Curriculum Overview Computing

Computing: essential skills and knowledge			
Year 1			
Sub-strand	Progression statement	What to look for guidance (Meeting expectations)	Relevant Switched on Computing unit(s)
Problem solving	C.1.1.1. Understand what algorithms are.	The pupil can understand algorithms as sequences of instructions in everyday contexts. The pupil can take real-world problems and then plan a sequence of steps to solve these. The problems could be moving a Blue-Bot from one point to another, or making some simple food items like a sandwich, smoothie or overnight oats. (E.g. In 1.1, recognise a set of directions as an algorithm. In 1.2, recognise the steps of a recipe as an algorithm.)	1.1: We are treasure hunters 1.2: We are TV chefs • We are TV chefs: Teacher notes • We are treasure hunters: Teacher notes
	C.1.1.2. Understand how algorithms are implemented as programs on digital devices; and that programs execute by following precise and unambiguous instructions.	The pupil can program floor turtles using sequences of instructions to implement an algorithm. The pupil can create a Blue-Bot (or similar) program using a number of steps in order before pressing the Go button. The length of the pupil's programs might increase over the year. (E.g. In 1.1, create a Blue-Bot program, implementing the complete algorithm for their solution.)	1.1: We are treasure hunters • We are treasure hunters: Teacher notes
Programming	C.1.2.1. Create and debug simple programs.	The pupil can give a sequence of instructions to a floor turtle. The pupil can create a Blue-Bot program using a sequence of instructions before running it using the Go button. The length of the pupil's programs might be expected to increase over the course of the year. (E.g. In 1.1, give the Blue-Bot a complete program.)	1.1: We are treasure hunters • We are treasure hunters: Teacher notes
Logical thinking	C.1.3.1. Use logical reasoning to predict the behaviour of simple programs.	The pupil can give explanations for what they think a program will do. The pupil can explain to the teacher, and to peers, what they think a program will do. This could be a program they or their peers have written, or it could be a familiar piece of software (including computer games). The pupil could use an audio recorder or video camera to capture their explanations. (E.g. In 1.1, explain what their own or another pupil's program will do before it is run.)	1.1: We are treasure hunters • We are treasure hunters: Teacher notes
Year 2			
Sub-strand	Progression statement	What to look for guidance (Meeting expectations)	Relevant Switched on Computing unit(s)
Problem solving	C.2.1.1. Understand what algorithms are.	The pupil can understand algorithms as sequences of instructions or sets of rules in everyday contexts. The pupil can recognise that common sequences of instructions or sets of rules can be thought of as algorithms. Examples could include recipes, but might also be procedures or rules in class, spelling rules, simple arithmetic operations or number patterns.	2.1: We are astronauts 2.2: We are games testers 2.3: We are photographers • We are astronauts: Teacher notes

		(E.g. In 2.1, recognise sets of directions as algorithms. In 2.2, recognise that the rules of a game are an algorithm. In 2.3, think of the steps to taking and editing photographs as an algorithm.)	• We are games testers: Teacher notes • We are photographers: Teacher notes
	C.2.1.2. The pupil can understand how algorithms are implemented as programs on digital devices, and that programs execute by following precise and unambiguous instructions.	The pupil can program on screen using sequences of instructions to implement an algorithm. The pupil can create programs as sequences of instructions when programming on screen. Their program could be written using simple programming apps (such as ScratchJr), perhaps using pre-prepared blocks and sprites. (E.g. In 2.1, program sprites in ScratchJr to solve the problems given to them. In 2.2, recognise how the Scratch games implement sets of rules.)	2.1: We are astronauts 2.2: We are games testers • We are astronauts: Teacher notes • We are games testers: Teacher notes
Programming	C.2.2.1. Create and debug simple programs.	The pupil can create a simple program on screen, correcting any errors. The pupil can create a simple program on screen (e.g. using ScratchJr) with a particular goal or purpose in mind (e.g. moving a sprite from one place to another). The pupil can debug any errors in their own code. (E.g. In 2.1, create their own program for the rocket sprite in ScratchJr, correcting any errors.)	2.1: We are astronauts 2.2: We are games testers • We are astronauts: Teacher notes • We are games testers: Teacher notes
Logical thinking	C.2.3.1. Use logical reasoning to predict the behaviour of simple programs.	The pupil can give logical explanations for what they think a program will do. The pupil can give logical explanations of what a program will do under given circumstances, including some attempt at explaining why it does what it does. The program could be one they have written or it could be a computer game or a familiar piece of software. The pupil could use an audio recorder or a video camera to record their explanations. (E.g. In 2.1, give logical explanations for what their own or their peers' programs will do. In 2.2, give logical explanations for what happens in the games.)	2.1: We are astronauts 2.2: We are games testers • We are astronauts: Teacher notes • We are games testers: Teacher notes
Year 3			
Sub-strand	Progression statement	What to look for guidance (Meeting expectations)	Relevant Switched on Computing unit(s)
Problem solving	C.3.1.1. Design, write and debug programs that accomplish specific goals.	The pupil can design and write a program using a block language, without user interaction. A typical program might be a scripted animation for a joke, part of a story, or linked to another area of the curriculum. Programs could use pre-built sprites or ones designed by the pupil. Expect programs to include movement and dialogue; they may also include sound effects and some use of costumes to allow for animated movement. There may be more than one sprite in the animation.	3.1: We are programmers 3.2: We are bug fixers • We are bug fixers: Teacher notes • We are programmers: Teacher notes
		The pupil can explore simulations of physical systems on screen.	3.1: We are programmers 3.2: We are bug fixers

