Curriculum Plans: Year 10 (Computer Science)

	Topic	Knowledge: By the end of the unit students will know:	Skills: What skills will students have developed by the end of this unit?	Key terms: What new key terms and vocabulary will be learnt in this unit?	Summative Assessment: How will pupils be assessed in this unit?
Term 1	Algorithms	• Principles of computational thinking: abstraction, decomposition, and algorithmic thinking. • The inputs, processes, and outputs of a problem and how they relate to algorithms. • Structure diagrams and their use in visualising problem-solving processes. • Pseudocode, flowcharts, and high-level languages for creating, interpreting, and refining algorithms. • Standard searching algorithms: linear search and binary search. • Standard sorting algorithms: bubble sort, merge sort, and insertion sort. • Identifying common errors in algorithms and fixing them. • Understanding and correcting algorithms through debugging and refining steps.	• Applying abstraction to simplify complex problems by focusing on essential details. • Using decomposition to break down complex problems into smaller, manageable parts. • Creating and interpreting structure diagrams to organize problem-solving approaches. • Writing pseudocode and creating flowcharts to describe problem-solving steps. • Implementing linear and binary search algorithms and comparing their efficiency. • Developing and applying sorting algorithms (bubble, merge, insertion) to organize data.	• Abstraction • Decomposition • Algorithmic Thinking • Pseudocode • Linear Search • Binary Search • Bubble Sort • Merge Sort • Insertion Sort • Trace Table	**Written Assessment –** End of topic test with MCQ on MS Forms

Curriculum Plans: Year 10 (Computer Science)

			• Using trace tables to track variable changes and detect errors in algorithms. • - Correcting and refining algorithms to ensure efficiency and accuracy.		
Term 1	Producing Robust Programs	• Defensive design considerations: anticipating misuse, input validation, and authentication. • How input validation ensures robust programs and prevents errors. • Methods of user authentication (e.g., passwords, CAPTCHA) and their importance in security. • Maintainability principles: sub-programs, naming conventions, indentation, and commenting. • The purpose and types of testing: iterative vs. terminal testing. • Identifying syntax and logic errors in code. • The use of test data categories: normal, boundary, invalid, and erroneous data.	• Implementing input validation to ensure data is within acceptable parameters (e.g., whitelists, length checks). • Anticipating misuse and applying exception handling to handle unexpected inputs or user behavior. • Writing authentication mechanisms to verify users and protect against misuse (e.g., strong password validation). • Using clear naming conventions, indentation, and comments to improve the readability and maintainability of code. • Applying iterative and final testing techniques	• Input Validation • Authentication • Exception Handling • Sub-programs • Iterative Testing • Syntax Error • Logic Error • Boundary Test Data	**Practical Task –** Programming

		• - Refining algorithms to improve robustness and error handling.	to identify and fix errors throughout the development cycle. • Differentiating between syntax and logic errors and applying fixes to ensure program correctness. • Creating test cases using different types of test data to ensure program reliability and robustness. • - Refining and improving algorithms for robustness and reliability through error detection and testing.		
Term 1	Boolean Logic	• Understanding Boolean logic operators: AND, OR, and NOT. • How to represent logic gates in diagrams and interpret their outputs. • Using truth tables to represent and evaluate logic gate outputs. • Combining Boolean operators to two levels (e.g., (A OR B) AND C). • The application of Boolean logic in solving problems and decision-making.	• Constructing logic diagrams using the operators AND, OR, and NOT. • Creating and interpreting logic gate diagrams to model real-world scenarios. • Completing and interpreting truth tables for different combinations of Boolean expressions. • Applying Boolean operators to two levels	• AND • OR • NOT • Truth Table • Logic Gate • Boolean Expression	

Curriculum Plans: Year 10 (Computer Science)

