THE SWEYNE PARK SCHOOL

CURRICULUM INTENT: **Computer Science**

Computer Science GCSE

In GCSE Computer Science, pupils explore how computer systems function, from the physical hardware to the software applications and programs that users interact with. They learn how to design, build, and adapt computer systems, applying these skills creatively to solve real-world problems. The course encourages pupils to make computers work for them, moving beyond basic functions to develop their own solutions. A strong emphasis is placed on understanding the risks associated with storing personal information electronically, alongside strategies for maintaining security and reducing the impact of online threats. The subject equips pupils with transferable skills, including logical thinking, problem-solving, and data handling, which are valuable across other subjects and future career paths. ICT skills are essential in modern workplaces, and the ability to write programs and manipulate data are highly regarded by employers.

Computer Science A Level

At A Level, Computer Science builds on this foundation by deepening students' understanding of how computers operate and developing advanced computational and problem-solving skills. Students gain practical experience through their NEA, where they design and implement bespoke software solutions. This includes working with Object-Oriented Programming, Event-Driven Programming, and Graphical User Interfaces. Such projects prepare students for further study at degree level or for direct entry into apprenticeships and careers in software development. The course also explores the fundamental building blocks of computing, giving students a broad and thorough understanding of the subject and opening the door to a wide range of future opportunities.

THE SWEYNE PARK SCHOOL

CURRICULUM MAP: **Computer Science**

Computer Science GCSE

Year	Knowledge (Topics /Contexts) What pupils will '**know**'.	Skills acquired What pupils will be able to '**do**'.	Concepts developed What pupils will '**understand**'.	Assessment (KPIs)
Throughout year 10 and 11	**Programming and programming concepts** - Basic programming constructs: sequence, selection, iteration. - Data types: integer, real, Boolean, character, string. - How to use variables and constants. - Concepts of functions, procedures, and scope. - String manipulation techniques. - Input/output - Structured programming principle - differences between low and high levels of programming language: - that all programming code must be translated into machine code before it is executed - the differences between and use of three types of translator: interpreter, compiler and assembler	Writing programs using Python that: - The importance of code readability, maintainability, and testing. - How scope affects variables. - The advantages of structured programming and modular design. - Use NOT, AND and OR when creating Boolean expressions - Use in-built functions - Use random number generation - Write algorithms in pseudocode involving sequence, selection and iteration - Use one- and two-dimensional arrays in the design of solutions to simple problems - Define the terms field, record and file - write simple procedures and functions - use parameters to pass data to procedures and functions	- Write programs using sequence, selection, iteration, and subroutines. - Manipulate strings - Test, debug, and refine programs systematically. - the importance of validating input - how to determine the correct output of an algorithm for a given set of data - how to identify and correct errors in algorithms	**Formative Assessment** - Regular low-stakes quizzes on core concepts and terminology - Use of mini whiteboards for spontaneous questioning and peer feedback - Annotated responses to short-answer questions with teacher guidance - Self-assessed tasks using scaffolding tools: vocabulary banks, exemplar answers, and AO-marked frameworks **Summative Assessment** - End-of-topic tests using past paper-style questions (multiple-choice, application, and extended responses) - Structured homework

Year	Knowledge (Topics /Contexts) What pupils will '**know'**.	Skills acquired What pupils will be able to '**do'**.	Concepts developed What pupils will '**understand'**.	Assessment (KPIs)
		- List validation checks that can be used on input data - write simple data validation routines - write a simple authentication routine involving a username and password - Write a test plan to test an algorithm - Use a trace table to trace through a program		- Formal assessment folders with marked and levelled work linked to feedback loops **Year 10 Mock Exams** - Paper 1: Computational Thinking and Programming Skills (shortened paper, mock under timed exam conditions - Paper 2 Computing Concepts ((shortened paper, mock under timed exam conditions) - Reflection tasks: pupils complete "What Went Well / Even Better If" sheets and action plans post-marking - Data used to inform intervention strategies and track progress across cohorts
Year 10 **Term 1**	**Data Representation** KPI3 Pupils understand the fundamentals of data representation. - Number bases: binary, decimal, hexadecimal. - Conversions between binary, decimal, and hexadecimal. - Binary arithmetic (addition, subtraction, shifts). - Character encoding systems: ASCII, Unicode. - Image representation (pixels, resolution, colour depth). - Sound representation (sampling, sample rate, bit depth). - Units of data: bit, nibble, byte, kilobyte, etc.	- Why binary is used in computers. - The impact of resolution and colour depth on image size and quality. - The trade-offs between lossless and lossy compression.	- Convert numbers between bases. - Perform binary arithmetic operations. - Calculate file sizes for images and sound files. - Choose appropriate compression methods for scenarios.	

