Year 13 OCR Computer Science A-Level	**Curriculum Intent:** The aims of this qualification are to enable learners to develop: • An understanding and ability to apply the fundamental principles and concepts of computer science, including: abstraction, decomposition, logic, algorithms and data representation • The ability to analyse problems in computational terms through practical experience of solving such problems, including writing programs to do so • The capacity to think creatively, innovatively, analytically, logically, and critically • The capacity to see relationships between different aspects of computer science		
Year 13 Computer systems Component 1	**Term 1:**	**Term 2:**	**Term 3**
Topic Titles (in order of delivery)	Paper 1: Data validation and exception handling OOP Reading / writing from a text and binary file NEA: Abstraction / Decomposition Paper 2: Stack Frame Recursion Static / Dynamic data structures Boolean Logic	Paper 1: Database NEA Development and testing Paper 2: Standard Algorithms – Binary Tree search / Dijkstra’s shortest path Regular Languages Context Free Languages – Backus-Naur Form Turing Machine and the Halting problem Data Compression Principles	Revision for Exam Completion of NEA
Key knowledge / Retrieval topics	Paper 1: How to use exception handling Use of aggregation / composition / polymorphism / overriding How to read and write from a binary and text file Paper 2: Describe when the stack frame is used, sub-routine calls Describe the process of recursion and how to use it. Differences between static and dynamic structures and their usage Boolean Logic • Logic gates and truth tables • Logic circuits for Boolean expression • half-adder / full adder • use of edge triggered D-type flip-flop as memory unit	Paper 1: Databases: • Be able to produce an Entity Relationship Diagram to describe a data model • Explain relational database • 3rd Normal form • SQL • Client Server databases Paper 2: Regular Languages • Finite State Machine • State transition diagrams • Mealy Machine • Maths for regular expressions • create regular expressions • Sets Context Free Languages – Backus-Naur Form • use	

		• why syntax can be checked using BNF or syntax diagrams Turing Machine and the Halting problem • know what a Turing machine is, and how they can be view as a single fixed program computer Data Compression Principles • Run length encoding • dictionary based	
Understanding / Sequence of delivery	Paper 1: Try – Catch – Finally blocks, when to use. Demonstrate and explain how OOP supports core concepts and improves programming techniques and maintainability Paper 2: Content of stack frame, return addresses Explain recursive techniques, situations when recursion is more useful than iteration Data Structures: Hash table. dictionary Boolean Logic • Logic gates and truth tables • Logic circuits for Boolean expression • half-adder / full adder use of edge triggered D-type flip-flop as memory unit	Paper 1: Paper 2: Be able to and use Regular Languages • Finite State Machine • State transition diagrams • Mealy Machine • Maths for regular expressions • Sets ○ Subset / proper subset / countable ○ Set operations Context Free Languages – Backus-Naur Form • use • why syntax can be checked using BNF or syntax diagrams Turing Machine and the Halting problem • states • state transition • alphabet • sensing / writing head • transition rules	
Assessments	NEA Preparation Programming Homework	PPE 1 Programming Homework	A-level exams

Year 13 OCR Computer Science A-Level	**Curriculum Intent:** The aims of this qualification are to enable learners to develop: • An understanding and ability to apply the fundamental principles and concepts of computer science, including: abstraction, decomposition, logic, algorithms and data representation • The ability to analyse problems in computational terms through practical experience of solving such problems, including writing programs to do so • The capacity to think creatively, innovatively, analytically, logically, and critically • The capacity to see relationships between different aspects of computer science		
Year 13 Algorithms and programming Component 02	**Term 1:**	**Term 2:**	**Term 3**
Topic Titles (in order of delivery)	Elements of computational thinking Problem solving and programming	Programming techniques Computational methods Algorithms	Algorithms
Key knowledge / Retrieval topics	Understand what is meant by computational thinking How computers can be used to solve problems and programs can be written to solve them	The use of algorithms to describe problems and standard algorithms	The use of algorithms to describe problems and standard algorithms
Understanding / Sequence of delivery	Thinking abstractly Thinking ahead Thinking procedurally Thinking logically Thinking concurrently	Algorithms (a) Analysis and design of algorithms for a given situation. (b) The suitability of different algorithms for a given task and data set, in terms of execution time and space. (c) Measures and methods to determine the efficiency of different algorithms, Big O notation (constant, linear, polynomial, exponential and logarithmic complexity).	Algorithms (d) Comparison of the complexity of algorithms. (e) Algorithms for the main data structures, (stacks, queues, trees, linked lists, depth-first (post-order) and breadth-first traversal of trees). (f) Standard algorithms (bubble sort, insertion sort, merge sort, quick sort, Dijkstra's shortest path algorithm, A* algorithm, binary search and linear search)
Assessments			