	C.3.1.2. Controlling or simulating physical systems.	The pupil can experiment with some on-screen simulations of physical systems, perhaps linked to topics from other curriculum areas, e.g. a ball bouncing on a bat or a car moving around a track. Many computer games include elements of computer simulations. The pupil can discuss what they have learned from using the simulation.	• We are bug fixers: Teacher notes • We are programmers: Teacher notes
	C.3.1.3: Solve problems by decomposing them into smaller parts.	The pupil can plan a project. Working with the teacher and, perhaps, other pupils, the pupil can develop an outline plan for a project in computing, involving multiple steps and resources, e.g. creating an animation, filming a video or conducting a survey. In video work, the plan might include identifying a subject; storyboarding the video; sourcing media; recording video; filming; editing; exporting.	3.1: We are programmers • We are programmers: Teacher notes
Programming	C.3.2.1. Use sequence, selection and repetition in programs; work with variables.	The pupil can use sequence in programs. In on-screen programming, the pupil's program should include a sequence of commands or blocks in an appropriate order. A typical program could be a simple scripted animation, e.g. telling a joke, a story or explaining an idea taken from elsewhere on the curriculum. The pupil's program might include multiple sprites; instructions could include movement, on-screen text, sound and/or costume changes.	3.1: We are programmers 3.2: We are bug fixers • We are bug fixers: Teacher notes • We are programmers: Teacher notes
	C.3.2.2. Work with various forms of input and output.	The pupil can write a program to produce output on screen. The pupil can create a program that produces output on screen, such as moving sprites or displayed text, e.g. a simple animation program.	3.1: We are programmers 3.2: We are bug fixers • We are bug fixers: Teacher notes • We are programmers: Teacher notes
Logical thinking	C.3.3.1. Use logical reasoning to explain how some simple algorithms work.	The pupil can explain a simple, sequence-based algorithm in their own words. The pupil can give an explanation for a simple algorithm based on a sequence of instructions. The algorithm could be one of their own, or a simple one with which they have been provided. The algorithms could be recorded graphically, e.g. as a storyboard.	3.1: We are programmers 3.2: We are bug fixers • We are bug fixers: Teacher notes • We are programmers: Teacher notes
	C.3.3.2. Use logical reasoning to detect and correct errors in algorithms and programs.	The pupil can use logical reasoning to detect errors in programs. The pupil can give well-thought-through reasons for errors they find in programs. Typically, the pupil can find errors by reasoning logically about the program code, but they might also be able to use logical reasoning to identify errors in programs when they are executed. The programs do not have to be written originally by the pupil.	3.1: We are programmers 3.2: We are bug fixers • We are bug fixers: Teacher notes • We are programmers: Teacher notes
Wider understanding	C.3.3.3. Understand computer networks including the Internet.	The pupil can understand that computer networks transmit information in a digital (binary) format. The pupil can explain that any information has to be converted to numbers before it can travel through computer networks. The pupil should understand that this conversion happens according to an agreed system or code.	3.5: We are co-authors 3.6: We are opinion pollsters • We are opinion pollsters: Teacher notes

