Sheldon School

BE KIND | BE BRAVE | BE THE BEST YOU

Curriculum Plan

KS4 – Computer Science

Year 10

Golden Threads	Enrichment	Review and Evaluation
How to use computational methods to create algorithms solve problems How to use a high-level programming language to turn algorithms into runnable code What are the fundamental principles behind how computers operate?	After school coding support club	Summer 2026

	Topics & Substantive Knowledge	Disciplinary Knowledge	Assessment	Misconceptions	Key Vocabulary	Knowledge Tracking
Term 1	1.1 Decomposition and abstraction 2.1 Binary 6 Problem solving with programming (this topic continues in every term) 2.1.1 understand that computers use binary to represent data (numbers, text, sound, graphics) and program instructions and be able to determine the maximum number of states that can be represented by a binary pattern of a given length 2.1.2 understand how computers represent and manipulate unsigned integers and two's complement signed integers 2.1.3 be able to convert between denary and 8-bit binary numbers (0 to 255, -128 to +127) 2.1.4 be able to add together two positive binary patterns 2.1.5 understand the concept of overflow in relation to the number of bits available to store a value 2.1.6 understand why hexadecimal notation is used and be able to convert between hexadecimal and binary	1.1.1 understand the benefit of using decomposition and abstraction to model aspects of the real world and analyse, understand and solve problems 6.1.1 be able to use decomposition and abstraction to analyse, understand and solve problems 6.1.2 be able to read, write, analyse and refine programs written in a high-level programming language 6.1.3 be able to convert algorithms (flowcharts, pseudocode*) into programs 6.1.4 be able to use techniques (layout, indentation, comments, meaningful identifiers, white space) to make programs easier to read, understand and maintain 6.1.5 be able to identify, locate and correct program errors (logic, syntax, runtime) 6.2.1 understand the function of and be able to identify the structural components of programs (variables, initialisation and assignment statements, command sequences, input/output) 6.2.2 be able to write programs that make appropriate use of sequencing 6.3.1 be able to write programs that make appropriate use of primitive data types (integer, real, Boolean, char) 6.3.2 be able to write programs that make appropriate use of variables 6.4.1 be able to write programs that accept and respond appropriately to user input 6.5.1 be able to write programs that use arithmetic operators (addition, subtraction, division, multiplication, modulus, integer division, exponentiation)	5 question MCQ retrieval at the start of each lesson Term 1 Week 7 short answer questions	Only computers use algorithms How computers store data Text-based programming is an easy skills to learn	Abstraction Algorithm Binary Bit Byte Decomposition Denary Flow chart Indentation Integers Overflow Error Pseudocode Sequence Sign Bit Two's Complement Variable	Topic 6 is an ongoing topic across all terms in year 10 and year 11 with links to 1.1 2.1 links to all other topics

Sheldon School

BE KIND | BE BRAVE | BE THE BEST YOU

Curriculum Plan

KS4 – Computer Science

Year 10

	Topics & Substantive Knowledge	Disciplinary Knowledge	Assessment	Misconceptions	Key Vocabulary	Knowledge Tracking
Term 2	2.2 Data representation 2.3 Data Storage and Compression 6 Problem solving with programming 2.2.1 understand how computers encode characters using 7-bit ASCII 2.2.2 understand how bitmap images are represented in binary (pixels, resolution, colour depth) 2.2.3 understand how analogue sound is represented in binary (amplitude, sample rate, bit depth, sample interval) 2.2.4 understand the limitations of binary representation of data when constrained by the number of available bits 2.3.2 understand the need for data compression and methods of compressing data (lossless, lossy)	6.2.1 understand the function of and be able to identify the structural components of programs (selection, repetition) 6.2.2 be able to write programs that make appropriate use of selection, repetition (condition-controlled) 6.3.3 be able to write programs that manipulate strings (length, position, substrings, case conversion) 6.5.2 be able to write programs that use relational operators (equal to, less than, greater than, not equal to, less than or equal to, greater than or equal to)	5 question MCQ retrieval at the start of each lesson Week 7 short answer questions	Binary is the only way to represent data in a computer Text is literally stored as characters in memory Audio, image and video files are not simply made up of binary data	ASCII Arithmetic Binary Shift Encode Hexadecimal Logical Binary shift Loop Repetition Selection Analogue Bit depth Bit rate Bitmap Colour depth Metadata Pixel Resolution Sample rate Sampling Vector	2.2 and 2.3 link to 2.1
Term 3	3.1 Hardware 6 Problem solving with programming 3.1.1 understand the von Neumann stored program concept and the role of main memory (RAM), CPU (control unit, arithmetic logic unit, registers), clock, address bus, data bus, control bus in the fetch-decode-execute cycle 3.1.2 understand the role of secondary storage and the ways in which data is stored on devices (magnetic, optical, solid state) 6.2.1 understand the function of and be able to identify the structural components of programs (subprograms, parameters)	6.2.1 understand the function of and be able to identify the structural components of programs (subprograms, parameters) 6.2.2 be able to write programs that make appropriate use of sequencing, selection, repetition (count-controlled), iteration (over every item in a data structure) and single entry/exit points from code blocks and subprograms 6.3.1 be able to write programs that make appropriate use of one-dimensional structured data types (string, array) 6.6.1 be able to write programs that use pre-existing (built-in, library) and user-devised subprograms (procedures, functions) 6.6.2 be able to write functions that may or may not take parameters but must return values, and procedures that may or may not take parameters but do not return values	5 question MCQ retrieval at the start of each lesson Week 6 short answer questions	Code in procedures and functions executes straight away	Accumulator ALU – Arithmetic and Logic Unit Cache CPU – Central Processing Unit Clock CU – Control Unit Embedded computer FDE – Fetch Decode Execute PC – Program Counter RAM – Random Access Memory ROM – Read Only Memory Register Von Neumann architecture	3.1 links to 3.2

