Curriculum Map Year 13 GCSE COMPUTER SCIENCE

Topic Name	**Term**	**Skills** developed with link to NC Subject content	Reflection on previous link in the curriculum	Progress to future link in the curriculum
Boolean algebra	*Autumn HT1*	• Construct a truth table for a variety of logic gates • Be familiar with drawing and interpreting logic gate circuit diagrams involving multiple gates • Complete a truth table for a given logic gate circuit • Write a Boolean expression for a given logic gate circuit • Draw an equivalent logic gate circuit for a given Boolean expression • Be familiar with the use of Boolean identities and De Morgan's laws to manipulate and simplify Boolean expressions • Write a Boolean expression for a given logic gate circuit, and vice versa • Understand the correspondence between a truth table and a Karnaugh map • Understand how to fill out a Karnaugh map for a given expression • Understand how to group items in a Karnaugh map • Interpret the groupings in a Karnaugh map • Simplify Boolean expressions with two, three or four variables using a Karnaugh map • Recognise and trace the logic of the circuits of a half adder and a full adder • Construct the circuit for a half adder • Understand the logic associated with D type flip flops • apply their knowledge in answers to a range of questions • be able to highlight areas of strength and any gaps in their understanding of computers	*Year 11: Boolean logic*	*Examinations and revision* *Year 13: Computational thinking* *Year 13: Algorithms*
Computational thinking	*Autumn HT1*	• Understand the nature of and need for abstraction • Describe the differences between an abstraction and reality • Devise an abstract model for a variety of situations • Identify the inputs and outputs for a given situation • Determine the preconditions for devising a solution to a problem • Understand the need for reusable program components • Understand the nature, benefits and drawbacks of caching • Identify the components of a problem • Identify the components of a solution to a problem • Determine the order of the steps needed to solve a problem • Identify sub-procedures necessary to solve a problem • Identify the points where a decision has to be taken • Determine the logical conditions that affect the outcome of a decision • Determine how decisions affect program flow • Determine which parts of a program can be tackled at the same time • Determine the benefits and trade-offs that might result from concurrent processing in a particular situation • Explore different strategies for problem-solving • Understand the concept and application of the "divide and conquer" approach • Describe features that make a problem solvable by computational methods	*Year 12: Data structures* *Year 13: Boolean algebra*	*Year 13: Algorithms* *Year 13: NEA Design, implantation, evaluation* *Year 13: Programming techniques*

		• Learn about and apply
		  • backtracking
		  • data mining
		  • heuristics
		  • performance modelling
		  • pipelining
		  • visualisation to solve problems		

NEA: Design	*Autumn HT1*	• Decompose the problem (a) Break down the problem into smaller parts suitable for computational solutions justifying any decisions made. • Describe the solution (a) Explain and justify the structure of the solution. (b) Describe the parts of the solution using algorithms justifying how these algorithms form a complete solution to the problem. (c) Describe usability features to be included in the solution. (d) Identify key variables / data structures / classes justifying choices and any necessary validation. • Describe the approach to testing (a) Identify the test data to be used during the iterative development and post development phases and justify the choice of this test data.	*Year 12: Software development* *Year 12: programming* *Year 12: NEA Analysis* *Year 13: Computational thinking*	*Year 13: NEA Implementation*
Legal, moral, ethical and cultural issues	*Autumn HT2*	• To be aware of computing related legislation, including: • The Data Protection Act 1998 • The Computer Misuse Act 1990 • The Copyright Design and Patents Act 1988 • The Regulation of Investigatory Powers Act 2000 • To understand that developments in digital technologies have enabled massive transformations in the capacity of organisations to monitor behaviour, amass and analyse personal information • Discuss the individual (moral), social (ethical) and cultural opportunities and risks of digital technology, including: • computers in the workforce • automated decision making • artificial intelligence • analysis of personal information • Discuss the environmental effects of computers • Discuss the cultural opportunities and risks of digital technology relating to: • censorship and the Internet • the monitoring of behaviour • piracy and offensive communications • layout, colour paradigms and character sets	*Year 10: Ethical, legal and environmental impacts*	*Examinations and revision*
Programming techniques	*Autumn HT2 and Spring HT3*	• Be familiar with the use of an IDE to develop and debug a program • Define what is meant by an algorithm and pseudocode • Learn how and when different data types are used • Learn the basic arithmetic operations available in a typical programming language • Write pseudocode solutions to simple problems • Use relational operators • Use Boolean operations AND, OR, NOT • Use the switch/case statement for selection • Use nested selection statements • Understand and use three different types of iterative statement: ○ while … endwhile	*Year 11: Python programming examination work* *Year 11: Relational databases and SQL* *Year 12: Software development* *Year 12: Data types* *Year 12: Data structures* *Year 12: programming* *Year 13: Computational thinking*	*Examinations and revision* *Year 13: Algorithms*

