Computer Science Unit Overview Year 13

Rationale for overall teaching order
Work for paper 1 (Section 1-4 from specification) is covered earlier as it is most likely to be useful for students' NEAs or for dealing with the skeleton program.
Work on NEA is completed in dedicated weekly sessions to ensure teacher overview of projects.
Work on skeleton program is completed in dedicated fortnightly sessions to ensure deep familiarity with this.

(2.3) – numbers/sections refer to specification
(https://filestore.aqa.org.uk/resources/computing/specifications/AQA-7516-7517-SP-2015.PDF)

Extension links
Computerphile - https://www.youtube.com/channel/UC9-y-6csu5WGm29I7JiwpnA
Programming tutorials - https://www.w3schools.com/

Section 10 not covered as usual, and online work showed Section 1 will need recapping. However, a significant amount of prep work for the NEA was completed.
Teacher 1 (8 lessons/fortnight) will re-cover Section 1 and then continue with the course as normal.
Teacher 2 (2 lessons/fortnight) will cover Section 10 and deal with working on the skeleton program.

Computer Science – Year 13 Autumn 1				
What are we learning?	**What knowledge, understanding and skills will we gain?**[1]	**What does mastery look like?**[2]	**How does this build on prior learning?**[3]	**What additional resources are available?**
4.2 Data structures NEA - Analysis & Design Skeleton program - section C practice Also: Recap of Section 1 covered during lockdown	Knowledge: Know the key terms: queue, stack, list, graph (including weighted, directed, and undirected versions), vertex/node, edge/arc, tree, dictionary, hash table, vectors, adjacency matrix and adjacency lists. Relational databases and entity-relationship (ER) diagrams. Understanding: Be able to distinguish between static and dynamic data structure and compare their uses. Compare use	Be able to recognise and manipulate the various structures in a wide variety of representations (eg horizontal or vertical tables, using arrays, standard graph notation) Be able to select the appropriate structure for a given situation and explain their choice.	Much of this can be considered to build on arrays used extensively in Year 12 (eg implementing a binary tree as a 2D array or three 1D arrays). Students taking Further Maths will be familiar with the concepts of graphs from D1 covered in Year 12 Spring 2.	Textbook p50-91 Resources – outline PowerPoints with suggested examples and scaffolding activities Practice questions (from past exams) and section assessments Python files include: Stackclass with pointer Network

Teacher 2 will cover Section 10 – Databases and SQL and cover the work on the skeleton program	of adjacent list and matrices and identify when each is more appropriate. Apply simple hashing algorithms and explain how collisions are dealt with. Use of dot product in vectors. Why databases are normalised. Why concurrent access can cause issues and how to they can be prevented/resolved. Skills: Programming simple implementations of stacks, queues, binary trees and graphs. Programming with dictionaries. Produce a data model from given data requirements for a simple scenario involving multiple entities.	Be able to read and manipulate programs implementing simple versions of the structures, tracing results into a table. Be able to compare ideas (eg methods to control concurrent access) in depth, giving advantages and disadvantages of each Confidently draw ER diagrams showing the relationships between entities Be able to interpret and write commands in SQL	This can also be used to consolidate Year12 Summer 2 work on classes as example implementations can utilise class structures. Students may have an awareness of databases from GCSE Computing (or similar) and may even have used Access. However most of the terminology required here will be new to students, and most will not have used SQL.	BinTree with arrays Dictionaries and hashing Current year's skeleton program and adjusted previous practice questions (created as used) Teacher 2 will use: Textbook p364-393 For practising SQL: https://www.w3schools.com/ Python files: - Using sqlite - Using sqlite advanced

