YEAR 1 COMPUTING PLANNING - ALGORITHMS AND PROGRAMMING

Class: **Term: Autumn 1** **Subject: Computing** **Topic: Algorithms & Programming (making things move through instructions)**

Differentiation and support	Cross curricular links
SEN / EAL: Simplify tasks to focus on input and running a sequence. Provide children with pre-set activities for them to program. Work in mixed ability pairs to support learning. Symbol cards will help make links to activities. **GT:** require additional, detailed information and provide greater opportunities for making predictions. Online activities can be made more complex. Research independently to apply skills in different situations. Support less able peers by assisting them in debugging programs.	**English:** Instructional writing, using instructional language. **Maths:** Directional language and position, angles and turns. **Science:** How technology and inventors help us in the wider world. **Geography:** Using and being familiar with maps and grids. **PSHCE:** Collaborative problem solving, working with others to solve a task.

Key vocabulary for unit	Curriculum objectives covered in unit:
Algorithm: An algorithm is a sequence of instructions and/or set of rules. **Sequence**: A set of actions or events that must be carried out in the same order every time. **Debug:** Finding where the sequence failed and predicting and solving the sequence, so that it works/achieves the desired output. **Inputs:** These are the means of communicating with computers e.g. keyboard and mouse **Outputs:** These are the means by which the computer relays information e.g. printer or monitor **Code:** The written symbol/number/word created for the device to move or do an action. **Sprite:** A computer graphic or object that can be moved through an algorithm. **Program (noun):** Code for a given task **Program (verb):** Input the code for a given task	• Understand what algorithms are; how they are implemented as programs on digital devices; and that programs execute by following precise and unambiguous instructions. • Create and debug simple programs • Use logical reasoning to predict the behaviour of simple programs • Recognise common uses of information technology beyond school.

Overview

- In this unit, children will become familiar with instructions and commands to make an object move. This involves creating a sequence or series of step-by-step instructions to move an object or to reach a goal. In formal computing language, remind children that this is called an algorithm.
- Children will discover (either through predicting and testing or through trial and error) that sometimes the sequence does not work as intended. Here the children will need to evaluate what has happened in the sequence and how it might be resolved. In this instance, the children are able to 'debug' their programmes and improve them to achieve the correct outcome or goal.
- This unit requires the children to 'have a go' and find out what might happen when they give an instruction to a device. Children will learn and build on previous experiences and outcomes to predict and make informed decisions before they create a sequence.
- **Please note; children may move through this unit at different paces depending on their existing experiences with computing devices. Allow for children to develop and consolidate their learning week by week, before moving them on to the next activity or lesson. Encourage children to discuss, share ideas and use logical reasoning, based on predictions and or past experiences, to debug their programmes**.

Unit Overview

Lesson 1: Creating sequences using words (unplugged)

Lesson 2: Creating sequences using code (unplugged)

Lesson 3: Programming a device

Lesson 4: Programming a sprite (one instruction at a time)

Lesson 5: Programming a sprite (several instructions at a time)

Lesson 6: Inputting programmes and debugging programmes

Need to organise at the start of the unit

Lessons 1 and 2: Book the hall

Lesson 2: Send letters home in advance for children to bring in programmable devices if would like them to do this

All Lessons: Have the display cards for the vocabulary that has been covered on display throughout lessons where children can see them e.g. under the IWB

You can access the complete Year 1 Computing Planning, and all of the resources needed to teach each lesson, at:

https://www.saveteacherssundays.com/computing/year-1/721/

W	LO	Activities	Resources	Success Criteria
1	To understand what a sequence is To create and follow sequences of instructions (45 mins)	This lesson will look at exploring the key concepts of instructional and directional language and involves creating 'unplugged' sequences Children will also begin to explore how to 'debug' their sequences, if the program doesn't achieve the correct objective or outcome **Intro:** Ask the children to think of some things that they always do in the same order each day e.g. the steps in brushing their teeth Explain that these are examples of a 'sequence', emphasising that the steps in a sequence always have to be followed *in the same order* each time the sequence is completed With the children's help, write the steps in some of the best examples of sequences that need to be completed in a certain order e.g. getting dressed, having a shower etc Exemplify how a sequence must be followed in the same order every time by using the example of brushing your teeth e.g. you always have to pick up the toothbrush before you can put toothpaste on it – you cannot put toothpaste on the toothbrush and then pick it up Discuss with children how all of the things on the devices (computers, tablets, phones etc) that we use work by following instructions that a person has given to them Explain that we use sequences of instructions to tell devices and the software on them what they should do e.g. to tell a robot how it should move Explain that to 'program' a device, such as a robot, means to give it instructions on what it should do Explain that when we program devices and the software on them, they will only do exactly what we tell them to do – they cannot guess what we mean Tell the children that they are all going to pretend to be robots and can only move when the teacher has given them an instruction to do so, and can only move in that exact way i.e. they can only do what they have been 'programmed' to do Introduce the key directional vocabulary for the lesson, using the vocabulary cards: forwards, backwards, left, right, left turn and right turn To help the children tell left from right: • tell the children to make the letter L with their thumb and index finger on each hand – the left hand should make the letter L the 'right way round' • stick the words left and right where the children can all see them Ask the children what the difference between left / left turn and right / right turn is, and demonstrate this Write a simple sequence for the children to follow e.g. 'forwards, forwards, forwards, left, left, forwards, forwards', and tell the children to follow it Ask the children how we could make these instructions better / shorter (by using numbers of steps e.g. 'forwards 3, left 3, forwards 2') Explain that grown up computer programmers always try to write code in the best, shortest way possible Explain that by writing this sequence for the robots (the children) and telling them to follow it, you have 'programmed' them Explain that this is using the word 'program' as a verb (a doing word) Explain that the word 'program' can also be used as a noun (a thing) Explain that the written instructions are a program: a set of instructions for completing a given task – in this case getting from A to B (from one point to another point) Write some more sequences for the children to follow, constantly reinforcing the meaning of the words 'sequence' and 'program' (as a noun and as a verb): • start with 'forwards' and 'backwards' only • introduce 'left' and 'right' • introduce 'left turn' and 'right turn' If some children do not end up in the same place as the other children, discuss with the class why this might have happened Can also start to whisper a different / incorrect sequence to one or more children and then ask the children what the difference might be between the sequence that most of them followed and the sequence that the others completed Create a course and model how to complete the main activity Debugging (the following introduces the idea of debugging but do not need to introduce the term 'debugging' in this	***Setup courses in advance of the lesson (see 'Main' section of lesson plan)*** Hall or large space Hoops, mats, cones, bean bags One set of instructional language cards for display: 1) print out, one per page 2) enlarge 3) laminate 4) keep for next year Enough sets of instructional language cards for each group: 1) print out with 6 on page 2) photocopy 3) laminate 4) keep for next year Paper and pencils and / or whiteboards and pens Assessment sheets	MUST: give and follow one instruction at a time e.g. 'forward 1' SHOULD: give and follow *more than* one instruction at a time e.g. 'forward 1, left 2, backwards 3' COULD: write complete programmes, test them and fix any errors