Sheldon School

BE KIND | BE BRAVE | BE THE BEST YOU

Curriculum Plan

KS4 – Computer Science

Year 10

	Topics & Substantive Knowledge	Disciplinary Knowledge	Assessment	Misconceptions	Key Vocabulary	Knowledge Tracking
Term 4	1.2 Algorithms 3.2 Software 6 Problem solving with programming 1.2.6 understand how standard algorithms (linear search) work 3.2.1 understand the purpose and functionality of an operating system (file management, process management, peripheral management, user management) 3.2.2 understand the purpose and functionality of utility software (file repair, backup, data compression, disk defragmentation, anti-malware)	6.3.1 be able to write programs that make appropriate use of two-dimensional structured data types (string, array, record) 6.4.3 understand the need for and be able to write programs that implement validation (length check, presence check, range check, pattern check)	5 question MCQ retrieval at the start of each lesson Week 6 short answer questions	The difference between an operating system, utility and application software	Antivirus Application software CLI – Command Line Interface Device Driver Disk cleaner Disk defragmentation Encryption Firewall Graphical User GUI-Interface Interrupt Malware Multitasking Operating system Peripheral System software Virus Windows	3.2 links to 3.1, 5.3 and 4.1
Term 5	5.3 Cybersecurity 6 Problem solving with programming 1.2.6 understand how standard algorithms (merge sort) work 3.2.3 understand the importance of developing robust software and methods of identifying vulnerabilities (audit trails, code reviews) 5.3.1 understand the threat to digital systems posed by malware (viruses, worms, Trojans, ransomware, key loggers) and how hackers exploit technical vulnerabilities (unpatched software, out-of-date anti-malware) and use social engineering to carry out cyberattacks	6.4.2 be able to write programs that read from and write to comma separated value text files 6.4.4 understand the need for and be able to write programs that implement authentication (ID and password, lookup)	5 question MCQ retrieval at the start of each lesson Week 6 short answer questions	Hacking is always how it is depicted in the media Hacking is always about computers and never about people and therefore protecting against it is all about computers	Adware Archiving Authentication Back doors Blagging Brute force attack Denial of service attack Hacking HTTP HTTPS Network forensics Penetration testing Pharming Phishing Proxy server Rootkit Shoulder surfing SQL injection Spyware SSL – Secure Sockets Layer Trojan horse	5.3 links to 4.1

Sheldon School

BE KIND | BE BRAVE | BE THE BEST YOU

Curriculum Plan

KS4 – Computer Science

Year 10

	Topics & Substantive Knowledge	Disciplinary Knowledge	Assessment	Misconceptions	Key Vocabulary	Knowledge Tracking
Term 6	5.1 Environmental 5.3 Cybersecurity 6 Problem solving with programming 5.1.1 understand environmental issues associated with the use of digital devices (energy consumption, manufacture, replacement cycle, disposal) 5.3.1 understand the threat to digital systems posed by malware (viruses, worms, Trojans, ransomware, key loggers) and how hackers exploit technical vulnerabilities (unpatched software, out-of-date anti-malware) and use social engineering to carry out cyberattacks 5.3.2 understand methods of protecting digital systems and data (anti-malware, encryption, acceptable use policies, backup and recovery procedures)		5 question MCQ retrieval at the start of each lesson	Computers are always negative for the environment	Adware Archiving Authentication Back doors Blagging Brute force attack Denial of service attack Hacking HTTP HTTPS Network forensics Penetration testing Pharming Phishing Proxy server Rootkit Shoulder surfing SQL injection Spyware SSL – Secure Sockets Layer Trojan horse	