HGS Curriculum Map Key Stage 4

Year Group: 11 Computer Science GCSE - Autumn

Time period	Autumn 1	Autumn 2
Topics/ sub-topics	**Algorithms**	**Logic and Languages**
Purpose	This is a theoretical unit covering Section 2.1 of the OCR GCSE (9-1) Computer Science specification. The first lesson introduces the concepts of computational thinking; abstraction, decomposition and algorithmic thinking. Lessons on searching and sorting algorithms are followed by those focusing on developing algorithms using flow diagrams and pseudocode.	The unit covers 2.3, 2.4 and 2.5 of the OCR GCSE (9-1) specification. It begins with Boolean logic diagrams and truth tables. Following this, students practise writing and coding algorithms, which incorporate defensive design. Different types of error, and how to detect them in a program code, is described, along with testing and creating a test plan. The unit concludes by looking at the classification of programming languages and the different types of translator used with high- and low-level languages.
Crucial Learning	**Students should be able to:** • state what is meant by an algorithm • state what is meant by abstraction • state what is meant by decomposition • state the sequence in which items in a sorted list will be examined in a linear and binary search • state the advantages and disadvantages of a linear and binary search • state an advantage of the merge sort and insertion sort over the bubble sort • show the state of a list after the first pass in a bubble sort • use a flowchart or pseudocode to define the steps in a simple algorithm • trace through a simple flow diagram or pseudocode algorithm to determine the output **Most students will be able to:** • explain how abstraction is used in a given scenario • explain how decomposition may be used in an algorithm for a given problem • explain how a binary search works • explain how a bubble sort works • show the state of a list at a given point in a bubble sort, merge sort or insertion sort • interpret, correct or complete a short algorithm **Some students will be able to:**	**Students should be able to:** • Recognise standard symbols used to represent NOT, AND, OR, NAND, NOR and XOR logic gates • Draw truth tables for the above logic gates • Describe some simple validation checks that can be applied to data • Select test data that covers normal (typical), boundary (extreme) and erroneous data • Complete a trace table to trace through a simple algorithm • Give examples of high-level and low-level languages • Give advantages of high-level languages over low-level languages • Explain the differences between a compiler, interpreter and assembler **Most students will be able to:** • Recognise a logic gate from its truth table • Draw a logic circuit to solve a given problem • Detect and correct errors in simple algorithms • Use a trace table to find errors or determine the purpose of an algorithm • Be able to justify the choice of test data • Give examples and reasons of when it might be appropriate to use a low-level language • Give examples of when it would be appropriate to use a compiler and interpreter **Some students will be able to:**

	• use pseudocode to define the steps in a complex algorithm • explain how a merge sort and an insertion sort work • correct or complete a complex algorithm	• Draw a logic circuit to implement a given written logic statement • Write more complex authentication routines • Write robust programs that apply checks to data entered by the user
Sequence	**Prior Knowledge** • In Yr7 students complete a short module 'How Computers Work'. In this unit they are introduced to simple searching and sorting algorithms.	**Prior Knowledge** • At KS3 pupils should have gained some experience of programming and creating algorithms. They should also be familiar with Booleans, Boolean expressions and Boolean operators.
	Future Learning: A-Level Computer Science (AQA): - 4.3 Fundamentals of algorithms	**Future Learning** A-Level Computer Science (AQA): - 4.6.2 Classification of programming languages - 4.6.3 Types of program translator - 4.6.4 Logic gates - 4.6.5 Boolean Algebra
Skills Acquired	**Computational Thinking Skills:** • Decomposition - breaking down complex problem or system into smaller, more manageable parts • Pattern recognition - looking for similarities among and within problems • Abstraction - focusing on the important information only, ignoring irrelevant detail • Algorithms - developing a step-by-step solution to the problem, or the rules to follow to solve the problem	**Computational Thinking Skills:** • Decomposition - breaking down complex problem or system into smaller, more manageable parts • Pattern recognition - looking for similarities among and within problems • Abstraction - focusing on the important information only, ignoring irrelevant detail • Algorithms - developing a step-by-step solution to the problem, or the rules to follow to solve the problem

Assessment: **Formative & summative**	**Assessment** **Verbal Feedback**: Regular use of peer, self and teacher feedback **Written Feedback:** Individual feedback of home learning assessments in the Showbie 'Marking and Feedback' folder. Students have dedicated improvement and reflection time at the start of each lesson. **Learning Grids:** - 2.1 Algorithms **Topic Tests:** - 2.1 Computational Thinking, Algorithms and Programming **SIMS:** - AUT 1 OGCU	**Assessment** **Verbal Feedback**: Regular use of peer, self and teacher feedback **Written Feedback:** Individual feedback of home learning assessments in the Showbie 'Marking and Feedback' folder. Students have dedicated improvement and reflection time at the start of each lesson. **Learning Grids:** - 2.3 Producing Robust Programs - 2.4 Computational Logic - 2.5 Translators **Topic Tests:** - 2.3 Robust Programs - 2.4 Computational Logic, Translation and Editors **SIMS:** - AUT 2 OGCU