

| 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 tooth-paste top, squeeze tooth-paste onto brush)   | Identify if an algorithm does what you want it to do (Does <a href="#">Bee-bot</a> reach its target?, Is <a href="#">Scratch conversation</a> 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 ( <a href="#">Perimeter</a> )   | Spot simple patterns in code or algorithm (simple regular 2d logo shapes)   | Define all the elements in something and then remove the ones that are not needed ( <a href="#">Music as code</a> ) | Following instructions to create ( <a href="#">Smoking car</a> , <a href="#">Music machine</a> , <a href="#">dressing up game</a> )   | Follow simple everyday sequences of instructions   | use single simple condition ( <a href="#">playground games</a> flowchart decisions)                     | Explore and use simple repetition in music and dance   | variable used to hold number or word and reported ( <a href="#">Quiz</a> , <a href="#">Counting machine</a> ) | simple inputs (keys, mouse click, switch etc) control on or off (Lego Wedo <a href="#">toilet fan</a> , <a href="#">car park barrier</a> )   | 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 <a href="#">bee-bot</a> routes to go from A-B)                           | Observe a working program and decompose its elements as a class ( <a href="#">Smoking car</a> , <a href="#">slug maze</a> ) | Pupil chooses to Adapt ideas that they have used to solve similar problems ( <a href="#">Selection investigation</a> , <a href="#">primary games maker</a> , <a href="#">Random word</a> ) | Spot patterns in algorithm or code and continue the patterns ( <a href="#">Coin machine</a> , <a href="#">program a clock</a> ) |   | Add small non critical adaptations ( <a href="#">Smoking car</a> , <a href="#">Music machine</a> , <a href="#">dressing up game</a> ) | Create simple everyday sequences of instructions   | create selection within a loop ( <a href="#">Slug maze</a> , <a href="#">selection investigation</a> )  | Create simple repeat x times loop (Scratch Jr, 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 <a href="#">toilet fan</a> , <a href="#">car park barrier</a> ) | identify where in the code or algorithm bug/ problem occurs                              |
| Create simple sequence algorithms using symbols ( <a href="#">Bee-bot</a> cards, <a href="#">human crane</a> cards)   | Recognise that one algorithm may be better ( <a href="#">Bee-bot</a> shorter distance, less time. Less algorithm symbols)                                    | Observe a working program and decompose its elements as an individual ( <a href="#">Tables Game</a> )                       |  |   |   | Adapt a given design for a new teacher given purpose ( <a href="#">Times tables game</a> )  | create sequence of symbols ( <a href="#">Bee-bot</a> cards, <a href="#">human crane</a> cards)   | Use single maths operator condition ( <a href="#">Maths quiz</a> )                                      | Create non terminating continuous (forever) loop (making sprite move <a href="#">slug maze</a> , <a href="#">crab maze</a> ) | variables that change inside a loop ( <a href="#">Counting machine</a> )                                      | Use sensors to control or report (Lego Wedo <a href="#">tilt switch</a> )  | 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 ( <a href="#">Primary games maker</a> )          |  |   |   | Repurposing ideas for a pupil chosen purpose ( <a href="#">Selection investigation</a> , <a href="#">Primary games maker</a> )        | 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 <a href="#">perimeter planning</a> ) | loops within loops for a reason  | variables interacting with other variables ( <a href="#">train computer to do maths</a> )                     |  | Debug simple repetition, selection & variable errors independently                       |
| write simple sequence algorithms using words ( <a href="#">Jam sandwich algorithm</a> ) read and decode musical algorithms ( <a href="#">Music Score</a> ) 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 ( <a href="#">Scratch music machine</a> notes and beat in separate blocks)                                      | Multiple conditions using AND OR NOT ( <a href="#">odd and even</a> )                                   | Create and use loops that terminates when condition met (repeat until loop in <a href="#">coin program</a> )                 |   |  | Debug repetition, selection & variable errors independently                              |
| Read and follow algorithms with selection and repetition ( <a href="#">Playground algorithms</a> , <a href="#">number machine</a> )   |  |   |  |   |   |   | create sequences or multiple sequences where timing is critical (control sprite on <a href="#">times tables game</a> , <a href="#">conversation sequence</a> ) |   |  |   |  | Dividing up code to find where the error is or running Scratch sets of blocks separately |
| Complete unfinished algorithms with selection and or repetition ( <a href="#">Coins sorter</a> , <a href="#">Program a clock</a> )  |  | <b>COLOUR KEY</b>   | <b>MAINLY KS1 ONLY</b>   | <b>KS1 &amp; KS2</b>  | <b>MAINLY KS2 ONLY</b>  | <b>KS2 &amp; KS3</b>  |  |   |  |   |  |  |
| Create an algorithm with selection or repetition  |  |   |  |   |   |   |  |   | <a href="#">code-it.co.uk</a>  |   |  |  |