Computing Progression Framework: Computer Science

National Curriculum Objective		Key Stage 1		Lower Key Stage 2		Upper Key Stage 2	
		Year 1	Year 2	Year 3	Year 4	Year 5	Year 6
		Create and debug simple programs.		**Use sequence, selection and repetition in programs; work with variables**		**Use sequence, selection, and repetition in programs; work with variables**	
Programming	Expected Standard	**The child can give a sequence of instructions to a floor turtle:** • The child is able to create a Bee Bot program using a sequence of instructions • The child is able to use the Go button to run the program *NB: The length of their programs will be expected to increase over the course of the year.*	**The child can create a simple program on screen, correcting any errors.** • The child should be able to create a simple program on screen (e.g. using the Blue Bot app, Scratch Jr or with prepared sprites and blocks in Scratch) with a particular goal or purpose in mind (e.g. drawing a shape or moving a sprite from one place to another). • The child should be able to debug any errors in their own code.	**The child can use sequence in programs.** • In on-screen programming, the child's program should include a sequence of commands or blocks in an appropriate order. • The child's program might include multiple sprites; instructions could include movement, on-screen text, sound and/or costume changes. *NB: 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 child can use sequence and repetition in programs.** • The child'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 turtle graphics, simple music or a simple game.	**The child can use sequence, selection and repetition in programs.** • The child'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. *NB: At this level, expect the child to be able to combine repetition with selection. Programs might include a computer game or a turtle graphics design.*	**The child can use sequence, selection, repetition and variables in programs.** • The child'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. *NB: At this level, expect the child to be able to combine repetition with selection and variables. Programs might include a simple smartphone app.*
				Work with various forms of input and output		**Work with various forms of input and output**	
	Expected Standard			**The child can write a program to produce output on screen.** • The child can create a program that produces output on screen, such as moving sprites or displayed text, e.g. a simple animation program.	**The child can write a program that accepts keyboard input and produces on-screen output.** • In Scratch (or similar), the child 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.	**The child can write a program that accepts keyboard and mouse input and produces output on screen and through speakers.** • In Scratch (or similar), the child can create a computer game using the keyboard or mouse for input and the screen and speakers for output.	**The child can write a program that accepts inputs other than keyboard and mouse and produces outputs other than screen or speakers.** • The child could create a smartphone app, using the touch screen and the GPS sensor or accelerometer for input, and the screen and speakers or headphones plus vibration motor or network connection for output.

National Curriculum Objective		Key Stage 1		Lower Key Stage 2		Upper Key Stage 2	
		Year 1	Year 2	Year 3	Year 4	Year 5	Year 6
		Use logical reasoning to predict the behaviour of simple programs.		**Use logical reasoning to explain how some simple algorithms work.**		**Use logical reasoning to explain how some simple algorithms work.**	
Logical Thinking	Expected Standard	**The child can give explanations for what they think a program will do:** • The child can explain to the teacher, and to their 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 child use an audio recorder or video camera to capture their explanations.	**The child can give logical explanations for what they think a program will do.** • The child 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 themselves have written or it could be a computer game or a familiar piece of software. • The child could use an audio recorder or a video camera to record their explanations.	**The child can explain a simple, sequence based algorithm in their own words.** • The child 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.	**The child can explain an algorithm using sequence and repetition in their own words.** • Given an algorithm using both sequence and repetition, the child 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.	**The child can explain a rule-based algorithm in their own words.** • When provided with a rule-based algorithm (e.g. for a computer game), the child should be able to explain what it does and how it works, in their own words.	**The child can give clear and precise logical explanations of a number of algorithms.** • Given an algorithm, the child 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.
				Use logical reasoning to detect and correct errors in algorithms and programs.		**Use logical reasoning to detect and correct errors in algorithms and programs.**	
	Expected Standard			**The child can use logical reasoning to detect errors in programs.** • The child can give well-thought-through reasons for errors they find in programs. Typically, the child 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	**The child can use logical reasoning to detect and correct errors in programs.** • The child can give well-thought-through reasons for errors they find in programs and explain how they have fixed these. The child 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 child	**The child 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 child can use logical reasoning to identify possible errors in the algorithm, explaining why they believe the algorithm is incorrect.	**The child 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 child can use logical reasoning to identify possible errors in the algorithm, explaining why they believe the algorithm is incorrect. • The child can use logical reasoning to suggest possible corrections to the algorithm, explaining why these would correct the bug they identified.
				Understand computer networks including the internet		**Understand computer networks including the internet**	
Logical Reasoning	Expected Standard			**The child can understand that computer networks transmit information in a digital (binary) format.** • The child can explain that any information has to be converted to numbers before it can travel through computer networks. The child should understand that this conversion happens according to an agreed system or code.	**The child can understand that the internet transmits information as packets of data.** • When working online, the child 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.	**The child can understand how data routing works on the internet**. • The child 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.	**The child can understand how mobile phone or other networks operate.** • The child can give an explanation of how mobile phone (or other) networks operate: they should know that information is transmitted digitally, and have some understanding of the network topology involved. In the case of mobile phone networks, the child should show some understanding of the interactions between a phone, cell transmitters/receivers and the network's control systems.
				Understand how networks can provide multiple services, such as the world wide web.		**Understand how networks can provide multiple services, such as the world wide web.**	
	Expected Standard			**The child can understand that email and videoconferencing are made possible through the internet.** • The child should know that email messages are sent and received through servers connected to the internet. The child should know that Skype and other videoconferencing systems also work through the internet, but these services may be direct, peer-to-peer connections rather than via servers.	**The child can understand how the internet makes the web possible.** • The child can give an explanation of how requests for web pages, and the HTML for those pages, are transmitted via the internet	**The child can understand how web pages are created and transmitted.** • The child 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 child should have an awareness of simple HTML tags (such as <h1> and <p>) for marking up a web page.	**The child can understand how domain names are converted into IP addresses on the internet.** • The child 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 child 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.

