Computer Science Year 10 Curriculum Overview

What is the Year 10 Computer Science curriculum aiming to achieve?		
What do we want our Year 10 Computer Scientists to be like?	How are we building on prior learning?	How can parents/carers support their child's learning?
• Be confident and resilient with their approach to problems • Be willing to experiment and explore programming problems • Be a reflective learner • Be able to decompose computational problems and make adaptations where needed	• Building upon an introduction to programming using Turtle • Understanding how computers work, and appreciate computer language • Building upon students ability to 'think like a computer'	• Encourage students to use IT programs available outside of lesson • Encourage students to explore and trouble shoot when using IT devices • Talk to students about their homework and their learning • Encourage students to purchase the revision material available

How are we organising the Year 10 Computer Science curriculum?						
	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
Topics	1.2.3: Units 1.2.4: Data Storage 2.1.1:Computational thinking	1.2.1: Primary Storage 1.2.2: Secondary Storage 2.1.2: designing, creating and refining algorithms 2.2.1: Programming fundamentals: Arithmetic and Boolean operators, Sequence and selection 2.2.2: Data Types	1.5.1: Operating Systems 1.5.2: Utility software 2.1.2: designing, creating and refining algorithms 2.2.1: iteration 2.4.1: Boolean logic	2.1.2: designing, creating and refining algorithms 2.2.1: iteration 2.4.1: Boolean logic 2.2.3: procedures and file handling	2.1.1:Computational thinking 2.2.3: procedures and file handling 1.4.1: Threats to computer systems and Network	1.4.1: Threats to computer systems and Network 1.4.2: identifying and preventing vulnerabilities Programming project
Threshold Concepts	Understanding binary, calculate capacity, convert denary to binary and Hexadecimal. Understand binary shifts. Understanding the principles of computational thinking.	Understand the types of storage and explore their characteristics. Understand the suitability of each type for the given scenario. Design algorithms. Understand datatypes. Explore basic programming constructs including the use of arithmetic and Boolean operators.	Understand the functions and features of an operating systems and utility software. Refining algorithms Explore count control and condition controlled iteration. Understand Truth tables and logic gates	Refining algorithms Explore count control and condition controlled iteration. Understand Truth tables and logic gates Understand the use of sub programs. Explore basic file handing operations.	Computational Thinking core concepts . Understand the use of sub programs. Explore basic file handing operations. Understand the threats and vulnerabilities posed to computer systems.	Understand the threats and vulnerabilities posed to computer systems. Explore basic string manipulation. Consolidation of programming techniques to date.
Skills	Convert binary to denary and vice versa. Convert binary to Hexadecimal. Be able to apply computational logic in order to problem solve.	Apply storage characteristics to select the appropriate method given the context. Be able to write an algorithm to answer a problem. Be able to cast using the correct data type.	Be able to use trace tables to assess the accuracy of an algorithm. Use iteration in Python. Be able to construct a truth table and logic gates for AND, NOT and OR gates. Use Boolean operators and	Discuss the wider implications of technology and apply opinions to given scenarios.	Be able to code using procedures and functions. Be able to program opening, reading and writing to/from files. Identify network threats.	Be able to perform basic string manipulation techniques. Identify network threats.

			arithmetic operators in Python.			
Enrichment within the curriculum	Students will have the opportunity to develop their computational thinking across the curriculum in different situations. Students complete independent work, paired work and group tasks throughout their time in Computer Science. Much of year 10 is about building their resilience in terms of digital literacy, and about developing their problem solving skills.					
Cross curricular links	• Computer Science is indeed a science where students work through and anticipate problems and solutions • Mathematics is another core foundation of Computer Science with many of the topics being data based and logic based; this is an excellent subject for developing your mathematical problem solving. • Within most units of work students have the opportunity to develop their use of technical vocabulary.					
Extra-curricular opportunities	Throughout an academic year students could have the opportunity to take part in various external Computer Science schemes within the school coding club. Students are encouraged to use various online sources to develop their computational thinking skills. There is also the use of revision café at particular times of the year. Support is also available during Tuesday and Wednesday lunch times.					

What are the intended outcomes of the Year 10 Computer Science curriculum?						
	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
Opportunities to show progress (Assessments)	End of Unit assessments	End of Unit assessments	End of Unit assessments	End of Unit assessments	End of Unit assessments	End of Unit assessments. PPEs
Impact on personal development (SMSC)	*Students develop their resilience to approaching Computational tasks, they consider the wider implications of technology and the impact these have within society. Programming is sometimes delivered in a paired: driver and navigator style of delivery reinforcing inclusion and their idea of supporting their peers.*					
Preparation for the next stage of education	*Year 10 of the GCSE in Computer Science is framed in such a way that it provides the foundation knowledge before moving onto some more technical theory content, or some trickier programming skills. Year 11 has a much heavier focus upon Component 2, which has a larger focus on application, problem solving and refinement.*					