			• We are co-authors: Teacher notes
	C.3.4.1. Understand how networks can provide multiple services, such as the World Wide Web.	The pupil can understand that email and videoconferencing are made possible through the Internet. The pupil should know that email messages are sent and received through servers connected to the Internet. The pupil should know that other systems also work through the Internet, but these services may be direct, peer-to-peer connections rather than via servers.	3.5: We are co-authors 3.6: We are opinion pollsters • We are opinion pollsters: Teacher notes • We are co-authors: Teacher notes
Year 4			
Sub-strand	Progression statement	What to look for guidance (Meeting expectations)	Relevant Switched on Computing unit(s)
Problem solving	C.4.1.1. Design, write and debug programs that accomplish specific goals.	The pupil can design and write a program using a block language to a given brief, including simple interaction. The pupil can write a program in Scratch or MakeCode (or similar) in which the user has to provide some input, perhaps as an answer to a question on screen, or by using key presses or the mouse. The program could be a simple game or a set of questions and typed responses.	4.1: We are software developers 4.2: We are makers • We are software developers: Teacher notes • We are makers: Teacher notes
	C.4.1.2. Controlling or simulating physical systems.	The pupil can develop their own simulation of a simple physical system on screen.	4.2: We are makers • We are makers: Teacher notes
	C.4.1.3. Solve problems by decomposing them into smaller parts.	The pupil can work with others to plan a project. Given a particular project, the pupil can work as part of a team to plan how to accomplish their goal, breaking the project down into a set of tasks. Examples of projects could include creating an educational game or monitoring the weather.	4.1: We are software developers 4.2: We are makers 4.6: We are meteorologists • We are meteorologists: Teacher notes • We are software developers: Teacher notes • We are makers: Teacher notes
Programming	C.4.2.1. Use sequence, selection and repetition in programs; work with variables.	The pupil can use sequence and repetition in programs. The pupil's program, typically written in Scratch, or similar, should include sequences of commands or blocks and some repetition. Repetition would typically be for a fixed number of times, but might also include exit conditions (e.g. repeat...until...). Programs might include simple music or a simple game.	4.1: We are software developers 4.2: We are makers 4.3: We are musicians 4.5: We are artists • We are artists: Teacher notes

			• We are software developers: Teacher notes • We are makers: Teacher notes • We are musicians: Teacher notes
	C.4.2.2. Work with various forms of input and output.	The pupil can write a program that accepts keyboard input and produces on-screen output. In Scratch (or similar), the pupil can write a program that displays a question, accepts typed input and responds in an appropriate way to what is typed. This might be used as the basis for a dialogue program or a simple maths game.	4.1: We are software developers 4.2: We are makers 4.3: We are musicians 4.5: We are artists 4.6: We are meteorologists • We are artists: Teacher notes • We are meteorologists: Teacher notes • We are software developers: Teacher notes • We are makers: Teacher notes • We are musicians: Teacher notes
Logical thinking	C.4.3.1. Use logical reasoning to explain how some simple algorithms work.	The pupil can explain an algorithm using sequence and repetition in their own words. Given an algorithm using both sequence and repetition, the pupil can give a coherent, logically reasoned explanation of what it does and how it works. Repetition is likely to be 'forever' or for a set number of times, although end conditions (e.g. repeat...until...) could be used.	4.1: We are software developers 4.6: We are meteorologists • We are meteorologists: Teacher notes • We are software developers: Teacher notes
	C.4.3.2. Use logical reasoning to detect and correct errors in algorithms and programs.	The pupil can use logical reasoning to detect and correct errors in programs. The pupil can give well-thought-through reasons for errors they find in programs and explain how they have fixed these. The pupil can find and correct errors by reasoning logically about the program code; they might also be able to use logical reasoning to identify errors in programs when executed and confirm that they have fixed these by testing the new version of their program. The programs do not have to be written originally by the pupil.	4.1: We are software developers • We are software developers: Teacher notes
Wider understanding	C.4.3.3. Understand computer networks including the Internet.	The pupil can understand that the Internet transmits information as packets of data. When working online, the pupil can explain that the information they send and receive is automatically broken down into packets of data, and that these sometimes take different routes across the Internet.	4.4: We are bloggers • We are bloggers: Teacher notes
		The pupil can understand how the Internet makes the web possible.	4.4: We are bloggers

