Computer Science Unit Overview Year 12

Rationale for splitting the specification between two teachers

The course very naturally falls into two parts being tested through a more practical Paper 1 and more theory-based paper 2. Although there are activities to supplement and support both parts of the course (especially where there is overlap anyway) the split enable one teacher to really focus and develop the students' programming skills whilst the other focuses on developing written techniques.

Teacher 1 (Sections 1-4, more practical, paper 1)

Teacher2 (Section 5-9, more theory, paper 2)

2.3 – numbers/sections refer to specification

(https://filestore.aqa.org.uk/resources/computing/specifications/AQA-7516-7517-SP-2015.PDF)

Extension links

Programming tutorials - https://www.w3schools.com/

Computerphile - https://www.youtube.com/channel/UC9-y-6csu5WGm29I7JiwpnA

Since we accept students who have not studied GCSE Computer Science, no specific catch up is needed. However, we understand that programming skills may be less practised than usual: additional teaching time is to be used primarily to focus on developing these skills.

Computer Science – Year 12 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?**
Teacher 1 3.1 Fundamentals of programming	Knowledge: Data types (integer, real/float, Boolean, character, string, date/time, records, arrays). Programming concepts (variable declaration, constant declaration, assignment, iteration, selection, subroutine (inc. parameters, returning values and global/local variables)). Arithmetic operations (inc integer division, exponentiation, rounding, truncation). Relational operations (equal to, not equal to, less than, greater than, less than or equal to, greater than or equal to). Boolean operations (NOT, AND, OR, XOR). String-handling operations. Random number generation. Structured programming Understanding: Students will understand how the above items can be used together to create programs that solve problems. They will understand how to develop subroutines that pass parameters and return values . Understand the structured approach to program design and construction and be able to construct and use hierarchy charts when designing programs. Skills: Developing and formatting readable code	When given a problem students should able to make decisions about what programming aspects, combined, will be able to created and effective solution to a problem In particular students should: Be able to explain choices for coding decisions Read pre-written code and spot errors	This unit builds on KS4 learning of programming. It reinforces the learning from KS4 and ensures the students can apply programming elements to the language they will use at A-Level	Textbook p2-34 Resources – outline PowerP oints with suggested examples and scaffolding activities Practice questions (from past exams) and section assessments Practice skeleton program (as part of assessment) – Number Guess Many Python files available to demonstrate and practise the concepts here.

Teacher 2 3.5 Data representations	Knowledge: Number systems N, Z, Q, R. Bases 2, 10, 16. Units to tebi. Two's complement binary. ASCII and Unicode. Parity bits, majority voting, check digits. Bitmaps. Analogue/digital conversion of sound. MIDI. Run-length encoding and dictionary-based methods. Caesar and Vernam ciphers. Understanding: Why hexadecimal is used as a shorthand for binary. How the same byte could represent many different things. Why unicode was introduced. How an ADC works. The difference between lossless and lossy compression and the advantages of each. Skills: Covert between different number bases. Calculation involving binary (without converting to decimal)	In particular students should: Through regular revisiting, be able to quickly recall all key terms used in this section Be able to give examples and perform required conversions and calculations quickly Be able to describe processes in sufficient detail to answer longer written exam questions Begin to apply their knowledge to a range of contexts (eg in programming, and harder exam questions)	Much of this builds directly on work covered in GCSE Computing, however note that not all students will have done GCSE Computing. Students tend to know the gist of topics from GCSE (eg Run-length encoding) but usually struggling to answer questions in the required depth without clear modelling. Using the practice exam questions throughout helps students to see the standard required.	Textbook p182-228 Resources – outline Powerpoints with suggested examples and scaffolding activities Practice questions (from past exams) and section assessments Practical tasks: Converting bases Error checking Skeleton – Secret Messages

Computer Science - Year 12 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?**
Teacher 1 3.2 Data structures	Knowledge: Be familiar with the concept of data structures and arrays (single and multi-dimensional). How to read/write from a text file and a binary (non-text) file. Understanding: How a 2D array can be used to represent a table, matrix or grid. Skills: Developing and formatting readable code, including the use of arrays. Continuing to build programming skills through practical experiences, including the use of a previous skeleton program.	When given a problem students should able to make decisions about what programming aspects, combined, will be able to created and effective solution to a problem In particular students should: Be able to explain choices for coding decisions Read pre-written code and spot errors Be able to confidently use and manipulate arrays (single and multi-dimensional) when programming	This unit builds on KS4 learning of programming, and most students will be familiar with the concept of a 1D array at least. It reinforces the learning from KS4 and ensures the students can apply programming elements to the language they will use at A-Level	Textbook p50-55 (AS parts only) Resources – outline PowerPoints with suggested examples and scaffolding activities Practice questions (from past exams) and section assessments Skelton program – Battleships (as part of assessment) Many Python files available to demonstrate and practise the concepts here.

