COMPUTER SCIENCE

Year 12

What are the aims and intentions of this curriculum?

This subject combines invention and excitement where students will look at the natural world through a digital prism. There are two units within this curriculum map, which are delivered concurrently. OCR's Computer Science will value computational thinking, helping learners to develop the skills to solve problems, design systems and understand the power and limits of human and machine intelligence. **At the end of this curriculum, students should be able to:**

- have an understanding and knowledge of the internal components of a computer system.
- know how to convert elements of data into machine code.
- understand the fundamentals of software development.
- apply their knowledge of laws and regulations that governs legal and ethical issues in computing.
- justify the application of various technology in different contexts including current and future uses.
- understand the benefits of applying computational thinking to solving problems.
- analyse various problems and apply appropriate algorithms to solve them.

Term	Topics	Knowledge and key terms	Skills developed	Assessment
Autumn 1	**1.1 The characteristics of contemporary processors, input, output and storage devices** **1.1.1 Structure and function of the processor**	**(a)** The Arithmetic and Logic Unit; ALU, Control Unit and Registers (Program Counter; PC, Accumulator; ACC, Memory Address Register; MAR, Memory Data Register; MDR, Current Instruction Register; CIR). Buses: data, address and control: how this relates to assembly language programs. **(b)** The fetch-decode-execute cycle, including its effect on registers. **(c)** The factors affecting the performance of the CPU, clock speed, number of cores, cache. **(d)** Von Neumann, Harvard and contemporary processor architecture.	• Understanding of the fundamental hardware of a computer system • Evaluate the development of computer technology and the effects it has had. • Understand and explain the Fetch-Execute cycle. • Explain the effect of the following on the performance of the CPU: ✓ clock speed ✓ number of processor cores ✓ cache size ✓ cache type. Compare the Von Neumann architecture.	• Group Presentations • Case Studies • End of topic quiz • End of term test • Microsoft Teams collaborative activities. • Home work • Class Discussions • Topic Worksheets • Past Paper question sheets
	1.1.2 Types of processor	**(a)** The differences between and uses of CISC and RISC processors. **(b)** Multicore and Parallel systems.	Develop their understanding of current and emerging technologies and how they work.	
	1.1.3 Input, output and storage	**a)** How different input, output and storage devices can be applied to the solution of different	• Become independent and discerning users of IT.	

		problems. **(b)** The uses of magnetic, flash and optical storage devices. **(c)** RAM and ROM. **(d)** Virtual storage.	• Be able to categorize secondary storage. Understand the differences between RAM and ROM in terms of their application.	
	1.2.3 Introduction to programming	**(a)** Procedural programming language techniques: ✓ program flow ✓ variables and constants ✓ procedures and functions ✓ arithmetic, Boolean and assignment operators ✓ string handling ✓ file handling.	• Use, understand and know how the following statement types can be combined in programs: ✓ variable declaration ✓ constant declaration ✓ assignment ✓ string handling ✓ file handling ✓ subroutine (procedure/function)	
		(b) Assembly language (including following and writing simple programs with Little Man Computer). See appendix 5d.	• Identify and use mnemonics from LMC • Write simple programs using Little Man Computing	
	1.4.1 Data Types	**(a)** Primitive data types, integer, real/floating point, character, string and Boolean. **(b)** Represent positive integers in binary. **(c)** Use of sign and magnitude and two's complement to represent negative numbers in binary. **(d)** Addition and subtraction of binary integers. **(e)** Represent positive integers in hexadecimal. **(f)** Convert positive integers between binary hexadecimal and denary. **(g)** Positive and negative real numbers using normalised floating point representation. **(h)** How character sets (ASCII and UNICODE) are used to represent text.	• Understand the concept of a data type. • Differentiate between the listed data types. • Know how to: ✓ represent negative and positive integers in two's complement ✓ perform subtraction using two's complement • Be able to convert between unsigned binary and decimal and vice versa. • Be able to add and subtract binary as well as to convert between decimal, binary and hexadecimal number bases. • Be familiar with the concept of a number base, in particular: ✓ decimal (base 10) ✓ binary (base 2) ✓ hexadecimal (base 16). Describe ASCII and Unicode coding systems for coding character data and explain why Unicode was introduced.	
	1.4.2 Data Structures	**a)** Arrays (of up to 3 dimensions), records, lists, tuples. **(b)** The properties of stacks and queues.	• Use arrays in the design of solutions to simple problems. Use stocks and queues to structure data.	