	C.4.4.1. Understand how networks can provide multiple services, such as the World Wide Web.	The pupil can give an explanation of how requests for web pages, and the HTML for those pages, are transmitted via the Internet.	• We are bloggers: Teacher notes
Year 5			
Sub-strand	Progression statement	What to look for guidance (Meeting expectations)	Relevant Switched on Computing unit(s)
Problem solving	C.5.1.1. Design, write and debug programs that accomplish specific goals.	The pupil can design, write and debug a program using a block language based on their own ideas. The pupil can design a program of their own and write this in a block-based language such as Scratch. The pupil can test and debug their code, explain what bugs they found and how they fixed them. The program need not be complex but it should be accomplished with a degree of independent working.	5.1: We are game developers 5.6: We are VR designers • We are VR designers: Teacher notes • We are game developers: Teacher notes
	C.5.1.2. Controlling or simulating physical systems.	The pupil can experiment with computer control applications.	5.1: We are game developers 5.6: We are VR designers • We are VR designers: Teacher notes • We are game developers: Teacher notes
	C.5.1.3. Solve problems by decomposing them into smaller parts.	The pupil can plan a solution to a problem using decomposition. The pupil can take a complex problem, identify component parts, use decomposition to break this problem down and then plan how they can solve the problem by working through the elements they have identified. Projects could include developing a computer game, creating a website or designing a building.	5.1: We are game developers 5.3: We are architects 5.6: We are VR designers • We are architects: Teacher notes • We are VR designers: Teacher notes • We are game developers: Teacher notes
Programming	C.5.2.1. Use sequence, selection, and repetition in programs; work with variables.	The pupil can use sequence, selection and repetition in programs. The pupil's program, typically written in Scratch, or similar, should include sequences of commands or blocks, some repetition and selection. Repetition might include exit conditions (e.g. repeat...until...). Selection would normally be of an if...then or if...then...else type. At this level, expect the pupil to be able to combine repetition with selection. Programs might include a computer game.	5.1: We are game developers 5.6: We are VR designers • We are VR designers: Teacher notes • We are game developers: Teacher notes
		The pupil can write a program that accepts keyboard and mouse input and produces output on screen and through speakers.	5.1: We are game developers 5.6: We are VR designers

	C.5.2.2. Work with various forms of input and output.	In Scratch (or similar), the pupil can create a computer game using the keyboard or mouse for input and the screen and speakers for output.	• We are VR designers: Teacher notes • We are game developers: Teacher notes
Logical thinking	C.5.3.1. Use logical reasoning to explain how some simple algorithms work.	The pupil can explain a rule-based algorithm in their own words. When provided with a rule-based algorithm (e.g. for a computer game), the pupil should be able to explain what it does and how it works, in their own words.	5.1: We are game developers 5.2: We are cryptographers • We are cryptographers: Teacher notes • We are game developers: Teacher notes
	C.5.3.2. Use logical reasoning to detect and correct errors in algorithms and programs.	The pupil can use logical reasoning to detect errors in algorithms. When given an algorithm for a particular purpose, e.g. a rule-based algorithm for a computer game or a sequence of steps to draw a geometric pattern, the pupil can use logical reasoning to identify possible errors in the algorithm, explaining why they believe the algorithm is incorrect.	5.1: We are game developers 5.2: We are cryptographers • We are cryptographers: Teacher notes • We are game developers: Teacher notes
Wider understanding	C.5.3.3. Understand computer networks including the Internet.	The pupil can understand how data routing works on the Internet. The pupil can give a coherent explanation of how data packets are routed from one computer to another on a separate network, which is also connected to the Internet.	5.2: We are cryptographers 5.4: We are web developers • We are cryptographers: Teacher notes • We are web developers: Teacher notes
	C.5.4.1. Understand how networks can provide multiple services, such as the World Wide Web.	The pupil can understand how web pages are created and transmitted. The pupil can explain how HTML is used to create a web page and how it is transmitted as packets of digital data over the Internet. The pupil should have an awareness of simple HTML tags for marking up a web page.	5.2: We are cryptographers 5.4: We are web developers • We are cryptographers: Teacher notes • We are web developers: Teacher notes
Year 6			
Sub-strand	Progression statement	What to look for guidance (Meeting expectations)	Relevant Switched on Computing unit(s)
Problem solving	C.6.1.1. Design, write and debug programs that accomplish specific goals.	The pupil can design, write and debug a program using a second programming language based on their own ideas. The pupil can design a program of their own and write this in a programming language other than Scratch (or whichever language has formed the focus for their programming in other years), such as MakeCode. The second language does not need to be text based, but Logo or Python could be used. The pupil can test and debug their code, explain what bugs they found and how they fixed these. The program need not be complex.	6.1: We are toy makers 6.2: We are computational thinkers 6.6: We are AI developers • We are AI developers: Teacher notes

