HIGH LITTLETON CHURCH OF ENGLAND PRIMARY SCHOOL
COMPUTING MEDIUM TERM PLAN TERM 3
2023-2024

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
Hedgehog (Y1)	**Buttons** Learners will be introduced to floor robots. They will talk about what the buttons on a floor robot might do and then try the buttons out. They will spend time linking an outcome to a button press. Learners will consider the direction command buttons, as well as the ‘clear memory’ and ‘run program’ buttons.	**Directions** Learners will think about the language used to give directions and how precise it needs to be. They will also work with a partner to give and follow instructions. These real-world activities should, at suitable points during this lesson, be related to the floor robot introduced in Lesson 1.	**Forwards and backwards** Learners will focus on programming the floor robot to move forwards and backwards. They will see that the robot moves forwards and backwards a fixed distance. This highlights the idea that robots follow a clear, fixed command in a precise and repeatable way. Learners will think about starting the robot from the same place each time. Using the same starting position with fixed commands will allow learners to predict what a program will do.	**Four directions** Learners will focus on programming the floor robot to move forwards and backwards. They will see that the robot moves forwards and backwards a fixed distance. This highlights the idea that robots follow a clear, fixed command in a precise and repeatable way. Learners will think about starting the robot from the same place each time. Using the same starting position with fixed commands will allow learners to predict what a program will do.	**Getting there** Learners will use ‘left turn’ and ‘right turn’ commands along with ‘forwards’ and ‘backwards’ commands. Doing this will allow learners to develop slightly more complex programs. Learners will create their programs in this lesson through trial and error, before moving on to planning out their programs in Lesson 5. In Activity 3, learners will predict where given programs will move the robot to. Learners will make their predictions by looking at the commands and matching the	**Routes** Learners will be encouraged to plan routes around a mat before they start to write programs for those routes. The activities in this lesson also introduce the concept of there being more than one way to solve a problem. This concept is valid for a lot of programming activities: the same outcome can be achieved through a number of different approaches, and there is not necessarily a ‘right’ approach. The lesson also introduces the idea of program	**POP task**

					program steps to movements.	design, where learners need to plan what they want their program to achieve before they start programming.	
Fox (Y2)	**Giving instructions** Learners will follow instructions given to them and give instructions to others. They will consider the language used to give instructions, and how that language needs to be clear and precise. Learners will combine several instructions into a sequence that can then be issued to another learner to complete. They will then consider a clear and precise set of instructions in relation to an algorithm, and will think about how computers can only follow clear and unambiguous instructions.	**Same but different** Learners will focus on sequences, and consider the importance of the order of instructions within a sequence. They will create sequences using the same instructions in different orders. They will then test these sequences to see how the different orders affect the outcome.	**Making predictions** Learners will use logical reasoning to make predictions. They will follow a program step by step and identify what the outcome will be.	**Mats and routes** Learners will design, create, and test a mat for a floor robot. This will introduce the idea that design in programming not only includes code and algorithms, but also artefacts related to the project, such as artwork.	**Algorithm design** Learners will design an algorithm to move their robot around the mat that they designed in Lesson 4. As part of the design process, learners will outline what their task is by identifying the starting and finishing points of a route. This outlining will ensure that learners clearly understand what they want their program to achieve.	**Break it down** Learners will take on a larger programming task. They will break the task into chunks and create algorithms for each chunk. This process is known as ‘decomposition’ and is covered further in key stage 2. Learners will also find and fix errors in their algorithms and programs. They will understand this process to be ‘debugging’.	**POP task**
Badger (Y3)	**Introduction to Scratch** Learners are introduced to a new programming environment: Scratch.	**Programming sprites** Learners will create movement for more than one sprite. In doing this, they will	**Sequences** Learners will be introduced to the concept of sequences by joining blocks of	**Ordering commands** Learners have the opportunity to experiment with sequences where	**Looking good** Learners develop an understanding of sequences by giving them the opportunity to	**Making an instrument** Learners will create a musical instrument in Scratch. They	**POP task**

	Learners will begin by comparing Scratch to other programming environments they may have experienced, before familiarising themselves with the basic layout of the screen.	design and implement their code, and then will create code to replicate a given outcome. Finally, they will experiment with new motion blocks.	code together. They will also learn how event blocks can be used to start a project in a variety of different ways. In doing this, they will apply principles of design to plan and create a project.	order is and is not important. They will create their own sequences from given designs.	combine motion and sounds in one sequence. They will also learn how to use costumes to change the appearance of a sprite, and backdrops to change the appearance of the stage. They will apply the skills in Activity 1 and 2 to design and create their own project, including sequences, sprites with costumes, and multiple backdrops.	will apply the concept of design to help develop programs and use programming blocks — which they have been introduced to throughout the unit. They will learn that code can be copied from one sprite to another, and that projects should be tested to see if they perform as expected.	
Otter (Y4)	**Programming a screen turtle** This lesson will introduce pupils to programming in Logo. Logo is a text-based programming language where pupils type commands that are then drawn on screen. Pupils will learn the basic Logo commands, and will use their knowledge of them to read and write code.	**Programming letters** In this lesson, pupils will create algorithms (a precise set of ordered instructions, which can be turned into code) for their initials. They will then implement these algorithms by writing them in Logo commands to draw the letter. They will debug their code by finding and fixing	**Patterns and repeats** In this lesson, pupils will first look at examples of patterns in everyday life. They will recognise where numbers, shapes, and symbols are repeated, and how many times repeats occur. They will create algorithms for drawing a square,	**Using loops to create shapes** In this lesson, pupils will work with count-controlled loops in a range of contexts. First, they will think about a real-life example, then they will move on to using count-controlled loops in regular 2D shapes. They will trace code to	**Breaking things down** In this lesson, pupils will focus on decomposition. They will break down everyday tasks into smaller parts and think about how code snippets can be broken down to make them easier to plan and work with. They will learn to create,	**Creating a program** In the final lesson, pupils will apply the skills that they have learnt in this unit to create a program containing a count-controlled loop. Over the course of the lesson, they will design wrapping paper using more than one	**POP task**

