Progression of Computing – Computer Science

<table>
<tr><th></th><th>KNOWLEDGE</th><th>PERFORMANCE OF SKILLS</th></tr>
<tr><td>EYFS</td><td colspan="2">Despite computing not being explicitly mentioned within the Early Years Foundation Stage (EYFS) statutory framework, there are many opportunities for young children to use technology to solve problems and produce creative outcomes. In particular, many areas of the framework provide opportunities for pupils to develop their ability to use computational thinking effectively, such as through undertaking projects involving the concepts and approaches which develop skills needed to access the National Curriculum when they transfer into year 1. Here is a sample of the things we do at Chilton to develop Computing skills.
- Taking a photo using an iPad
- Use of directional and positional language
- Following instructions
- Access to devices such as laptops, phones, tills, scanners, remote control toys, Purple Mash, online interactive board games.</td></tr>
<tr><td>Year 1</td><td>Unit: Computer Science

Prior knowledge

• To know that an algorithm is a set of instructions.
• To know computer programs turn algorithms into codes.
• To know that programs follow precise and unambiguous instructions.
• To know if something does not work it is because the code is incorrect.</td><td>All children (WTS)
• Can sometimes use algorithms as programs on digital devices.
• Can sometimes create simple programs.
• Can sometimes begin to debug simple programs.
• Can sometimes predict the behaviour of simple programs by using logical reasoning.
• Can sometimes read code one line at a time.
• Can sometimes work out what is wrong when steps are out of order in instructions.
• Can sometimes make good guesses on what is going to happen in a program.

Most children (EXS)
• Can use algorithms as programs on digital devices.
• Can create simple programs.
• Can begin to debug simple programs.
• Can predict the behaviour of simple programs by using logical reasoning.
• Can read code one line at a time.
• Can work out what is wrong when steps are out of order in instructions.
• Can make good guesses on what is going to happen in a program.
Some children (GDS)
• Can always use algorithms as programs on digital devices.
• Can always create simple programs.
• Can always begin to debug simple programs.</td></tr>
</table>

		• Can always predict the behaviour of simple programs by using logical reasoning. • Can always read code one line at a time. • Can always work out what is wrong when steps are out of order in instructions. • Can always make good guesses on what is going to happen in a program.
Year 2	**Unit: Computer Science** **Prior knowledge** • To know that an algorithm is a set of instructions. • To know computer programs turn algorithms into codes. • To explain what an algorithm is. • To explain what will happen in a program. • To understand that algorithms follow a sequence. • To understand how to debug a program.	**All children (WTS)** • Can sometimes plan algorithms carefully for them to work when turned into code. • Can sometimes design and create a simple program. • Can sometimes find and correct errors in programs. • Can sometimes identify something in a program that has an action or effect. • Can sometimes debug simple programs. **Most children (EXS)** • Can plan algorithms carefully for them to work when turned into code. • Can design and create a simple program. • Can find and correct errors in programs. • Can identify something in a program that has an action or effect. • Can debug simple programs. **Some children (GDS)** • Can always plan algorithms carefully for them to work when turned into code. • Can always design and create a simple program. • Can always find and correct errors in programs. • Can always identify something in a program that has an action or effect. • Can always debug simple programs.
Year 3	**Unit: Computer Science** **Prior knowledge** • To explain what an algorithm is. • To know how to turn an algorithm into a code. • To know a variable stores information while a program is running. • To know 'if' statements, repetition and variables. • To know how the internet can be used for communication.	**All children (WTS)** • Can sometimes make real-life situations into an algorithm. • Can sometimes design an algorithm carefully. • Can sometimes identify errors in programs. • Can sometimes experiment with timers in programs. • Can sometimes identify the different in using a timer or repeated command. • Can sometimes read programs with several steps.

