pdf).

Outline
Learners are introduced to different data types, allowing the development of an understanding of pseudocode as a way of representing an algorithm prior to formal coding in a programming language. Operators for assignment and structures for conditional statements and iteration are introduced. Algorithms previously represented as flowcharts are converted to pseudocode, and arrays are introduced.

Suggested teaching time
Based on a total time allocation of 130 contact hours for this Cambridge O Level Computer Science course, it is recommended that this unit should take about 12 hours.

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
2.2.1 2.1.2	Declare and use variables and constants Understand and use basic data types: Integer, Real, Char, String and Boolean Understand and use pseudocode for assignment, using ← Understand and use pseudocode, using the following commands and statements: INPUT and OUTPUT (e.g. READ and PRINT) totalling (e.g. Sum ← Sum + Number) counting (e.g.	Teacher introduces concepts of constants and variables; brainstorm to identify basic data types. **(W)** Learners obtain definitions of each type from web research. **(P)/(I)** Teacher explains the role of pseudocode as a step between informal problem-solving or use of flowcharts and the formal approach of a programming language. **(W)** Using a simple algorithm, represented as a paragraph of English or as a flowchart, learners convert this to pseudocode. Simple examples could be numerical problems such addition, subtraction, multiplication,	*Cambridge IGCSE Computer Studies Coursebook* pp. 39–42 Theory notes on data types: www.teach-ict.com/gcse_computing/ocr/216_programming/handling_data/home_handling_data.htm *Cambridge IGCSE Computer Studies Coursebook* pp. 260-8 *Cambridge IGCSE Computer Studies Revision Book* Chp 8.3 Pseudocode in 'Absolute beginner's guide to programming':

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
	Count ← Count + 1) Understand and use pseudocode, using the following conditional statements: IF ... THEN ... ELSE ... ENDIF CASE ... OF ... OTHERWISE ... ENDCASE Understand and use standard flowchart symbols to represent the above statements, commands and structures	division; more challenging examples could be set in the real world, such as the use of an ATM. **(G)** Class brainstorm to revise use of variable names, assignment commands (←) and arithmetic operators (+, -, *, /); teacher introduces input and output commands (e.g. READ and PRINT), conditional statements (IF ... THEN ... ELSE ... ENDIF, CASE OF ... OTHERWISE ... ENDCASE). **(W)** Learners convert previous flowchart algorithms to pseudocode. **(G)/(I)**	http://books.google.co.uk/
2.1.2 2.2.2	Understand and use pseudocode, using the following loop structures: FOR ... TO ... NEXT REPEAT ... UNTIL WHILE ... DO ... ENDWHILE Declare and use one-dimensional arrays, for example: A[1:n] Show understanding of the use of one-dimensional arrays, including the use of a variable as an index in an array Read or write values in an array using a FOR ... TO ... NEXT loop	Introduction of FOR ... TO ... NEXT loop command by teacher. **(W)** Learners use this (for example) to input specific number of items to calculate the average. Stress the need to initialise variables before the loop and to output results after exiting the loop. **(G)/(I)** Teacher leads an introduction to arrays, explaining how to declare the size of one-dimensional arrays; for example: A[1:n]; the use of index variables in arrays; reading values into an array using a FOR … TO … NEXT loop to increment the index variable. **(W)** Learners amend the previous task to read a set of data into an array and calculate the average. **(G)** Teacher introduces REPEAT ... UNTIL and WHILE ... DO ... ENDWHILE loops, explaining difference of bottom-testing and top-testing. **(W)** Learners identify which type of loop is most	Notes on arrays: www.teach-ict.com/gcse_computing/ocr/216_programming/handling_data/miniweb/pg10.htm *Cambridge IGCSE Computer Studies Revision Book* Chp 8.4 – example questions in 8.5

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
		appropriate for about six different scenarios; explain rationale to class, who agree (or otherwise) on choice and rationale. **(G)**	
		Teacher introduces nested loops; learners design algorithms using nested loops. **(W)**/**(G)**	
		Summative testing of learners' capability with pseudocode can be made using exam-type questions.	

Unit 8: Programming concepts

Recommended prior knowledge
Learners need to have studied Units 6 and 7 before starting this unit.

Context
This unit completes the process of converting an algorithm from an abstract idea to a working computer program. A range of different types of programming languages exist; this unit looks at the different levels of language and the processes for translation into machine code. It also provides learners with opportunities to convert algorithms into functional programs. Candidates are not expected to have expertise in any specific computer language, but to understand the basic principles of syntax.

It is essential that learners have the opportunity of writing programs using a high-level programming language, such as Visual Basic, Delphi/Pascal or Python. Scratch and a control programming language could be used during the early stages of the delivery of this unit in order to introduce learners to programming concepts and routines. References to programming languages are given in the resource lists below.

Outline
Following consideration of the concepts of sequence, selection and repetition, writing an algorithm as a flowchart and in pseudocode, and identifying and correcting errors in pseudocode, this unit looks at the need for high-level and low-level languages. It considers the use of assemblers, interpreters and compilers for translation of the code written by a programmer into machine code that can be used by the processor.

