Year 12 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 12 Computer systems Component 1	**Term 1:**	**Term 2:**	**Term 3**
Topic Titles (in order of delivery)	Paper 1: Computer Systems The characteristics of contemporary processors, input, output and storage devices Software and software development Paper 2: Programming Computational Thinking • Thinking abstractly • Thinking ahead • Think procedurally • Thinking logically / concurrently Fundamentals of Python • Use of IDE (Python) • Different data types • Basic Console programs • Commenting and debugging	Paper 1: Computer Systems Exchanging data Paper 2 Programming: Data types, data structures and algorithms Programming paradigms • Procedural • Declarative • Object Oriented • Event Driven	Paper 1: Computer Systems Legal, moral, cultural, and ethical issues Standard Algorithms • Linear Search • Binary Search • Binary Tree • Bubble and Insertion Sort • Merge and quick sort • Optimisation – Dijkstra / A* algorithms Paper 2: Programming Functions Analyse and define algorithms to solve problems NEA: Project Proposals Analysis
Key knowledge / Retrieval topics	Paper 1: Computer Systems Components of a computer and their uses (a) The Arithmetic and Logic Unit; ALU, Control Unit and Registers (Program Counter; PC, Accumulator; ACC, Memory Address Register; MAR, Memory Data Register; MDR, Current Instruction Register; CIR). Buses: data, address and control: how this relates to assembly language programs. (b) The Fetch-Decode-Execute Cycle; including its effects on registers. (c) The factors affecting the performance of the CPU: • clock speed • number of cores	Paper 1: Computer Systems How data is exchanged between different systems (a) Lossy vs Lossless compression. (b) Run length encoding and dictionary coding for lossless compression. (c) Symmetric and asymmetric encryption. (d) Different uses of hashing. Networks: a) Characteristics of networks and the importance of protocols and standards. (b) The internet structure: • The TCP/IP Stack. • DNS • Protocol layering. • LANs and WANs.	Paper 1: Computer Systems The individual moral, social, ethical, and cultural opportunities, and risks of digital technology. Legislation surrounding the use of computers and ethical issues that can or may in the future arise from the use of computers. • Computers in the workforce. • Automated decision making. • Artificial intelligence. • Environmental effects. • Censorship and the Internet. • Monitor behaviour. • Analyse personal information. • Piracy and offensive communications.

	• cache. (d) The use of pipelining in a processor to improve efficiency. (e) Von Neumann, Harvard and contemporary processor architecture Types of software and the different methodologies used to develop software Paper 2: Programming Variables Operators Basic C# instructions Sequence / Selection / Iteration	• Packet and circuit switching. (c) Network security and threats, use of firewalls, proxies and encryption. (d) Network hardware. (e) Client-server and peer to peer. Paper 2: Programming How data is represented and stored within different structures. Different algorithms that can be applied to these structures • Arrays / Tuples / records • Queues • Lists / Linked Lists • Stacks • Hash tables • Graphs • Trees	• Layout, colour paradigms and character sets. Standard algorithms • Recognise, use, and trace each algorithm • Explain usage Paper 2: Programming Be able to analyse a problem Be able to develop algorithms to solve specific problems Be able to compare different algorithms in terms of Big O: • Time efficiency • Space efficiency NEA: Completion of project proposal Completion of Analysis
Understanding / Sequence of delivery	Paper 1: Computer Systems Structure and function of the processor Types of processor Input, output and storage Systems Software Applications Generation Software Development Types of Programming Language Paper 2: Programming Abstraction (a) The nature of abstraction. (b) The need for abstraction. (c) The differences between an abstraction and reality. (d) Devise an abstract model for a variety of situations. Thinking ahead (a) Identify the inputs and outputs for a given situation. (b) Determine the preconditions for devising a solution to a problem. (c) The nature, benefits and drawbacks of caching. (d) The need for reusable program components. Think procedurally (a) Identify the components of a problem.	Paper 1: Computer Systems Compression, Encryption and Hashing Databases Networks Web Technologies Boolean Algebra Paper 2: Programming Subroutines and recursion Data Types Data Structures • Arrays / Tuples / records • Queues • Lists / Linked Lists • Stacks • Hash tables • Graphs • Trees	Paper 1: Computer Systems Computing related legislation Moral and ethical Issues Paper 2: Programming Analysis of problems Design of algorithms to solve a problem Comparing algorithms Functions Big O notation of functions • Constant time • Linear • Polynomial • Exponential • Logarithmic NEA: Requirements of the NEA Requirements of a project proposal Requirements of Analysis

	(b) Identify the components of a solution to a problem. (c) Determine the order of the steps needed to solve a problem. (d) Identify sub-procedures necessary to solve a problem. Thinking logically (a) Identify the points in a solution where a decision has to be taken. (b) Determine the logical conditions that affect the outcome of a decision. (c) Determine how decisions affect flow through a program. Thinking concurrently (a) Determine the parts of a problem that can be tackled at the same time. (b) Outline the benefits and trade offs that might result from concurrent processing in a particular situation.		
Assessments	Knowledge check Programming Homework	PPE 1 Programming Homework	Year 12 PPE NEA Approved project proposal and analysis

Year 12 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 12 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			