Computer Science - Year 13 Autumn 2				
What are we learning?	**What knowledge, understanding and skills will we gain?[1]**	**What does mastery look like?[2]**	**How does this build on prior learning?[3]**	**What additional resources are available?**
4.3 Fundamentals of Algorithms 4.4 Theory of Computation NEA - Implementation; Skeleton program - section D practice Teacher 2 will finish Section 10 and continue working with the skeleton program.	Knowledge: Breadth-first and Depth-first graph traversal algorithms. Pre-, post-, and in-order tree traversal algorithms. Reverse Polish notation (RPN). Linear, binary, and binary tree searching algorithms. Bubble sort and merge sort. Dijkstra's algorithm. Idea of abstraction. Set notation. Regular expressions. Backus-Naur (BN). Time complexity and big O notation. Tractable/intractable problems. Turing machines. Mealy machines. Understanding**:** Describe typical uses of graph-traversal and tree-traversal algorithms. Understand where and why RPN is used. Analyse the time complexity of searching algorithms. Be able to link FSM with regular expressions. Understand which BN expressions cannot be represented by regular languages. Understand the importance of the idea of a Universal Turing machine and the Halting problem. Skills: Be able to draw a binary search tree from given information. Be able to trace algorithms written in psuedocode for any of the above situations. Programming adjustments to the skeleton program.	Students should be able to apply their knowledge of the base algorithms to a variety of contexts and formats (past exam questions are a good source of practice for this). Although not strictly required by the specification, a high-performing student could be able to write programs to perform the various algorithms in a variety of contents (eg using arrays or class structures) Students should be able to adjust the skeleton program to account for error handling or minor functionality changes.	This builds directly on Year 13 Autumn 1 by using the graphs introduced there and applying algorithms to them. Mealy machines build on FSM covered in Year 12 Spring 2. The idea of tracing algorithms (in trace-tables) is used throughout the course but often found difficult by students. These topics provide ample opportunity for strengthen that skill. Maths students will be familiar with the basics of set notation but probably not subsets. Also the notation for binary using non-standard maths notation. Further Maths students may be familiar with Bubble sort and Dijkstra's algorithm if they have studied D1, although the style of exam question (often using pseudocode) is very different in Computer Science.	Textbook p92-181 Resources – outline Powerpoints with suggested examples and scaffolding activities Practice questions (from past exams) and section assessments Python files include: Graph traversal Tree traversal Bubble and merge sort Network with Dijkstra Current year's skeleton program and check the AQA wikibooks page for suggested changes to the program.

Computer Science - Year 13 Spring 1

What are we learning?	What knowledge, understanding and skills will we gain?[1]	What does mastery look like?[2]	How does this build on prior learning?[3]	What additional resources are available?
4.5-4.9 A level additions to AS content (especially 4.9 The Internet) NEA - Implementation Skeleton program - further practice Teacher 2 will focus on working on the skeleton program and oversee finishing the NEA	Knowledge: Floating point binary and absolute/relative error. Key features of vector graphics. Adder and half-adder logic circuits. D-type flip-flops. Interrupts in the fetch-execute cycle. The internet and how it works. Internet security. The TCP/IP protocol. Client-server model and thin- vs thick-client computing. CRUD applications and REST. Recognise JSON and XML. IP address structure and subnet masking. DHCP and NAT. Understanding: Compare advantages and typical uses of fixed point binary to floating point binary. Understand why floating point binary is normalised. Explain the use and placement of a D-type flip-flop in a logic circuit. Understand the use of stacks in interrupts. Explain the purposes of the four layers in the TCP/IP protocol. Compare JSON with XML Skills: Be able to read, write and convert between various number forms (decimal, floating point binary etc). Be construct a half-adder logic circuit, and recognise a full-adder circuit.	Students are confident in using the various forms of binary. Students will know and be able to justify the time complexities of the standard known algorithms. Additionally, they will be able to work out the time complexities of non-standard algorithms described (eg in pseudocode).	Section 4.5-4.7 build directly upon those sections from Year 12, and some time can be taken to revise the Y12 work (eg looking at all logic circuits when introducing the new adder circuits) Section 4.8 has nothing new from Year 12 this is a good time to review and look at a new case study. Section 4.9 is mostly new content also some terms (eg IP, virus) maybe be familiar from GCSE or general use.	Textbook p134-363 (A level sections only) Resources – outline Powerpoints with suggested examples and scaffolding activities Practice questions (from past exams) and section assessments Current year's skeleton program – opportunity to use student ideas in questions/potential changes
Computer Science - Year 13 Spring 2				