		○ do (or repeat) … until ○ for … next • Be familiar with subroutines, their uses and advantages • Use subroutines that return values to the calling routine • Use arguments/parameters to pass data to subroutines by value and by reference • Contrast the use of local and global variables • Use recursion to solve simple problems • Trace a recursive algorithm • Compare recursion to an iterative approach • Describe the features of an object oriented language: • classes, objects, methods, attributes, inheritance, encapsulation and polymorphism • Write pseudocode for a class definition • Write pseudocode to instantiate an object and use its methods • Draw inheritance diagrams • Describe the advantages of an object oriented approach to programming		
NEA: Development	*Autumn HT2 and Spring HT3*	• Iterative development process (a) Provide annotated evidence of each stage of the iterative development process justifying any decision made. (b) Provide annotated evidence of prototype solutions justifying any decision made. • Testing to inform development (a) Provide annotated evidence for testing at each stage justifying the reason for the test. (b) Provide annotated evidence of any remedial actions taken justifying the decision made.	*Year 12: Software development* *Year 12: programming* *Year 13: NEA Analysis, design*	*Year 13: NEA design, implementation* *Examinations and revision*
Algorithms	*Spring HT4*	• Analyse the suitability of different algorithms for a given task and data set • Be familiar with measures and methods to determine the efficiency of different algorithms • Define constant, linear, polynomial, exponential and logarithmic functions • Use Big-O notation to compare the time complexity of algorithms • Be able to derive the time complexity of an algorithm • Write and trace algorithms for linear search and binary search • Analyse the time complexity of the linear search and binary search algorithms • Describe and trace the binary tree search algorithm • Be able to describe the bubble sort and insertion sort algorithms • Be able to trace the bubble sort and insertion sort algorithms • Understand and be able to trace the merge sort and quick sort algorithms • Be able to trace depth-first and breadth-first algorithms • Describe typical applications of each • Understand and be able to trace Dijkstra's shortest path algorithm • Be aware of applications of the shortest path algorithm • Describe the A* algorithm	*Year 11: Python programming examination work* *Year 11: Algorithms* *Year 11: Boolean logic* *Year 12: Software development* *Year 12 Data types* *Year 12: Data structures* *Year 12: programming* *Year 13: Programming techniques*	*Examinations and revision*
NEA: Evaluation	*Spring HT4*	• Testing to inform evaluation (a) Provide annotated evidence of testing the solution of robustness at the end of the development process. (b) Provide annotated evidence of usability testing (user feedback).	*Year 11: Algorithms* *Year 12: Data types* *Year 12: Software development* *Year 12: programming* *Year 13: Analysis, design, implementation*	

		• Success of the solution (a) Use the test evidence from the development and post development process to evaluate the solution against the success criteria from the analysis. • Describe the final product (a) Provide annotated evidence of the usability features from the design, commenting on their effectiveness. • Maintenance and development (a) Discuss the maintainability of the solution. (b) Discuss potential further development of the solution.		