			• We are computational thinkers: Teacher notes • We are toy makers: Teacher notes
	C.6.1.2. Controlling or simulating physical systems.	The pupil can design, write and debug their own computer control application.	6.1: We are toy makers 6.6: We are AI developers
			• We are AI developers: Teacher notes • We are toy makers: Teacher notes
	C.6.1.3. Solve problems by decomposing them into smaller parts.	The pupil can solve problems using decomposition, tackling each part separately. The pupil can take a complex problem, identify component parts, use decomposition to break this problem down and then plan how they can solve the problem by working through the elements they have identified. they can then use their plan to solve the original problem.	6.1: We are toy makers 6.6: We are AI developers
			• We are AI developers: Teacher notes • We are toy makers: Teacher notes
Programming	C.6.2.1. Use sequence, selection and repetition in programs; work with variables.	The pupil can use sequence, selection, repetition and variables in programs. The pupil's program should include sequences of commands or blocks, repetition, selection and variables. Repetition might include exit conditions (e.g. repeat...until...) and perhaps a counter-variable for iteration. Selection would normally be of an if...then or if...then...else type. At this level, expect the pupil to be able to combine repetition with selection and variables.	6.1: We are toy makers 6.2: We are computational thinkers
			• We are computational thinkers: Teacher notes • We are toy makers: Teacher notes
	C.6.2.2. Work with various forms of input and output.	The pupil can write a program that accepts inputs other than keyboard and mouse and produces outputs other than screen or speakers.	6.1: We are toy makers 6.2: We are computational thinkers
			• We are computational thinkers: Teacher notes • We are toy makers: Teacher notes
Logical thinking	C.6.3.1. Use logical reasoning to explain how some simple algorithms work.	The pupil can give clear and precise logical explanations of a number of algorithms. Given an algorithm, the pupil can describe what it does and, using logical reasoning, give precise explanations of how it works. Algorithms could be linked to programming projects, but might include a key algorithm such as binary search.	6.1: We are toy makers 6.2: We are computational thinkers
			• We are computational thinkers: Teacher notes • We are toy makers: Teacher notes

	C.6.3.2. Use logical reasoning to detect and correct errors in algorithms and programs.	The pupil can use logical reasoning to detect and correct errors in algorithms (and programs). When given an algorithm for a particular purpose, e.g. a rule-based algorithm for a smartphone app, the pupil can use logical reasoning to identify possible errors in the algorithm, explaining why they believe the algorithm is incorrect. The pupil can use logical reasoning to suggest possible corrections to the algorithm, explaining why these would correct the bug they identified.	6.1: We are toy makers 6.2: We are computational thinkers • We are computational thinkers: Teacher notes • We are toy makers: Teacher notes
Wider understanding	C.6.3.3. Understand computer networks including the Internet.	The pupil can understand how mobile phone or other networks operate. The pupil can give an explanation of how networks operate: they should know that information is transmitted digitally, and have some understanding of the network topology involved.	6.3: We are publishers 6.4: We are connected • We are connected: Teacher notes • We are publishers: Teacher notes
	C.6.4.1. Understand how networks can provide multiple services, such as the World Wide Web.	The pupil can understand how domain names are converted into IP addresses on the Internet. The pupil can give some explanation of how a domain name is converted into an IP address using the distributed domain name system (DNS) using something similar to a set of phone books. The pupil should show an awareness of the looked-up addresses (DNS records) being copied (cached), and that more local records are used in preference to more authoritative records in most circumstances.	6.3: We are publishers 6.4: We are connected • We are connected: Teacher notes • We are publishers: Teacher notes