What are we learning?	What knowledge, understanding and skills will we gain?[1]	What does mastery look like?[2]	How does this build on prior learning?[3]	What additional resources are available?
4.11 Big Data 4.12 Functional Programming NEA - Testing & Evaluation (finish) Teacher 2 will focus on working on the skeleton program and oversee finishing the NEA	Knowledge: Know that big data is a catch-all term for data that won't fit the usual containers. Big data can be described in terms of volume, velocity and variety. Functional programming terminology: function type, first-class object, function application, partial function applications, composition of functions, higher order functions. Be familiar with map, filter, reduce/fold and typical list processing with head: [tail]. Understanding: Processing of big data can be distributed across several servers. Functional programming is a solution because of immutable data structures. Understand the functional programming paradigm. Skills: Use a fact based model for representing data and graph schema for capturing the structure of a dataset Time should also be given to finishing the NEA.	Students should be able to identify the features that make data 'big' in any content. Students should be able to follow and add to graph schema from descriptions. Although not strictly required by the specification a high-performing student might practice writing programs using a functional programming paradigm (Python file FP6 provides an opportunity to do this)	The idea of big data is common in Section 8 style questions and this could be used as another opportunity to look as a case study for that section. Students are likely to have used functions (with return) in other programs, and this can help them understand the Functional programming paradigm.	Textbook p382-408 Resources – outline Powerpoints with suggested examples and scaffolding activities Practice questions (from past exams) and section assessments Six Python files (FP1 to FP6) introducing students to concepts of Functional Programming and using it in Python. Current year's skeleton program

What are we learning?	What knowledge, understanding and skills will we gain?[1]	What does mastery look like?[2]	How does this build on prior learning?[3]	What additional resources are available?
Guided revision on targeted topics (eg Section 10 databases, reviewing skeleton program)	Knowledge: revisiting and reconsolidating all knowledge from the 2 year course Understanding: building a deeper understanding of the course through regular review and practice Skills: particular focus on ensuring exam technique is secure	Students can confidently tackle questions on a range of topics. Work is presented clearly in logical steps. When faced with an unusual or difficult context students are not afraid to try several approaches to find a correct solution .	This is all about consolidating and applying prior learning.	Past papers Revision PowerPoints and questions banks to help revise specific topics. Current resources include topics: Data structures TCP/IP Assembly code Buses, stored program Digital and analogue Longer answer questions Sound revision Digital camera and RFID Section 4 review Section 10 review Final checklist Final paper 2 revision Excel spreadheet of past papers for students to record performance and identify areas for improvement

Computer Science - Year 12 Summer 2				
What are we learning?	**What knowledge, understanding and skills will we gain?**	**What does mastery look like?**	**How does this build on prior learning?**	**What additional resources are available?**
(revision of requested topics although exam usually quite early)	Knowledge: revisiting and reconsolidating all knowledge from the 2 year course Understanding: building a deeper understanding of the course through regular review and practice Skills: particular focus on ensuring exam technique is secure	Students can confidently tackle questions on a range of topics. Work is presented clearly in logical steps. When faced with an unusual or difficult context students are not afraid to try several approaches to find a correct solution .	This is all about consolidating and applying prior learning.	Past papers Revision PowerPoints and questions banks to help revise specific topics. Current resources include topics: Data structures TCP/IP Assembly code Buses, stored program Digital and analogue Longer answer questions Sound revision Digital camera and RFID Section 4 review Section 10 review Final checklist Final paper 2 revision Excel spreadheet of past papers for students to record performance and identify areas for improvement.