		• Identifying how Boolean logic is used in digital circuits and computer systems.	in both logic diagrams and truth tables. • Solving problems using logic diagrams and truth tables with Boolean operators. • - Designing circuits and systems that use Boolean logic to control outputs based on inputs.		
Term 2	Programming Languages and IDEs	• Characteristics and purpose of different levels of programming languages (high-level and low-level). • The purpose of translators in programming (assemblers, compilers, interpreters). • Characteristics and functions of assemblers, compilers, and interpreters. • The relationship between machine code and assembly language. • Common tools and facilities available in an Integrated Development Environment (IDE). • The role of an IDE in writing, testing, and debugging programs.	• Differentiating between high-level and low-level programming languages based on characteristics like speed, efficiency, and ease of coding. • Writing and translating code using different types of translators (assemblers, compilers, and interpreters). • Explaining the role of assemblers, compilers, and interpreters and identifying their use cases. • Writing simple programs in both high-level and low-level languages to compare functionality.	• High-level Language • Low-level Language • Assembler • Compiler • Interpreter • Machine Code • Integrated Development Environment (IDE)	**Written Assessment –** Mock Exam Assessed in the end of

Curriculum Plans: Year 10 (Computer Science)

			• Using IDE features like code completion, error diagnostics, and debugging tools to improve code development efficiency. • Identifying and utilising common IDE features to write and debug programs. • Translating source code into machine code and explaining how translation affects execution speed and efficiency.		
Term 2	Programming Constructs	• The use of variables, constants, operators, inputs, outputs, and assignments in programming. • Understanding and using the three basic programming constructs: sequence, selection, and iteration (count- and condition-controlled loops). • Common arithmetic and Boolean operators, including +, -, *, /, AND, OR, and NOT. • Data types such as integer, real, Boolean, character, string, and casting between data types.	• Writing programs that use variables, constants, and different operators (arithmetic, comparison, Boolean). • Implementing control flow in programs using sequence, selection (if statements), and iteration (loops). • Using arithmetic and Boolean operators in expressions to perform calculations and logical comparisons. • Applying data types correctly and	• Variable • Constant • Operator • Sequence • Selection • Iteration • String Manipulation • File Handling • SQL (Structured Query Language) • Array (one-dimensional,	**Online Assessment –** MCQ on MS Forms

Curriculum Plans: Year 10 (Computer Science)

		• Basic string manipulation techniques (e.g., length, case conversion, substring). • Basic file handling operations: open, read, write, and close files. • The use of records to store data and SQL to search for data. • One- and two-dimensional arrays to store and manipulate data. • The use of sub-programs (functions and procedures) to produce structured code. • - Random number generation in programming.	performing type casting in programs. • Manipulating strings and using string functions to modify or extract information. • Writing programs that handle files, including reading from and writing to files. • Creating programs that use records for structured data storage and SQL queries to manipulate databases. • Using arrays in programs to store and process lists of data, including two-dimensional arrays. • Creating structured programs using functions and procedures to modularize code. • - Writing programs that use random numbers for simulations or game logic.	two-dimensional) • Function, Procedure • Random Number Generation	
Term 2	1.6 - Ethical, Legal, Cultural and	• Impacts of digital technology on society, including ethical, legal, cultural, environmental, and privacy issues.	• Evaluating the impact of digital technology on ethical, legal, and	• Ethical Issues • Data Protection Act 2018	**Online Assessment –** AI (TILF or SmartRevise)

Curriculum Plans: Year 10 (Computer Science)

	Environmental Impacts	• Key legislation relevant to computer science: Data Protection Act 2018, Computer Misuse Act 1990, Copyright Designs and Patents Act 1988. • Differences between open-source and proprietary software. • Ethical concerns around technology use (e.g., internet regulation, privacy, and the digital divide). • Environmental impacts of technology, including manufacturing, usage, and disposal of devices. - The importance of software licenses and their role in software distribution and development.	environmental concerns. • Applying legal knowledge to real-world scenarios involving data protection and computer misuse. • Differentiating between open-source and proprietary software in terms of benefits and limitations. • Discussing and forming arguments for and against topics such as internet regulation and privacy concerns. • Assessing environmental impacts of technology and suggesting improvements for sustainability. - Evaluating the use of open-source vs. proprietary software in different scenarios.	• Computer Misuse Act 1990 • Copyright, Designs, and Patents Act 1988 • Open-Source Software • Proprietary Software	
Term 3	Study Leave and Exam				