Learners have the opportunity of using a number of different high-level languages to produce working programs. They will be able to develop their programming skills, of iteration by the use of FOR…NEXT, REPEAT…UNTIL and WHILE…DO loops and to incorporate the use of arrays into their programming.

Suggested teaching time
Based on a total time allocation of 130 contact hours for this Cambridge O Level Computer Science course, it is recommended that this unit should take about 12 hours.

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
1.3.7	Show understanding of the need for both high-level and low-level languages Show understanding of the need for assemblers when translating programs written in assembly language	Whole class brainstorms the nature of a program and its requirements (data input and output; manipulation of data of various types and structures; sequence, selection, repetition and subprogram intercommunication; the concepts of totals and counting). **(W)** Teacher introduces learners to different types of programming languages by considering: • historical origins of computer programming in machine-specific types of language (machine language and assembly language) • the characteristics of these languages • the need for an assembler translation program for assembly language • why they are still used for certain applications. **(W)**	*Cambridge IGCSE Computer Studies Coursebook* pp. 255–9 *Cambridge IGCSE Computer Studies Revision Book* Chp 8.1 An introduction to different levels of programming language: www.teach-ict.com/gcse_computing/ocr/216_programming/programming_languages/home_programming_languages.htm
1.3.7	Show understanding of the need for compilers when translating programs written in a high-level language Show understanding of the use of interpreters with high-level language programs	Learners research the characteristics of high-level languages; the need for compiler and/or interpreter translation programs for these languages; why they are preferred for many applications. **(G)/(I)**	*Cambridge IGCSE Computer Studies Coursebook* pp. 257–259 Introduction to high-level language: www.teach-ict.com/gcse_computing/ocr/216_programming/ **Extension work:** History of compiler writing: http://en.wikipedia.org/wiki/History_of_compiler_writing First high-level language to have a complete compiler: http://en.wikipedia.org/wiki/Fortran The first programming language to express operations using English-like statements:

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
			http://en.wikipedia.org/wiki/FLOW-MATIC
2.2.1	Understand and use the concepts of sequence, selection, repetition, totalling and counting Use predefined procedures/functions	Introduction to programming with Scratch – teacher presentation should cover: • different data types and their declaration • iteration, counting and totalling – implementation of some examples previously devised as pseudocode representations • calling procedures/functions/sub-routines. **(W)** Followed by a range of learner practical activities. These can be differentiated by task to provide appropriate challenge for learners. **(G)/(I)** Repetition of the previous sequence of activities using: • a control programming language (e.g. GO, Logo, Flowol) • a more conventional procedural language such as Visual Basic, Python or Pascal. **(G)/(I)**	'Scratch' – simple programming language that makes it easy to create animations, games, music, interactive stories, etc. without complex syntax: http://scratch.mit.edu/ Some simple tasks in Scratch: www.teach-ict.com/programming/scratch/scratch_home.htm Flowol website: www.flowol.com/Default.aspx The Python website: www.python.org/ Free downloadable version of Pascal: www.lazarus.freepascal.org/ Tutorials for various programming languages: www.codecademy.com/learn Some LOGO websites and ideas: www.mathcats.com/gallery/logodownloadinfo.html
2.2.2	Declare and use one-dimensional arrays, for example: A[1:n] Show understanding of the use of one-dimensional arrays, including the use of a variable as an index in an array Read or write values in an array using a FOR … TO … NEXT loop	Learners write programs from algorithms developed in the previous unit (Unit 7), to read values from a data source (data statement, keyboard) into a specified array and calculate e.g. average, largest, smallest. These will have been tailored to give appropriate challenge to learners. **(G)/(I)**	Notes on arrays (as for Unit 7): www.teach-ict.com/gcse_computing/ocr/216_programming/handling_data/miniweb/pg10.htm

Unit 9: Databases

Recommended prior knowledge
Learners need to have studied Units 6, 7 and 8 before starting this unit.

Context
This unit draws together skills, knowledge and understanding developed in previous units and applies them to data handling situations. It is expected that learners will use data handling software such as Microsoft Access or Base from OpenOffice to generate single-table databases and to retrieve specific data from them. Online database systems such as search engines also provide opportunities to use and refine queries.

Examination questions testing learners' understanding of performing queries will include the database structure and syntax for carrying out queries.

Outline
This unit begins with a review of database systems that learners will be familiar with, to develop the basic concepts of records and fields. Data types are re-visited, considering the storage requirements for each type. This leads to calculations of field, record and file sizes for those examples being considered. Learners then consider a context for defining and developing a single-table database for a familiar data-handling situation. Characteristics of a primary key are developed, and applied to the learners' databases. Strategies and techniques for retrieval of data are then developed.

Suggested teaching time
Based on a total time allocation of 130 contact hours for this Cambridge O Level Computer Science course, it is recommended that this unit should take about 9 hours.

