	Year 11	Year 11	Year 11	Year 11	Year 11	Year 11	Year 11	Year 11
	G	F	E/D	C	C/B	B	A	A*
	1 Old level 1	2 Old level 1	3 Old level 2	4 Old level 3/4	5 Old level 4	6 Old level 5	7 Old level 6	8 Old level 7
Computer Science	Understands what an algorithm is and is able to express simple linear (non-branching) algorithms symbolically. Understands that computers need precise instructions. Demonstrates care and precision to avoid errors. Knows that users can develop their own programs, and can demonstrate this by creating a simple program in an environment that does not rely on text e.g. programmable robots etc. Executes, checks and changes programs. Understands that programs execute by following precise instructions. Understands that computers have no intelligence and that computers can do nothing unless a program is executed. Recognises that all software executed on digital devices is programmed.	Understands that algorithms are implemented on digital devices as programs. Designs simple algorithms using loops, and selection i.e. if statements. Uses logical reasoning to predict outcomes. Detects and corrects errors i.e. debugging, in algorithms. Uses arithmetic operators, if statements, and loops, within programs. Uses logical reasoning to predict the behaviour of programs. Detects and corrects simple semantic errors i.e. debugging, in programs. Recognises that a range of digital devices can be considered a computer. Recognises and can use a range of input and output devices. Understands how programs specify the function of a general purpose computer.	Designs solutions (algorithms) that use repetition and two-way selection i.e. if, then and else. Uses diagrams to express solutions. Uses logical reasoning to predict outputs, showing an awareness of inputs. Creates programs that implement algorithms to achieve given goals. Declares and assigns variables. Uses post-tested loop e.g. 'until', and a sequence of selection statements in programs, including an 'if, then and else' statement. Knows that computers collect data from various input devices, including sensors and application software. Understands the difference between hardware and application software, and their roles within a computer system. Understands the difference between the internet and internet service e.g. world wide web.	Shows an awareness of tasks best completed by humans or computers. Designs solutions by decomposing a problem and creates a sub-solution for each of these parts (decomposition). Recognises that different solutions exist for the same problem. Understands the difference between, and appropriately uses if and if, then and else statements. Uses a variable and relational operators within a loop to govern termination. Designs, writes and debugs modular programs using procedures. Knows that a procedure can be used to hide the detail with sub-solution (procedural abstraction). Understands why and when computers are used. Understands the main functions of the operating system. Understands how to effectively use search engines, and knows how search results are selected, including that search engines use 'web crawler programs'.	Understands that iteration is the repetition of a process such as a loop. Recognises that different algorithms exist for the same problem. Represents solutions using a structured notation. Can identify similarities and differences in situations and can use these to solve problems. Understands that programming bridges the gap between algorithmic solutions and computers. Has practical experience of a high-level textual language, including using standard libraries when programming. Uses a range of operators and expressions e.g. Boolean, and applies them in the context of program control. Selects the appropriate data types. Defines data types: real numbers and Boolean. Knows that digital computers use binary to represent all data. Understands how bit patterns represent numbers and images. Knows that computers transfer data in binary. Understands the relationship between binary and file size (uncompressed). Recognises and understands the function of the main internal parts of basic computer architecture. Understands the concepts behind the fetch-execute cycle. Understands how search engines rank search results. Understands how to construct static web pages using HTML and CSS. Understands data transmission between digital computers over networks, including the internet i.e. IP addresses and packet switching.	Understands a recursive solution to a problem repeatedly applies the same solution to smaller instances of the problem. Recognises that some problems share the same characteristics and use the same algorithm to solve both (generalisation). Understands the notion of performance for algorithms and appreciates that some algorithms have different performance characteristics for the same task. Uses nested selection statements. Appreciates the need for, and writes, custom functions including use of parameters. Knows the difference between, and uses appropriately, procedures and functions. Understands and uses negation with operators. Uses and manipulates one dimensional data structures. Detects and corrects syntactical errors. Understands how numbers, images, sounds and character sets use the same bit patterns. Performs simple operations using bit patterns e.g. binary addition. Understands the relationship between resolution and colour depth, including the effect on file size. Distinguishes between data used in a simple program (a variable) and the storage structure for that data. Understands the von Neumann architecture in relation to the fetch-execute cycle, including how data is stored in memory. Understands the basic function and operation of location addressable memory.	Recognises that the design of an algorithm is distinct from its expression in a programming language (which will depend on the programming constructs available). Evaluates the effectiveness of algorithms and models for similar problems. Recognises where information can be filtered out in generalizing problem solutions (abstraction). Uses logical reasoning to explain how an algorithm works. Represents algorithms using structured language. Appreciates the effect of the scope of a variable e.g. a local variable can't be accessed from outside its function. Understands and applies parameter passing. Understands the difference between, and uses, both pre-tested e.g. 'while', and post-tested e.g. 'until' loops. Applies a modular approach to error detection and correction. Knows the relationship between data representation and data quality. Understands the relationship between binary and electrical circuits, including Boolean logic. Understands how and why values are data typed in many different languages when manipulated within programs. Knows that processors have instruction sets and that these relate to low-level instructions carried out by a computer. Understands the client-server model including how dynamic web pages use server-side scripting and that web-servers process and store data entered by users.	Designs a solution to a problem that depends on solutions to smaller instances of the same problem (recursion). Understands that some problems cannot be solved computationally. Designs and writes nested modular programs that enforce reusability utilising sub-routines where ever possible. Understands the difference between 'While' loop and 'For' loop, which uses a loop counter. Understands and uses two dimensional data structures. Performs operations using bit patterns e.g. conversion between binary and hexadecimal, binary subtraction etc. Understands and can explain the need for data compression, and performs simple compression methods. Has practical experience of a small (hypothetical) low level programming language. Understands and can explain Moore's Law. Understands and can explain multitasking by computers.
Literacy	The quality of written work is limited; structure and style are simplistic. Work contains many significant errors of spelling, punctuation and grammar, which obscure meaning.	The quality of written work is basic and its meaning is often unclear; work has a basic structure but lacks fluency of style. Some elements of work are fit for purpose but work contains some significant errors of spelling, punctuation and grammar, which sometimes obscure meaning.	The quality of written work is generally sound and its meaning is usually clear; work has an appropriate structure and some attempt at a fluent style. Work is generally fit for purpose but contains minor errors of spelling, punctuation and grammar.		The quality of written work is good, has clear meaning and uses an appropriate structure and style. Work is fit for purpose; it contains a few errors of spelling, punctuation and grammar, but these do not obscure meaning.		The quality of written work is excellent, enhances meaning and uses a clear structure and fluent style. Work has good spelling, punctuation and grammar.	