		lesson): 1) create a course for one child to come and be a robot for 2) write a program for the child robot to follow to navigate the course, but have a deliberate error in the program e.g. 'left 2' instead of 'right 2' 3) ask the other children to think, pair, share to spot and correct the error in the program 4) take a suggested correction and re-run the program to test if it is now correct **Main:** Have pre-set courses for the children to complete to get from one hoop / mat to another hoop / mat, with the courses becoming more complex Can vary these courses throughout the lesson, or else let the children design their own courses, so that they are not completing the same course more than once A course can be made more difficult by: • not being straight i.e. the children have to use instructions other than forwards • banning the use of left and right, so that left turn and right turn have to be used • adding obstacles e.g. cones • having something that the children need to collect on the way e.g. bean bags Each course should be within an enclosed, marked out area. Tell the children that they are going to take it in turns to: • be a robot • program a robot Explain to the children that they will have to program a robot (partner) to get from one hoop / mat to the other hoop / mat, using the fewest steps / instructions possible Remind the children that the child that is the robot can only move according to the instructions given to them Children to verbally give one instruction at a time e.g. to say 'forward 1', wait for their partner to stop, then say' turn left', wait for their partner to stop, say 'forward 3' etc As the children become more confident, ask them to give more than one instruction at once e.g. 'forward 1, turn left, forward 3' For children who are completing the task confidently, ask them to write their program on paper / a whiteboard, *before* running it Children to then fix any errors they find in their written program once they have run it Explain that all grown up computer programmers make mistakes and important part of their work is to test their programmes and correct any errors Emphasise that this means that the children should show their corrections, rather than pretend that their program prediction is always perfect and without any errors **Plenary:** Revise the meaning of the key vocabulary from the lesson: sequence and program (as a verb) and program (as a noun) What problems did the children encounter during the lesson e.g. partners taking bigger or smaller steps than they expected, partners forgetting to say how many steps How did the children overcome these problems e.g. agreeing a uniform step size How could we make the courses more challenging and what other instructions could be added e.g. require different sized steps, require a diagonal turn, require a jump, require a hop to avoid an obstacle, such as a 'pressure pad' etc What mistakes did the children who tried to write their programmes before running them make e.g. forgetting the number of steps, missing out steps, getting left and right mixed up How did the children overcome these problems e.g. by turning to face the direction that they would be facing at a given point in the course Children complete the pupil column of the assessment for the lesson		

2	To write and follow instructions written in 'code' To begin to understand what 'code' and 'algorithms' are To understand what 'debugging' means (45 mins)	This lesson involves a similar structure to the previous one, but introduces the terms 'code' and 'algorithm', and explains what these terms mean It also asks the children to write their instructions in symbols as 'code', rather than in words **Intro:** Ask the children to think, pair, share the instructional words that we used in the previous lesson, and revise these using the cards from Lesson 1 Ask the children to think, pair, share the meaning of the key vocabulary from the previous lesson: sequence, program (as a verb) and program (as a noun), and continually reinforce these terms in today's lesson Ask the children to think, pair, share some examples of sequences that they often perform e.g. brushing their teeth, getting dressed etc – discuss any examples given (or introduce one) that are not sequences because they can be performed in a different order e.g. making breakfast Revise how we use sequences of instructions to tell devices and the software on them what they should do Ask the children to think, pair, share some of what we did in the previous lesson Explain that grown up computer programmers have to use special languages to instruct computers, in the same way that we need to use French to give instructions to a person who only speaks French Explain that the language that computer programmers use to instruct computers is called 'code' Explain that this week we are going to learn to write in a 'code' Explain that we will be doing a similar activity to last week, but this week the 'robots' will only understand the 'code', and not words like 'forwards' or 'left' – just like real devices and their software do not understand instructions given in everyday English Introduce the symbols for forwards, backwards, left, right, turn left and turn right, *without* their labels Discuss what each symbol might mean and ask the children to justify their responses e.g. "the arrow pointing that way means 'go left' because it is pointing to the left" Show the children the symbols *with* their labels, and explain / revise what each symbol / word means, including physically demonstrating how a child should move for each command Write a short sequence in the style of the previous lesson e.g. 'forwards, forwards, forwards, left, left, forwards, forwards'. Ask the children to write on their whiteboards how they think they would write this using the symbols that we will use for our 'code' today (it would be ↑↑↑←←↑↑) Ask the children how we could make these instructions better / shorter (by using numbers of steps e.g. ↑3←2↑2). Explain that a sequence of instructions such as ↑↑↑←←↑↑ or ↑3←2↑2 is called an 'algorithm' Explain that we will be writing 'algorithms' using 'code' for our robots Revise how a 'program' is a set of instructions for performing a task, such as getting from one point to another point Explain that a 'program' is made up of one or more 'algorithms', written in 'code' Tell the children that today's 'robots' must ignore any verbal instructions, pointing, instructions written in words etc – they can only do what the written 'code' tells them to do i.e. they can only do what they have been 'programmed' to do Write some more sequences for the children to follow, using the symbols, and constantly reinforce the meaning of the words 'sequence', 'program' (as a noun and as a verb), 'code' and 'algorithm': • start with 'forwards' and 'backwards' only • introduce 'left' and 'right' (to help the children tell left from right tell them to make the letter L with their thumb and index finger) • introduce 'left turn' and 'right turn' If some children do not end up in the same place as the other children, discuss with the class why this might have happened Can also write a different / incorrect program on a small whiteboard / piece of paper for one or more children to follow, and then ask the children what the difference might be between the program that most of them followed and the program that the others completed Create a course and model how to complete the main activity Debugging: 1) create a course for one child to come and be a robot for 2) write a program for the child robot to follow to navigate the course, but have a deliberate error in the program 3) ask the other children to think, pair, share to spot and correct the error in the program 4) take a suggested correction and re-run the program to test if it is now correct Explain to the children that the process of testing code, and finding and fixing errors in it, is called 'debugging' – 'bugs'	***Setup courses in advance of the lesson (see 'Main' section of lesson plan)*** Hall or large space Hoops, mats, cones, bean bags Instructional language cards from previous lesson One set of instructional language cards for this lesson for display – labelled and unlabelled: 1) print out, one per page 2) enlarge 3) laminate 4) keep for next year Enough sets of instructional language cards for this lesson for each group: 1) print out ones with labels and 6 on page 2) photocopy 3) laminate 4) keep for next year Paper and pencils and / or whiteboards and pens (for intro and main) Recording sheets Assessment sheets	MUST: create and follow instructions written in 'code' e.g. ↑3 SHOULD: write programmes in symbols, before running them, then test and debug them COULD: design a course and program for other children to debug and / or create and include their own instructions and corresponding symbols