Autumn 2	**1.2 Software and software development**	**Types of software and the different methodologies used to develop software**		• Group Presentations • Case Studies • End of topic quiz • End of term test • Microsoft Teams collaborative activities. • Home work • Class Discussions • Topic Worksheets • Past Paper question sheets
	1.2.1 Operating Systems	**(a)** The need for, function and purpose of operating systems. **(b)** Memory Management (paging, segmentation and virtual memory). **(c)** Interrupts, the role of interrupts and Interrupt Service Routines (ISR), role within the Fetch-Decode-Execute Cycle. **(d)** Scheduling: round robin, first come first served, multi-level feedback queues, shortest job first and shortest remaining time. **(e)** Distributed, embedded, multi-tasking, multi-user and real time operating systems. **(f)** BIOS. **(g)** Device drivers. **(h)** Virtual machines	• Understand the relationship between hardware and software • Know that the OS handles interrupts, scheduling, resource management, managing hardware to allocate processors, memories and I/O devices among competing processes. • Understand the term 'embedded system' and explain how an embedded system differs from a Distributed system. • Know the instance where software is used to take on the function of a machine including executing intermediate code or running an operating system within another. • Understand the need for, and attributes of, different types of software.	
	1.2.2 Applications generation	**(a)** The nature of applications, justifying suitable applications for a specific purpose. **(b)** Utilities. **(c)** Open source vs closed source. **(d)** Translators: Interpreters, compilers and assemblers.	• Understand the functions of the following software: ✓ open source ✓ closed source ✓ utility programs ✓ libraries ✓ translators (compiler, assembler, interpreter).	
	1.4.3 Boolean Algebra	**(a)** Define problems using Boolean logic. See appendix 5d. **(b)** Manipulate Boolean expressions, including the use of Karnaugh maps to simplify Boolean expressions. **(c)** Use logic gate diagrams and truth tables.	• Write a Boolean expression for a given logic gate circuit. • Use Karnaugh maps appropriately. • Complete a truth table for a given logic gate circuit. • Construct truth tables for the following logic gates: ✓ NOT ✓ AND ✓ OR	
	2.1 Elements of computational thinking	**Understand what is meant by computational thinking** **(a)** The nature of abstraction.	• Students should have experience of using abstraction to model aspects of the external world in a program.	

	2.1.1 Thinking abstractly	**(b)** The need for abstraction. **(c)** The differences between an abstraction and reality. **(d)** Devise an abstract model for a variety of situations.		
	2.1.2 Thinking ahead	**(a)** Identify the inputs and outputs for a given situation. **(b)** Determine the preconditions for devising a solution to a problem. **(c)** The need for reusable program components.	• Be aware that before a problem can be solved, it must be defined, the requirements of the system that solves the problem must be established	
	2.1.3 Thinking procedurally	**(a)** Identify the components of a problem. **(b)** Identify the components of a solution to a problem. **(c)** Determine the order of the steps needed to solve a problem. **(d)** Identify sub-procedures necessary to solve a problem.	• The capacity to think creatively, innovatively, analytically, logically and critically • Practical skills in the context of solving a realistic problem	
	2.1.4 Thinking logically	**(a)** Identify the points in a solution where a decision has to be taken. **(b)** Determine the logical conditions that affect the outcome of a decision. **(c)** Determine how decisions affect flow through a program.		
Spring 1	**1.3 Exchanging data**	**How data is exchanged between different systems**		• Group Presentations • Case Studies • End of topic quiz • End of term test • Microsoft Teams collaborative activities. • Home work • Class Discussions • Topic Worksheets • Past Paper question sheets
	1.3.1 Databases	**(a)** Relational database, flat file, primary key, foreign key, secondary key, entity relationship modelling. See appendix 5d and 5e. **(b)** Methods of capturing, selecting, managing and exchanging data.	• Distinguish between database keys. • Draw entity relationship diagrams to express a given situation.	
	1.3.2 Networks	**(a)** Characteristics of networks and the importance of protocols and standards. **(b)** Internet structure: ✓ The TCP/IP stack. ✓ DNS. ✓ Protocol layering. ✓ LANs and WANs. ✓ Packet and circuit switching. **(c)** Client-server and peer to peer.	• Appreciate the importance of protocols and standards. • Describe the 4 layer TCP/IP model: ✓ application layer ✓ transport layer ✓ internet layer ✓ link layer. • Explain the following and describe situations	