Year	Knowledge (Topics /Contexts) What pupils will '**know'**.	Skills acquired What pupils will be able to '**do'**.	Concepts developed What pupils will '**understand'**.	Assessment (KPIs)
	- Compression techniques: lossless and lossy.			
Term 2	**Fundamentals of algorithms** KPI1 Pupils understand the fundamentals of algorithms. - The terms algorithm, decomposition, and abstraction. - Standard search algorithms: linear search, binary search. - Standard sorting algorithms: bubble sort, merge sort, insertion sort. - Representing algorithms using pseudocode and flowcharts.	- Why decomposition and abstraction are used to solve problems. - The efficiency of algorithms and the concept of computational thinking. - How different algorithms solve the same problem in different ways.	- Apply decomposition and abstraction to solve problems. - Trace through search and sort algorithms with given data. - Write, interpret, and refine algorithms using pseudocode or flowcharts.	
Term 2 -	**Computer Systems, Systems architecture** - The CPU: control unit, ALU, registers, cache. - The fetch-decode-execute cycle. - Factors affecting CPU performance: clock speed, cores, cache size. - Embedded systems and their uses. - Primary storage (RAM, ROM), secondary storage (magnetic, optical, solid state). - Tertiary storage uses.	- How hardware and software interact. - Why embedded systems are designed for specific tasks. - How performance factors impact overall processing speed. • Why RAM and ROM are both required. • Trade-offs between different types of secondary storage.	- Describe the function of CPU components. - Explain the stages of the fetch-decode-execute cycle. - Evaluate the suitability of embedded systems in given contexts • Compare different storage technologies for a given scenario. • Select appropriate storage media based on requirements.	

Year	Knowledge (Topics /Contexts) What pupils will '**know'**.	Skills acquired What pupils will be able to '**do'**.	Concepts developed What pupils will '**understand'**.	Assessment (KPIs)
	- Characteristics: capacity, speed, portability, durability, reliability, cost	• How storage characteristics affect suitability.		
Term 3 -	**Computer Networks** KPI5 Pupils are able to describe the fundamentals of computer networks - Types of networks: LAN, WAN. - Factors affecting network performance. - The roles of hardware: routers, switches, WAPs, NICs, transmission media. - Client-server vs peer-to-peer models. - The Internet, DNS, hosting, the cloud. - Virtual networks and their use.	- How different network types are structured and used. - Why factors like bandwidth and latency affect performance. - The advantages and disadvantages of cloud computing.	- Compare and contrast client-server and peer-to-peer models. - Explain how DNS works to resolve domain names. - Evaluate scenarios for using LANs, WANs, and virtual networks.	
Term 3	**Cyber Security** KPI6 Pupils are able to describe the fundamentals of cyber security. - Types of cyber threats: malware, phishing, brute force, denial of service, data interception, SQL injection. - Forms of social engineering: blagging, phishing, pharming, shoulder surfing. - Methods to identify and prevent vulnerabilities: penetration testing, anti-	- How social engineering exploits human factors. - Why organisations implement layered security measures. - The importance of protecting data and systems from attack.	- Identify and evaluate different cyber threats in scenarios. - Suggest appropriate measures to reduce security risks.	