		being mistakes in the code Explain that debugging is something that all grown up computer programmers have to do **Main:** Have pre-set courses for the children to complete to get from one hoop / mat to another hoop / mat, with the courses becoming more complex Can vary these courses throughout the lesson, or else let the children design their own courses, so that they are not completing the same course more than once A course can be made more difficult by: • not being straight i.e. the children have to use instructions other than forwards • banning the use of left and right, so that left turn and right turn have to be used • adding obstacles e.g. cones • having something that the children need to collect on the way e.g. bean bags Each course should be within an enclosed, marked out area Explain to the children that they will have to program a robot (partner) to get from one hoop / mat to the other hoop / mat, using the fewest steps / instructions possible Remind the children that the child that is the robot can only follow the code Children to write one piece of code at a time e.g. write '↑1' and show their partner, wait for their partner to stop, then write '←2' and show their partner, wait for their partner to stop etc As the children become more confident, ask them to give more than one instruction at once e.g. write '↑1 ←2' and show their partner For children who are completing the task confidently, ask them to write their program, *before* running it Children to then fix any errors they find in their written program once they have run it Emphasise that the children should show their corrections, rather than pretend that their program prediction is always perfect and without any errors Extension option 1: 1) children to design a course 2) write the program for it, but with one deliberate mistake 3) ask another pair / group to come and use their program to navigate the course, find the error and correct it Extension option 2: Children to design their own course to include some instructions, and some symbols for them, that they make up e.g. to include a diagonal step forward - , a jump - etc **Plenary:** Revise the meaning of the key vocabulary from the lesson: sequence, program (as a verb), program (as a noun), code, debug and algorithm Discuss if using the code was easier or more difficult than giving the instructions in words (as we did in the previous lesson) and why e.g. easier because it was quicker to write than the words; harder because you had to remember what each symbol meant What mistakes did the children who tried to write their programmes before running them make e.g. forgetting the number of steps, missing out steps, getting left and right mixed up How did the children overcome these problems e.g. by turning to face the direction that they would be facing at a given point in the course What other instructions could we add and what symbols could they be represented with e.g. large step forward - ↑L, diagonal step forward - , jump - , hop - H etc Children complete the pupil column of the assessment for the lesson		