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
2.3	Define a single-table database from given data storage requirements Choose and specify suitable data types	Teacher leads a review of a range of databases (manual, printed, electronic). Refer back to data types (Unit 7); identify structure of each record, data type of each field, field size, and record size. Issues of coding data should also be discussed. **(W)** Assess learners' understanding with a quiz or similar activity. A pre-prepared set of questions can be used, or groups of learners could produce their own questions to test others. **(W)/(G)/(I)** Class brainstorm to identify appropriate contexts for learners' own database design, relevant to their experience and interests e.g. school register, clubs,	*Cambridge IGCSE Computer Studies Coursebook* pp. 37–41 *Cambridge IGCSE Computer Studies Revision Book* pp. 140–4 Overview of database issues: www.teach-ict.com/gcse_computing/ocr/databases/concepts/miniweb/index.htm Databases and data capture: www.bbc.co.uk/schools/gcsebitesize/ict/databases/2databasesrev1.shtml

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
		library, video hire, stock in small shop. **(W)** Learners select a context, identify fields, calculate field lengths, data types in each field, consider coding and validation for each field. **(G)/(I)** Learners create a sample database in a sensible context; each learner needs to add about 20 records (enough to search later), so a data identification/ collection exercise will be needed. This could be done as homework. **(G)/(I)** Use software to create this database. Learners will need instruction on using the software if they have not used it before. **(W)/(G)/(I)**	Review of data types: www.teach-ict.com/gcse_new/data_info_knowledge/data_types/miniweb/index.htm Discussion about coding data: www.teach-ict.com/gcse_new/data_info_knowledge/coding/home_coding.htm Quiz-type activities: http://quizlet.com/2647975/gcse-ict-database-keywords-flash-cards/ e.g. Microsoft Access, OpenOffice Base Videos to support first-time use of the software can be found at (Access): www.teach-ict.com/videohome.htm or (Base) http://showmedo.com/videotutorials/series?name=axggl6j0a
2.3	Choose a suitable primary key for a database table	Learners look at database examples again; identify primary key in each case by class discussion. **(W)** Quiz/short questions to identify key fields in five sample databases, of increasing difficulty. **(G)/(I)** Learners identify primary key in their own databases. **(G)/(I)**	Information on record structure and key fields: www.bbc.co.uk/schools/gcsebitesize/dida/using_ict/databasesrev2.shtml Data organisation: www.igcseict.info/theory/5/dbase/index.html
2.3	Perform a query-by-example from given search criteria	Teacher leads a discussion of query structure and strategies. **(W)** Learners carry out examples of querying using specimen databases. Learners write and carry out searches on their own databases. This is then repeated using other learners' databases and they can give constructive criticism of other learners' work. **(G)/(I)**	Overview of searching: www.bbc.co.uk/schools/gcsebitesize/ict/databases/2databasesrev5.shtml

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
		Common issues can be drawn together by the teacher in a class session. **(W)**	*Cambridge IGCSE Computer Studies Revision Book* pp. 205, 217
		In groups or pairs learners carry out searches using search engines, using Boolean logic. e.g. Google, Yahoo. **(G)/(P)**	
		Learners' understanding can be tested with questions from textbooks and/or past/specimen papers. **(G)/(I)**	See past papers for syllabus 7010 and specimen papers for syllabus 2210 for questions on this topic available at http://teachers.cie.org.uk

Unit 10: Use of pre-release materials

Recommended prior knowledge
Learners need to be proficient in the problem solving and programming techniques specified in the syllabus. Therefore, this unit should not be attempted until Units 6, 7 and 8 have been completed. It cannot be started until the pre-release materials have been received.

Context
The pre-release materials will contain specifications for a computer program that must be developed by each learner. Paper 2 will require learners to answer specific questions about issues that will have been addressed in the course of the development of their own solution. These questions carry a total of 20 marks, contributing 16% of the total marks for the certification.

This unit also provides opportunities to revise knowledge, skills and understanding developed in all the previous units.

Outline
Learners receive the pre-release materials in advance of the examination. This includes a scenario and a number of specific tasks that need to be addressed in relation to this scenario. Learners are required to consider each of the tasks and create a computer program using a procedural high-level programming language such as Visual Basic, Python or Pascal/Delphi, that will provide a solution to these tasks. Teachers are expected to give support to learners in finding methods and reaching solutions.

Suggested teaching time
Based on a total time allocation of 130 contact hours for this Cambridge O Level Computer Science course, it is recommended that this unit should take about 20–25 hours.

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
2.1.1 2.1.2 2.2.1 2.2.2	Use skills knowledge and understanding to find methods to address the tasks specified in the pre-release materials	Teacher presents the pre-release materials and holds a class discussion to ensure that all the learners have a clear understanding of what is required of them. **(W)** Initial class brainstorm to consider possible methods of solution; learners then work in groups or individually to refine solutions by: • designing algorithms • creating flowcharts and/or pseudocode • testing these with appropriate test data • coding the solution using an appropriate high-level language • testing the program with appropriate test data.	As specified in Units 6, 7 and 8.

Syllabus ref	Learning objectives	Suggested teaching activities	Learning resources
		(G)/(I) Teacher monitors progress and gives support as necessary. More able learners should be encouraged to extend their solution beyond the confines of the specifications given in the pre-release materials; for example, they may design input/output screens, and extend the functionality of their programs. **(G)/(I)/(P)**	

A DIVISION OF
CAMBRIDGE ASSESSMENT