Year	Knowledge (Topics /Contexts) What pupils will '**know'**.	Skills acquired What pupils will be able to '**do'**.	Concepts developed What pupils will '**understand'**.	Assessment (KPIs)
	malware software, firewalls, user access levels, passwords, encryption, physical security.			
Year 11 **Term 1** -	**Relational databases and structured query language (SQL)** KPI7 Pupils understand Relational databases and SQL - Key terms: database, table, record, field, primary key, foreign key, relationship (one-to-one, one-to-many). - Common data types used in databases (e.g., integer, real, Boolean, text, date/time). - Data quality concepts: validation (presence, length, type/format, range), verification (double entry, visual check). - Core SQL keywords: SELECT, FROM, WHERE, ORDER BY, AND/OR/NOT, LIKE (with wildcards), BETWEEN, IN. - Simple aggregate functions: COUNT, SUM, AVG, MIN, MAX. - Purpose of JOINs to query related data across tables (focus on INNER JOIN).	- Why relational databases are used to reduce duplication and maintain data integrity. - The role of primary and foreign keys in enforcing relationships between tables. - The difference between validation and verification and how they improve data quality. - How filtering, sorting, and aggregation change the result set returned by a query. - How JOIN operations use key fields to combine data from related tables. -	- Design simple table structures with suitable fields, data types, and a clear primary key. - Sketch a simple entity–relationship diagram (ERD) for a one-to-many relationship. - Write SQL queries to: select specific fields, filter with WHERE using comparison and logical operators, and sort with ORDER BY. - Use LIKE with wildcards, BETWEEN and IN to refine queries. - Apply aggregate functions (COUNT, SUM, AVG, MIN, MAX) and, where appropriate, GROUP BY to summarise results. - Write a basic two-table INNER JOIN using a foreign key to retrieve related records.	**Formative Assessment** - Regular low-stakes quizzes on core concepts and terminology - Use of mini whiteboards for spontaneous questioning and peer feedback - Annotated responses to short-answer questions with teacher guidance - Summative Assessment - End-of-topic tests using past paper-style questions (multiple-choice, application, and extended responses) **Structured homework** - Year 11 Mock Exams - Paper 1: Computational Thinking and Programming Skills (mock under timed exam conditions - Paper 2 Computing Concepts (paper, mock under timed exam conditions)
Term 2	**Fundamentals of algorithms** KPI1 Pupils understand the fundamentals of algorithms.			

Year	Knowledge (Topics /Contexts) What pupils will '**know'**.	Skills acquired What pupils will be able to '**do'**.	Concepts developed What pupils will '**understand'**.	Assessment (KPIs)
	- the different search algorithms - the different sorting algorithms	- Compare and contrast merge sort and bubble sort algorithms.	- how binary and linear search algorithms work: - how bubble and merge sort algorithms work.	- Data used to inform intervention strategies and track progress across cohorts
Term 2	**Impact of digital technology** KPI8 Pupils can discuss the ethical, legal and environmental impacts of digital technology on wider society, including issues of privacy. - Legislation: Data Protection Act, Computer Misuse Act, Copyright, Designs and Patents Act, Creative Commons licensing, Freedom of Information Act. - Cultural and ethical issues around digital technology use. - Environmental impacts: energy use, e-waste, sustainability	- How legislation governs the use of digital data. - The impact of technology on culture, employment, and lifestyles. - The consequences of digital technology on the environment.	- Apply laws to real-world digital scenarios. - Discuss cultural, ethical, and environmental impacts with examples.	

Computer Science GCE A Level

Year	Knowledge (Topics /Contexts) What pupils will '**know'**.	Skills acquired What pupils will be able to '**do'**.	Concepts developed What pupils will '**understand'**.	Assessment (KPIs)
Throughout year 12	**4.1 Fundamentals of programming** - Built-in data types (integer, real/float, Boolean, character, string, date/time). - Programming constructs: assignment, selection, iteration, subroutines (procedures/functions). - Difference between local and global variables, and scope. - Concepts of exception handling and recursion. - Programming paradigms: procedural and object-oriented (encapsulation, inheritance). - File I/O: reading/writing binary files.	- Why different data types exist (precision, memory, operations). - How control structures combine to form logic. - Why local variables are used (avoiding unintended interference). - How exception handling increases robustness; how recursion works. - Trade-offs and design in different programming paradigms. - How file I/O differs from in-memory operations. -	- Declare and use variables with appropriate data types. - Write, adapt, and extend programs using control structures. - Use subroutines with parameters and return values; manage scope. - Implement exception handling and recursive algorithms. - Write code in procedural and object-oriented styles. - Perform file I/O operations in a programming language.	**Homework** Worksheets reinforcing lesson content. Problem solving challenges. Practice examination style questions. **Formative Assessment** • Regular low-stakes quizzes on core concepts and terminology • Use of mini whiteboards for spontaneous questioning and peer feedback • Annotated responses to short-answer questions with teacher guidance • Self-assessed tasks using scaffolding tools: vocabulary banks, exemplar answers, and AO-marked frameworks **Summative Assessment** • End-of-topic tests using past paper-style questions (multiple-choice, application, and extended responses) • Structured homework **Year 12 Mock Exams**
Year 12 Term 1	**4.5 Fundamentals of data representation** - Binary, hexadecimal, decimal systems and conversions. - Signed number representation (two's complement). - Character encoding (ASCII, Unicode). - Image representation (bitmap, vector). - Sound representation.	- Why binary is used in computing. - Trade-offs between resolution, colour depth, file size and quality. - Why error detection/correction is essential	- Convert between number bases and perform binary arithmetic. - Interpret signed binary numbers using two's complement. - Calculate file sizes of images/sound given parameters. - Design simple error checking methods.	

