Algorithm	Algorithm (programming) Evaluation	Decomposition	Generalisation	Pattern Spotting	Abstraction	Design	Sequence	Selection	Repetition	Variable Use	Input Output	Debugging
Think of a simple everyday algorithm (Remove toothpaste top, squeeze toothpaste onto brush)	Identify if an algorithm does what you want it to do (Does Bee-bot reach its target?, Is Scratch conversation easy to read?)	Break a simple everyday algorithm into parts (breakfast, getting changed, walk to school)	Recognise where an idea is adapted and used again as directed by the teacher (Perimeter)	Spot simple patterns in code or algorithm (simple regular 2d logo shapes)	Decide which parts of a project are most important or should be started first (Magic Carpet)	Following instructions to create	Follow simple everyday sequences of instructions	use single simple condition (playground games flowchart decisions)	Explore and use simple repetition in music and dance	variable used to hold number or word and reported (Quiz, Counting machine)	simple inputs (keys, mouse click, switch etc) control on or off (Lego Wedo toilet fan, car park barrier, Traffic Lights)	Recognise that there is a problem say what problem is
read and follow symbol sequence algorithm (PE Cards, jump, step etc)	Recognise that there are more than one algorithm to do the same thing (Identify two bee-bot routes to go from A-B)	Observe a working program and decompose its elements as a class (Smoking car, slug trail, Magic Carpet, Europe)	Pupil chooses to Adapt ideas that they have used to solve similar problems (Selection investigation, primary games maker, Random word)	Spot patterns in algorithm or code and continue the patterns (Coin machine, program a clock)	Define all the elements in something and then remove the ones that are not needed (Music as code)	Add small non critical adaptations (Smoking car, Music machine, dressing up game)	Create simple everyday sequences of instructions	create selection within a loop (Slug maze, selection investigation)	Create simple repeat x times loop (Scratch Jr Dance, scratch music notes in repeat loop, repeat x in logo)	multiple non connected variables used	Changing state other than on off such as fast slow bright dim etc (Lego Wedo toilet fan, car park barrier)	identify where in the code or algorithm bug/ problem occurs
Create simple sequence algorithms using symbols (Bee-bot cards, human crane cards)	Recognise that one algorithm may be better (Bee-bot shorter distance, less time. Less algorithm symbols)	Observe a working program and decompose its elements as an individual (Tables Game)				Adapt a given design for a new teacher given purpose (Times tables game)	create sequence of symbols (Bee-bot, human crane, Scratch Junior Travel,	Use single maths operator condition (Maths quiz)	Create non terminating continuous (forever) loop (making sprite move slug maze, crab maze)	variables that change inside a loop (Counting machine)	Use sensors to control or report (Lego Wedo tilt switch)	Debug simple sequence errors independently
Read and follow written sequence algorithms (Forward 3, right 90)	Design an algorithm/code for a specific person or group of people (design a Scratch program for younger pupils)	Create a program by decomposing it into parts and solving parts separately (Primary games maker)				Repurposing ideas for a pupil chosen purpose (Selection investigation, Primary games maker)	create sequence of simple code that can be easily read (Scratch Junior blocks, say or think blocks in Scratch)	Multiple selection beyond if and else (Scratch use of multiple ifs perimeter planning, Lego Wedo Toilet Fan)	loops within loops for a reason (Clock)	variables interacting with other variables (train computer to do maths, Tickle Version)		Debug simple repetition, selection & variable errors independently
write simple sequence algorithms using words (Jam sandwich algorithm) read and decode musical algorithms Music Score() use rules algorithms	Evaluate more complex code that does the same thing (Logo design with procedures and without, Scratch code with loops and without)						create multiple sequences running concurrently (Scratch music machine notes and beat in separate blocks)	Multiple conditions using AND OR NOT or more advanced maths (odd and even, Algebra Inverse)	Create and use loops that terminates when condition met (repeat until loop in coin program)			Debug repetition, selection & variable errors independently
Read and follow algorithms with selection and repetition (Playground algorithms, counting machine)							create sequences or multiple sequences where timing is critical (control sprite on times tables game, conversation sequence)					Dividing up code to find where the error is or running Scratch sets of blocks separately
Complete unfinished algorithms with selection and or repetition (Coins sorter, Program a clock)		**COLOUR KEY**	MAINLY KS1 ONLY	KS1 & KS2	MAINLY KS2 ONLY	KS2 & KS3						
Create an algorithm with selection or repetition									code-it.co.uk		code-it.co.uk	