National Curriculum Objective		Key Stage 1		Lower Key Stage 2		Upper Key Stage 2	
		Year 1	Year 2	Year 3	Year 4	Year 5	Year 6
		Understand what algorithms are.		**Design, write and debug programs that accomplish specific goals.**		**Design, write and debug programs that accomplish specific goals**	
Problem Solving	Expected Standard	**The child understands algorithms as sequences of instructions in everyday contexts:** • The child can take real-world problems and then plan a sequence of steps to solve these (problems could be moving a Bee Bot from one point to another, or making some simple food items like a sandwich, smoothie or pizza)	**The child can understand algorithms as sequences of instructions or sets of rules in everyday contexts.** • The child 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.	**The child 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 child. 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.	**The child can design and write a program using a block language to a given brief, including simple interaction.** • The child can write a program in Scratch (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.	**The child can design, write and debug a program using a block language based on their own ideas.** • The child can design a program of their own and write this in a block-based language such as Scratch. The child can test and debug their code, explain what bugs they found and how they fixed them. The program need not be complex (a simple game or a turtle graphics program would suffice) but it should be accomplished with a degree of independent working.	**The child can design, write and debug a program using a second programming language based on their own ideas.** • The child 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 TouchDevelop or App Inventor. The second language does not need to be text based, but Logo or Python could be used. The child can test and debug their code, explain what bugs they found and how they fixed these. The program need not be complex - a simple app would suffice.
		Understand how algorithms are implemented as programs on digital devices; and that programs execute by following precise and unambiguous instructions.		**Controlling or simulating physical systems.**		**Controlling or simulating physical systems**	
	Expected Standard	**The child can program floor turtles using sequences of instructions to implement an algorithm:** • The child can create a Bee Bot (or similar) program using a number of steps in order before pressing the Go button. *NB: The length of their programs should increase over the year.*	**The child can program on screen using sequences of instructions to implement an algorithm.** • The child can create programs as sequences of instructions when programming on screen. Their program could be written using simple programming apps (such as Blue Bot or Lightbot), Scratch Jr or Scratch, perhaps using pre-prepared blocks and sprites in this case.	**The child can explore simulations of physical systems on screen.** • The child 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 child can discuss what they have learned from using the simulation.	**The child can develop their own simulation of a simple physical system on screen.** • The child can create a Scratch (or similar) program to simulate a simple physical system. This could be in the form of a simple animation or an on-screen prototype for a product made in design and technology.	**The child can experiment with computer control applications.** • The child can use simple computer control and/or sensors with products they make in design and technology, perhaps using Lego WeDo kits, MaKey MaKey or similar.	**The child can design, write and debug their own computer control application.** • The child can add computer control and/or sensors to a smartphone app or to products they design and make in design and technology, perhaps using Lego WeDo kits, MaKey MaKey or similar. The child can show evidence of designing, writing and debugging their program, ensuring that this functions correctly on the available hardware platform.
Problem Solving				**Solve problems by decomposing them into smaller parts**		**Solve problems by decomposing them into smaller parts**	
	Expected Standard			**Working with the teacher and, perhaps, other children, the child can develop an outline plan for a project in computing, involving multiple steps and resources.** • The child could create an animation, film a video or conduct a survey. In video work, the plan might include identifying a subject; storyboarding the video; sourcing media; recording video; filming; editing; exporting.	**The child can work with others to plan a project.** • Given a particular project, the child 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, developing a wiki or monitoring the weather.	**The child can plan a solution to a problem using decomposition.** • The child 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.	**The child can solve problems using decomposition, tackling each part separately.** • The child 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. Projects can be extended, such as developing a smartphone app.