Computing

Preparing students for tomorrow, bit by bit

The Computing department will help to create, share, and apply knowledge in all branches of Computer Science and ICT. We will educate students to be successful, ethical, and effective problem-solvers with a passion to innovate and create, rather than just passive consumers and users of technology. We will develop an understanding and appreciation of all aspects of digital products, from how they work to how they look. We will foster curiosity and encourage exploration to create students who can contribute positively to the well-being of our society and who are prepared to tackle the complex 21st Century challenges facing the world.

Summary focus areas:

- Innovate, create, develop
- Solving 21st Century problems
- Active developers not passive consumers

Autumn		Spring		Summer	
Logical thinking and reasoning	Algorithms Pseudo-code	Basic programming constructs	Advanced Scratch programming	Computer systems and hardware	Developing systems

Homework for Computing is set weekly to support and extend the students' studies from their lessons. Work may be a mixture of practical, computer-based tasks and paper-based written work or design tasks. Activities set as homework may be:

- Preparatory work or research ahead of a new topic or concept being discussed in lessons.
- Extension work that allows the student to explore a topic in more depth or in other contexts.
- Application work that allows students to practise skills or demonstrate abilities.

Students are expected to spend around an hour on a homework activity each week and work is marked promptly to help students to identify and understand their weaknesses to make incremental improvements over the course of the year.

Unit	Duration (lessons)	Learning Objectives/Outcomes
Logical thinking and reasoning	8	• understand the concept of logic • understand the basic logical operations – AND, OR, NOT • understand how to approach problems in a logical, structure and thorough manner • be able to demonstrate logical thinking and reasoning through the completion of puzzles and logic problems
Algorithms	8	• understand that algorithms are computational solutions that always finish and return an answer • be able to interpret simple algorithms to deduce their function • be able to create algorithms to solve simple problems • be able to detect and correct errors in simple algorithms
Pseudo-code	8	• understand that pseudo-code is a simplified computer programming-like way of representing how a program is supposed to work • understand a range of pseudo-code systems, including flowcharts • be able to interpret simple pseudo-code programs to deduce their function • be able to write simple pseudo-code programs to solve problems
Basic programming constructs	10	• understand what is meant by the terms data and information • be able to describe the difference between a constant and a variable • understand when to use constants and variables in problem solving scenarios • understand the different data types available to them. As a minimum, students should know about integer, Boolean, real, character and string data types and how these are represented in the programming language(s) they are using • be able to explain the purpose of data types within code understand and be able to program with 1 and 2 dimensional arrays • understand the need for structure when designing coded solutions to problems • understand how problems can be broken down into smaller problems and how these steps can be represented by the use of devices such as flowcharts and structure diagrams • understand and be able to describe the basic building

		blocks of coded solutions (i.e. sequencing, selection and iteration) • know when to use the different flow control blocks (i.e. sequencing, selection and iteration) to solve a problem • be able to use NOT, AND and OR when creating Boolean expressions and have experience in using these operators within coded solutions
Advanced Scratch programming	10	• understand what procedures and functions are in programming terms • know when the use of a procedure or function would make sense and would simplify the coded solution • know how to write and use their own simple procedures and functions • be able to use complex mathematical constructs to control the flow and outcomes of a program (e.g. random numbers, modulo) • be able to devise, write and test programs to solve a range of real-life problems • understand how programs can control and respond to real-world objects through basic robotic control and sensor feedback
Computer systems and hardware	12	• be able to define a computer system (i.e. hardware and software working together to create a working solution) • understand and be able to discuss the importance of computer systems to the modern world • understand that computer systems must be reliable and robust and be able to discuss the reasons why this is important • be able to describe and explain the fundamental pieces of hardware required to make a functioning computer system • be able to discuss how developments in different hardware technologies (including memory and processor) are leading to exciting innovative products being created, e.g. in the mobile and gaming industries • be able to categorise devices as input or output depending on their function • be able to describe the purpose of the processor (CPU) • understand how different components link to a processor (ROM, RAM, I/O, storage, etc) • be able to explain the effect of common CPU characteristics on the performance of the processor. These should include clock speed, number of cores and cache size/types • know the differences between non-volatile and volatile memory • understand the purpose of both types of memory and when each should be used • be able to explain the purpose of virtual memory and

		cache memory • be able to explain the concept that data and instructions are stored in memory and processed by the CPU • understand what secondary storage is and be able to explain why it is required • be able to describe the most common types of secondary storage • understand how optical media, magnetic media and solid state work • understand that computers use the binary alphabet to represent all data and instructions • understand the terms bit, nibble, byte, kilobyte, megabyte, gigabyte and terabyte • understand that a binary code could represent different types of data such as text, image, sound, integer, date, real number • understand how binary can be used to represent positive whole numbers (up to 255) • understand how sound and bitmap images can be represented in binary • understand how characters are represented in binary and be familiar with ASCII and its limitations • understand why hexadecimal number representation is often used and know how to convert between binary, denary and hexadecimal
Developing systems	12	• understand the software development life cycle • be able to explain what commonly occurs at each stage of the software development life cycle • be able to identify at which stage of the software development life cycle a given step would occur • understand that there are several lifecycle models that can be used (e.g. cyclical, waterfall, spiral) • be able to discuss the advantages and disadvantages of these lifecycle models • understand what prototyping is • be able to discuss the advantages and disadvantages of using prototyping when developing solutions • have experience of using prototyping to create solutions to simple problems • understand the need for rigorous testing of coded solutions • understand the different types of tests that can be used, including unit/modular testing • be able to create suitable test plans and carry out suitable testing to demonstrate their solutions work as intended • be able to hand test simple code designs/algorithms using trace tables