Year	Knowledge (Topics /Contexts) What pupils will '**know'**.	Skills acquired What pupils will be able to '**do'**.	Concepts developed What pupils will '**understand'**.	Assessment (KPIs)
	- Error detection methods (parity, checksums).			• Paper 1: Online Computational Thinking and Programming Skills (shortened paper, mock under timed exam conditions • Paper 2 Computing Concepts ((shortened paper, mock under timed exam conditions) • Reflection tasks: pupils complete "What Went Well / Even Better If" sheets and action plans post-marking • Data used to inform intervention strategies and track progress across cohorts Retrieval practice starters throughout the course. End of topic tests. Regular quizzes using Smart Revise and Isaac Computer Science.
	4.6 Fundamentals of computer systems - Types of software: system, application, utility, libraries. - Translators: compiler, interpreter, assembler, intermediate languages. - Logic gates, Boolean algebra, De Morgan's laws. - Operating system functions: scheduling, memory, I/O, file systems,	- How system software supports applications and hardware interaction. - Advantages/disadvantages of compilation vs interpretation. - How Boolean logic underpins digital circuits. - Role of operating systems in abstraction and resource management.	- Identify software categories and translator functions. - Simplify Boolean expressions and draw logic circuits. - Explain OS scheduling and memory management with examples. -	
Term 2	**4.7 Fundamentals of computer organisation and architecture** - CPU structure: registers (PC, MAR, MDR, CIR, status), buses (address, data, control). - Stored-program concept. - Instruction set operations (load, store, arithmetic, logic, branching). - Addressing modes (immediate, direct). - External devices and their characteristics.	- How CPU components interact in the fetch-execute cycle. - Trade-offs in instruction sets and addressing modes. - Differences between storage devices (speed, capacity, volatility).	-	
	4.9 Fundamentals of communication and networking	- How transmission methods and protocols affect performance. - How networks scale and why addressing/subnetting is important. - Why layered protocols are used.	- Calculate subnet masks and plan IP addressing. - Trace packet transmission through TCP/IP layers.	

Year	Knowledge (Topics /Contexts) What pupils will '**know'**.	Skills acquired What pupils will be able to '**do'**.	Concepts developed What pupils will '**understand'**.	Assessment (KPIs)
	- Transmission methods: serial, parallel, synchronous, asynchronous. - Network concepts: bandwidth, latency, packet switching, topologies, IP addressing, subnetting. - Protocols: TCP/IP stack, HTTP, SMTP, POP3, SSH. - Network security threats and mitigations.	- How security vulnerabilities arise and are managed.	- Propose mitigation strategies for network security threats	
	4.4 Theory of computation - Concepts of abstraction, automation, procedural/functional/data abstraction. - Finite state machines, regular expressions, classification of algorithmic problems. - Turing machine model.	- How abstraction and automation enable building of complex systems. - Relationships between computational models, expressiveness, decidability.	- Draw state diagrams and transition tables. - Simulate simple Turing machines. - Reason about solvable vs unsolvable problems.	
Term 3	**4.2 Fundamentals of Data Structures** - Definitions of arrays, records, queues, stacks, trees, graphs, hash tables, dictionaries. - Abstract Data Types (ADTs) and separation of interface vs implementation. - Operations: push/pop (stack), enqueue/dequeue (queue), traversal (trees/graphs).	- Why certain data structures suit particular tasks (trade-offs in operations, memory, speed). - Principle of abstraction (hiding implementation details). - Algorithmic costs (time complexity) of operations.	- Use and manipulate data structures in programs. - Design and use ADTs and choose appropriate implementations. - Implement and trace standard operations (add, remove, search).	