	• To know how to use email to respond to others.	• Can sometimes predict what a program will do. **Most children (EXS)** • Can make real-life situations into an algorithm. • Can design an algorithm carefully. • Can identify errors in programs. • Can experiment with timers in programs. • Can identify the different in using a timer or repeated command. • Can read programs with several steps. • Can predict what a program will do. **Some children (GDS)** • Can always make real-life situations into an algorithm. • Can always design an algorithm carefully. • Can always identify errors in programs. • Can always experiment with timers in programs. • Can always identify the different in using a timer or repeated command. • Can always read programs with several steps. • Can always predict what a program will do.
Year 4	**Unit: Computer Science** **Prior knowledge** • To understand how to debug a program. • To know a variable stores information while a program is running. • To know how to change variables in programming. • To identify errors in code. • To understand that network and communication components are in many devices.	**All children (WTS)** • Can sometimes turn a real-life situation into an algorithm, using a design that shows how this is done. • Can sometimes use repetition in coding. • Can sometimes use timers in programs to create repetition effects. • Can sometimes use selection in programming. • Can sometimes use variables in programming. • Can sometimes use user inputs and outputs. • Can sometimes recognise the main components of hardware which allow computers to join and form a network. • Can sometimes read programs that contain several steps and predict the outcomes. **Most children (EXS)** • Can turn a real-life situation into an algorithm, using a design that shows how this is done. • Can use repetition in coding.

		• Can use timers in programs to create repetition effects. • Can use selection in programming. • Can use variables in programming. • Can use user inputs and outputs. • Can recognise the main components of hardware which allow computers to join and form a network. • Can read programs that contain several steps and predict the outcomes. **Some children (GDS)** • Can always turn a real-life situation into an algorithm, using a design that shows how this is done. • Can always use repetition in coding. • Can always use timers in programs to create repetition effects. • Can always use selection in programming. • Can always use variables in programming. • Can always use user inputs and outputs. • Can always recognise the main components of hardware which allow computers to join and form a network. • Can always read programs that contain several steps and predict the outcomes.
Year 5	**Unit: Computer Science** **Prior knowledge** • To understand that network and communication components are in many devices. • To know that organising code supports debugging. • To know the importance of computer networks and how they solve problems and enhance communication. • To recognise the main dangers through computer networks. • To explain what personal information is and know how to keep it safe.	**All children (WTS)** • Can sometimes make more complex real-life problems into algorithms. • Can sometimes test and debug programs. • Can sometimes convert algorithms that contain sequence, selection, and repetition into code. • Can sometimes use sequence, selection, repetition, and other coding structures. • Can sometimes organise code carefully. • Can sometimes use appropriate online communication according to digital content. **Most children (EXS)** • Can make more complex real-life problems into algorithms. • Can test and debug programs. • Can convert algorithms that contain sequence, selection, and repetition into code. • Can use sequence, selection, repetition, and other coding structures. • Can organise code carefully.

		• Can use appropriate online communication according to digital content. **Some children (GDS)** • Can always make more complex real-life problems into algorithms. • Can always test and debug programs. • Can always convert algorithms that contain sequence, selection, and repetition into code. • Can always use sequence, selection, repetition, and other coding structures. • Can always organise code carefully. • Can always use appropriate online communication according to digital content.
Year 6	**Unit: Computer Science** **Prior knowledge** • To know the importance of computer networks and how they solve problems and enhance communication. • To understand and interpret a program in parts. • To explain what a program is. • To explain the difference between the internet and the World Wide Web. • To explain what WAN and LAN is and describe the process of how access to the internet in school is possible.	**All children (WTS)** • Can sometimes turn a complex programming task into an algorithm. • Can sometimes identify the important aspects of a programming task. • Can sometimes decompose important aspects of a programming task. • Can sometimes identify appropriate coding structures. • Can sometimes test and debug programs to identify a bug cause. • Can sometimes identify a specific line of code that is causing a problem. • Can sometimes translate algorithms that include sequence, selection, and repetition into code. • Can sometimes use inputs and outputs. • Can put separate parts of a program together in an algorithm **Most children (EXS)** • Can turn a complex programming task into an algorithm. • Can identify the important aspects of a programming task. • Can decompose important aspects of a programming task. • Can identify appropriate coding structures. • Can test and debug programs to identify a bug cause. • Can identify a specific line of code that is causing a problem.

		• Can translate algorithms that include sequence, selection, and repetition into code. • Can use inputs and outputs. • Can put separate parts of a program together in an algorithm **Some children (GDS)** • Can always turn a complex programming task into an algorithm. • Can always identify the important aspects of a programming task. • Can always decompose important aspects of a programming task. • Can always identify appropriate coding structures. • Can always test and debug programs to identify a bug cause. • Can always identify a specific line of code that is causing a problem. • Can always translate algorithms that include sequence, selection, and repetition into code. • Can always use inputs and outputs. • Can always put separate parts of a program together in an algorithm