Cambridgeshire Progression in Computing Capability

Programming:

Purpose of study ~ Computing programmes of study: Key stages 1 and 2

A high-quality computing education equips pupils to use computational thinking and creativity to understand and change the world. Computing has deep links with mathematics, science, and design and technology, and provides insights into both natural and artificial systems. The core of computing is computer science, in which pupils are taught the principles of information and computation, how digital systems work, and how to put this knowledge to use through programming.

Building on this knowledge and understanding, pupils are equipped to use information technology to create programs, systems and a range of content. Computing also ensures that pupils become digitally literate – able to use, and express themselves and develop their ideas through, information and communication technology – at a level suitable for the future workplace and as active participants in a digital world.

Aims:

- The national curriculum for computing aims to ensure that all pupils:
- can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation
- can analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems
- can evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems
- are responsible, competent, confident and creative users of information and communication technology.

Theme Overview: Programming

It's worth noting that computer science aims to cover two distinct, but related, aspects. There's a focus on computer science itself (the ideas and principles that underpin how digital technology works) but this sits alongside the practical experience of programming, almost certainly the best way for primary pupils to learn about computer science.

Computer Science is more than programming, but programming is an absolutely central process for Computer Science. In an educational context, programming encourages creativity, logical thought, precision and problem-solving, and helps foster the personal, learning and thinking skills required in the modern school curriculum. Programming gives concrete, tangible form to the idea of "abstraction", and repeatedly shows how useful it is.

Computing at School (CAS) March 2012: Computing in the National Curriculum – A Guide for Primary Teachers

Cambridgeshire Progression in Computing Capability

	Early Capability		Middle Capability		Later Capability	
	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6
National Curriculum	▪ 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		▪ Design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts ▪ Use sequence, selection, and repetition in programs; work with variables and various forms of input and output ▪ Use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs			
Cambridgeshire Capability Statements	Pupils create, **debug** and implement instructions (simple **algorithms**) as **programs** on a range of digital devices. Pupils understand that **digital devices** follow precise and unambiguous instructions. Pupils understand that digital devices can **simulate** real situations.	Pupils understand that **algorithms** are implemented as **programs** on **digital devices.** Pupils create and **debug programs** to achieve specific goals and understand the importance of **sequence**. Pupils use the **principles of logical reasoning** to plan and predict the behaviour of simple **programs**. Pupils solve problems on and off screen.	Pupils create **programs** to accomplish specific goals using an increasing range of **digital devices** and **applications**. They can **decompose** programs to test them and understand how making even small changes to an **algorithm** can have a significant impact on the outcome. They begin using **simple repetition** (e.g. *'repeat x times'* and *'repeat forever'*) and understand how this can be used to improve **efficiency** in their programs.	Pupils create and debug **programs** containing **simple repetition** (e.g. *'repeat x times'* and *'repeat forever'*) as well as more **complex repetition** (e.g. *'nested loops'*) Pupils increasingly use their programming capability to control or simulate a range of different **outputs** in **physical systems.** Pupils begin to explore and notice the similarities and differences between **programming languages** and use this knowledge to help them create and **debug programs** efficiently.	Pupils create, **deconstruct** and refine **programs** to accomplish specific goals. They create programs with **loops** which terminate when **conditions** are met or continue whilst **conditions** are present (e.g. *'repeat until'* and *'repeat whist'*). Pupils understand and use simple **selection** (e.g. *if/then* and *if/then/else*) to create **interactive programs** based on **conditions** being met / not met. They begin to use simple **operators** within their programs.	Pupils create, **deconstruct** and refine an increasingly complex range of **programs** to accomplish specific goals. Pupils create **programs** which store, change and report **variables** (e.g. scores in a game or time) and can include multiple **variables** in a single **program**. Pupils can explain why they have structured **algorithms** as they have and describe the effect this has on a **program**.

education
info@theictservice.org.uk

Draft – August 2020
www.theictservice.org.uk/primary-computing