|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Area | Foundation Stage | Year 1 | Year 2 | Year 3 | Year 4 | Year 5 | Year 6 |
| **Computer Science** | * Be able to give a floor robot instructions to make it move.
* Use simple software and explain what you are doing.
* Understand what happens when you click a button or touch an icon.
 | * Give instructions to a friend and follow their instructions to move around a space.
* Describe what happens when buttons are pressed on a robot or device.
* Press buttons in the correct order to make a robot follow a short sequence.
* Understand what an algorithm is and be able to create a simple algorithm.
* Understand and explain how algorithms are used in every day life.
* Begin to predict what will happen for a short sequence of instructions.
* Begin to use different software or applications to create movement and patterns on a screen.
* Use the word debug to correct an algorithm that doesn’t work in the way it was intended.
 | * Understand what an algorithm is and demonstrate simple linear algorithms.
* Be able to explain the order needed to do things to make something happen and to talk about it as an algorithm.
* Programme a robot or software to do a particular task.
* Look at a basic program and explain what will happen.
* Use programming software and applications to make objects move.
* Use logical reasoning to predict and debug more complex programs.
* Can create and debug with improved confidence & efficiency.
* Begin to program using simple block code.
 | * Understand how an algorithm is implemented using a sequence of precise instructions.
* Can predict the outcome of a sequence of precise instructions.
* Repeatedly test a program and recognise when they need to debug it.
* Detect a problem in an algorithm, which could result in a different outcome to the one intended.
* Understand what inputs and outputs are, how they can be used.
* Provide examples of how to use inputs and outputs effectively.
* Designs, writes, executes and debugs programs of increasing complexity that accomplish a specific goal.
* Use logical reasoning to predict and debug more complex programs including inputs and outputs.
 | * Design simple algorithms using loops and repeats, whilst detecting and correcting errors is debugging.
* Write and execute an efficient program, using loops such as forever, repeat & repeat until commands.
* Decompose a problem into smaller parts with some verbal reasoning.
* Has an understanding of how sequencing, using inputs and repetition in programs has specific effects on the output, works with ‘loops’ and understands their effect.
* Recognise that an algorithm will help to sequence more complex programs.
* Use logical reasoning to predict and debug more complex programs including loops and repeats.
 | * Program a condition that uses a sensor to detect a change, which can select an action within a program.
* Decomposes more open-ended problems into smaller parts, provides some reasoning for their choices.
* Approaches a range of problems using computationally thinking concepts, helping them to design other algorithms for other specific outcomes.
* Design, write and execute an efficient program, including selection (IF…THEN) command.
* Change an input to a program to achieve a different output.
* Use logical reasoning to predict and debug more complex programs including selection.
* Uses programs linked to physical systems and sensors e.g. the alarm goes off when the sensor is triggered.
* Design, write and execute an efficient program, which demonstrates and understanding of the difference between, and appropriate use of IF…THEN, IF…THEN…ELSE, and nested IF statements.
 | * Understand the importance of planning, testing and correcting algorithms.
* Demonstrate a range of different strategies to solve a problem including: abstraction, decomposition, logic & evaluation.
* Understand why sequence & patterns are important when creating simple algorithms that are part of a more complex program.
* Gives reasoning for each step within algorithms and applying them to a program.
* Understand & develop complex flow diagrams.
* Use a variable to increase programming possibilities.
* Use a variable and relational operators (e.g. < = >) within a loop to stop a program.
* Evaluate the effectiveness and efficiency of an algorithm while continually testing the programming of that program.
* Use different inputs (including sensors) to control a device or onscreen action and predict what will happen.
* Use logical reasoning to predict and debug more complex programs including: selection, variables and operators.
 |