Year	Knowledge (Topics /Contexts) What pupils will '**know'**.	Skills acquired What pupils will be able to '**do'**.	Concepts developed What pupils will '**understand'**.	Assessment (KPIs)
	4.3 Fundamentals of algorithms - Common algorithms: linear search, binary search, merge sort, Dijkstra's algorithm. - Big-O notation for complexity analysis. - Concepts of divide-and-conquer, optimisation, heuristics.	- How algorithm efficiency is measured. - Why some problems are intractable or require heuristic approaches. - How to evaluate algorithm correctness and robustness. -	- Trace and analyse algorithms step-by-step. - Choose and adapt algorithms for problems. - Test and justify correctness of algorithms. -	
	4.11 Big Data - Definition and features of Big Data (volume, velocity, variety). - Issues: data cleaning, real-time processing, ethical/privacy concerns.	- Why traditional databases struggle with Big Data. - Challenges of ensuring quality, security, and privacy in Big Data.	- Evaluate Big Data projects and architectures. - Propose alternative approaches (e.g. distributed processing, NoSQL).	
Year 13	**4.10 Fundamentals of databases** - Tables, records, fields, keys (primary, foreign). - Relationships, queries, and normalization (1NF, 2NF, 3NF).	- Why databases use relational models. - How normalization reduces redundancy and improves integrity.	- Design normalized relational schemas. - Write SQL queries to retrieve and manipulate data.	
	4.12 Fundamentals of functional programming - Functional programming concepts: immutability, recursion, higher-order functions, first-class functions	- Differences between functional and imperative programming paradigms. - Advantages and limitations of functional approaches.	- Write functional-style code using recursion and higher-order functions. - Translate small problems into functional solutions	
-	**4.8 Consequences of uses of computing** - Ethical, legal, cultural, and social issues in computing.	- Impacts of computing on individuals, organisations, and society. - How laws regulate and shape ethical computing practices	- Analyse scenarios to identify ethical/legal consequences. - Apply legal and ethical frameworks to real-world cases	

Year	Knowledge (Topics /Contexts) What pupils will '**know'**.	Skills acquired What pupils will be able to '**do'**.	Concepts developed What pupils will '**understand'**.	Assessment (KPIs)
	- Relevant laws: data protection, copyright, cybersecurity			
Year 13 **Term 1**	**4.14 NEA – Non-Examined Assessment** – The project allows students to develop their practical skills in the context of solving a realistic problem or carrying out an investigation. The project is intended to be as much a learning experience as a method of assessment; students have the opportunity to work independently on a problem of interest over an extended period, during which they can extend their programming skills and deepen their understanding of computer science - Stages of software development: analysis, design, implementation, testing, evaluation. - Techniques: decomposition, abstraction, prototyping, algorithm design.	- Why systematic approaches improve reliability and maintainability. - How decomposition and abstraction reduce complexity. - The most important skill that is assessed is a student's ability to create a programmed solution to a problem or investigation. This is recognised by allocating 42 of the 75 available marks to the technical solution and a lower proportion of marks for supporting documentation to reflect the expectation that reporting of the problem, its analysis, the design of a solution or plan of an investigation and testing and evaluation will be concise	- Produce design artefacts (flowcharts, pseudocode, test plans). - Carry out each stage of project development in the NEA. - Evaluate software solutions effectively.	Formative feedback at specific points during the project. Retrieval practice starters throughout the course. Practice examination questions. Regular quizzes
Term 2	**Paper 1 – Skeleton Code** Pupils are provided with a program written by AQA the exam board. The program is available from September 1st. Time will be taken to understand the function of the program and the mechanics of the program, with a target of being familiar enough to be able to explain the program			

Year	Knowledge (Topics /Contexts) What pupils will '**know'**.	Skills acquired What pupils will be able to '**do'**.	Concepts developed What pupils will '**understand'**.	Assessment (KPIs)
	constructs as well as amend and add to the program online under examination conditions.			
Term 3	Revision and Reinforcement		-	