			where they might be used: ✓ peer-to-peer networking ✓ client-server networking.	
	2.2 Problem solving and programming **2.2.1 Programming techniques**	**(a)** Programming constructs: sequence, iteration, branching. **(b)** Global and local variables. **(c)** Modularity, functions and procedures, parameter passing by value and reference. **(d)** Use of an IDE to develop/debug a program.	• Be able to express the solution to a simple problem as an algorithm using pseudo-code, with the standard constructs: ✓ sequence ✓ branching ✓ iteration • Be able to convert an algorithm from pseudo-code into high level language program code. • Apply the structure of the waterfall lifecycle in software development.	
	2.2.2 Software Development	**(a)** Understand the waterfall lifecycle, agile methodologies, extreme programming, the spiral model and rapid application development. **(b)** The relative merits and drawbacks of different methodologies and when they might be used. **(c)** Writing and following algorithms. **(d)** Different test strategies, including black and white box testing and alpha and beta testing. **(e)** Test programs that solve problems using suitable test data and end user feedback, justify a test strategy for a given situation.	• Discuss relevant software development methodologies including their advantages and disadvantages. Students should have practical experience of designing and applying test data, normal, boundary and erroneous to the testing of programs so that they are familiar with these test data types and the purpose of testing.	
Spring 2	**1.3.3 Web Technologies** **1.5 Legal, moral, ethical and cultural issues** **1.5.1 Computing related legislation**	**(a)** HTML, CSS and JavaScript. See appendix 5d. **(b)** Lossy v lossless compression. The individual moral, social, ethical and cultural opportunities and risks of digital technology. Legislation surrounding the use of computers and ethical issues that can or may in the future arise from the use of computers **(a)** The Data Protection Act 1998. **(b)** The Computer Misuse Act 1990. **(c)** The Copyright Design and Patents Act 1988.	• Be able to build webpages with the implementation of CSS and JavaScript. • Understand the fundamentals of file compression and related calculations. • An understanding of the consequences of using computers unlawfully.	• Group Presentations • Case Studies • End of topic quiz • End of term test • Microsoft Teams collaborative activities. • Home work • Class Discussions • Topic Worksheets

		(d) The Regulation of Investigatory Powers Act 2000.		• Past Paper question sheets
	2.3 Algorithms **2.3.1 Algorithms**	**The use of algorithms to describe problems and standard algorithms** **(a)** Analysis and design of algorithms for a given situation. **(b)** Standard algorithms (bubble sort, insertion sort, binary search and linear search). **(c)** Implement bubble sort, insertion sort. **(d)** Implement binary and linear search. **(e)** Representing, adding data to and removing data from queues and stacks. **(f)** Compare the suitability of different algorithms for a given task and data set.	• Be able to develop solutions to simple logic problems. • Know when and how to use different algorithm sorting and searching methods.	
Summer 1	**1.5.2 Ethical, moral and cultural issues**	**(a)** The individual moral, social, ethical and cultural opportunities and risks of digital technology: ✓ Computers in the workforce. ✓ Automated decision making. ✓ Artificial intelligence. ✓ Environmental effects. ✓ Censorship and the Internet. ✓ Monitor behaviour. ✓ Analyse personal information. ✓ Piracy and offensive communications. Layout, colour paradigms and character sets.	• Understand the professional, ethical, legal, security and social issues and responsibilities • Understand that: ✓ developments in computer science and the digital technologies have dramatically altered the shape of communications and information flows in societies, enabling massive transformations in the capacity to: o monitor behaviour o amass and analyse personal information o distribute, publish, communicate and disseminate personal information.	✓ Group Presentations ✓ Case Studies ✓ End of topic quiz ✓ End of term test ✓ Microsoft Teams collaborative activities. ✓ Home work ✓ Class Discussions ✓ Topic Worksheets ✓ Past Paper question sheets ✓ Internal programming project.
Summer 2	**Alliance Challenge** **Transition to Year 13**	• Projects • Revision • One-to-one tutoring • Alliance Challenge	Internal Programming Project.	✓ Past Paper question sheets ✓ Internal programming project.