		any errors that they spot.	using the same annotated diagram as in Lesson 2. They will use this algorithm to program a square the ‘long’ way, and recognise the repeated pattern within a square. Once they know the repeated pattern, they will use the repeat command within Logo to program squares the ‘short’ way.	predict which shapes will be drawn, and they will modify existing code by changing values within the code snippet.	name, and call procedures in Logo, which are code snippets that can be reused in their programming.	shape, which they will create with a program that uses count-controlled loops. They will begin by creating the algorithm, either as an annotated sketch, or as a sketch and algorithm, and then implement it as code. They will debug their work throughout, and evaluate their programs against the original brief.	
Robin (Y5)	**Connecting Crumbles** In this lesson, your learners will become familiar with the Crumble controller and the programming environment used to control it. Learners will connect a Sparkle to a Crumble and then program the Crumble to make the Sparkle flash different colour patterns. Learners will also use infinite loops, which were introduced to the learners in the previous school year.	**Combining output components** In this lesson, learners will connect a Sparkle and a motor to the Crumble controller. Learners will design sequences of actions for these components. They will then apply their understanding of repetition by using count-controlled loops when implementing their design as a program.	**Controlling with conditions** In this lesson, learners will be introduced to conditions, and how they can be used in programs to control their flow. They will identify conditions in statements, stating if they are true or false. Learners will be introduced to a Crumble switch, and learn how it can provide the	**Starting with selection** In this lesson, learners will develop their understanding of how the flow of actions in algorithms and programs can be controlled by conditions. They will be introduced to selection and then represent conditions and actions using the ‘if…then…’ structure. Learners	**Drawing designs** In this lesson, learners will apply their understanding of microcontrollers and selection when designing a project to meet the requirements of a given task. To support their understanding, learners will identify how selection might be used in real-world situations, then they will consider	**Writing and testing algorithms** In this final lesson of the unit, learners will develop Crumble programs to control the model of a fairground ride they built in Lesson 5. First, learners will identify how they are going to use selection before writing an algorithm to	**POP task**

			Crumble controller with an input that can be used as a condition. They will explore how to write programs that use an input as a condition.	will create algorithms that include selection. They will use their algorithms to guide their program writing. Learners will see that infinite repetition is required to repeatedly check if a condition has been met.	how they can apply this knowledge to design their project. Learners will produce design sketches to show how their model will be made and how they will connect the microcontroller to its components.	meet the requirements of the given task. They will then implement their algorithms as code. Learners will run their programs to identify any bugs, and then return to the code or algorithm to debug it where necessary. Finally, to conclude the unit, learners will evaluate their designs.	
Deer (Y6)	**Introducing variables** Learners are introduced to variables. They see examples of real-world variables (score and time in a football match) before they explore them in a Scratch project. Learners then design and make their own project that includes variables. Finally, learners identify that variables are named and that they can be letters (strings) as well as numbers.	**Variables in programming** Learners understand that variables are used in programs, and that they can only hold a single value at a time. They complete an unplugged task that demonstrates the process of changing variables. Then, learners explore why it is important to name variables and apply their learning in a Scratch project in which they make,	**Improving a game** Learners apply the concept of variables to enhance an existing game in Scratch. They predict the outcome of changing the same change score block in different parts of a program, then they test their predictions in Scratch. Learners also experiment	**Designing a game** Learners work at the 'design' level of abstraction, where they create their artwork and algorithms. Learners first design the sprites and backgrounds for their project, then they design their algorithms to create their program flow.	**Design to code** Learners implement the algorithms that they created in Lesson 4. In doing this, they identify variables in an unfamiliar project and learn the importance of naming variables. They also have the opportunity to add another variable to enhance their project.	**Improving and sharing** Learners build on the project that they created in Lesson 5. They consider how they could improve their own projects and make small changes to achieve this. Learners then have the opportunity to add a variable independently. Finally, learners	**POP task**

		name, and update variables.	with using different values in variables, and with using a variable elsewhere in a program. Finally, they add comments to their project to explain how they have met the objectives of the lesson.			evaluate each other's projects; they identify features that they liked and features that could be improved.	