Teacher 2 3.6 Computer systems	Knowledge: Classification of hardware and software. Role of an OS. Classification of high- and low-level languages. Compilers and interpreters. Bytecode. Logic gates. Basics of Boolean algebra. Understanding: the need for, and attributes of, different types of software (OS, utility, library, translator). Advantages and disadvantages of high-and low-level languages. Explain the differences between compilers and interpreters and situations in which each is appropriate. Skills: Complete a truth table for a given logic circuit. Write a Boolean expression for a given logic gate (and vice versa). Use Boolean identities and De Morgan's laws to manipulate and simplify Boolean expressions.	In particular students should: Through regular revisiting, be able to quickly recall all key terms used in this section and give examples of each where appropriate. Be able to compare ideas (eg high and low level, or compilers and interpreters) in depth, giving advantages and disadvantages of each Confidently use logic gate circuits and Boolean expressions and truth tables Show logical steps to simplify an expression using Boolean algebra Become more confident in applying their knowledge to a range of contexts (eg in programming, and harder exam questions)	Again, much of this builds directly on work covered in GCSE Computing, however note that not all students will have done GCSE Computing. Logic operators are used practically in Autumn1 (Teacher1). Students tend to know the gist of topics from GCSE (eg classification of software) but usually struggle to answer questions in the required depth without clear modelling. Using the practice exam questions throughout helps students to see the standard required.	Textbook p230-264 Resources – outline PowerPoints with suggested examples and scaffolding activities Practice questions (from past exams) and section assessments Practical tasks: Bitwise operators

Computer Science - Year 12 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?**
Teacher 1 3.3 Systematic approach to problem solving	Knowledge: The five stages of software development: Analysis, Design, Implementation, Testing, Evaluation (ADITE) Understanding: The concepts of normal, erroneous and boundary data for testing. Skills: Continued development of programming skills, including the use of a new skeleton program (note that although this section can be tested explicitly in AS exams, it is examined through the NEA project for the full A level)	Students should be able to identify the stages of development (ADITE) in a given scenario. Students should be able to identify, and give examples of, normal, erroneous and boundary data for testing. When given a problem students should able to make decisions about what programming aspects, combined, will be able to created and effective solution to a problem In particular students should: Be able to explain choices for coding decisions Read pre-written code and spot errors Be able to confidently use and manipulate arrays (single and multi-dimensional) when programming	This unit builds on KS4 learning of programming and the previous terms's work on programming – adding new contexts for students to develop and extend their skills.	Textbook p408-416 Resources – outline PowerPoints with suggested examples and scaffolding activities Practice questions (from past exams) and section assessments Skelton program – Plant Growing Simulation (as part of assessment)

Teacher 2 3.7 Computer organisation and architecture	Knowledge: Basic internal components of a computer system (processor, main memory, buses). Harvard and von Neumann architectures. The stored program concept. The Fetch-Execute cycle and the role of registers within it. The processor instruction set. Immediate and direct addressing. Factors affecting processor performance. Assembly code operations. External hardware devices (barcode reader, digital camera, laser printer, RFID). Secondary storage devices (hard disk, optical disk, soli-state drive) Understanding: How the different bus widths affect system performance. The use of buses within the Fetch-Execute cycle. Compare and contrast the different external hardware for use in a given context. Skills: Be able to understand and write simple programs using the standard AQA assembly language.	In particular students should: Through regular revisiting, be able to quickly recall all key terms used in this section and give examples of each where appropriate. Be able to compare ideas (eg Harvard and von Neumann architectures) in depth, giving advantages and disadvantages of each Confidently use the standard AQA assembly language Become more confident in applying their knowledge to a range of contexts (eg in programming, and harder exam questions)	Again, much of this builds directly on work covered in GCSE Computing, however note that not all students will have done GCSE Computing. Students tend to know the gist of topics from GCSE (eg classification of software) but usually struggle to answer questions in the required depth without clear modelling. Using the practice exam questions throughout helps students to see the standard required.	Textbook p266-299 Resources – outline PowerPoints with suggested examples and scaffolding activities Practice questions (from past exams) and section assessments

Computer Science - Year 12 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?**
Teacher 1 3.4 Theory of computation	Knowledge: Definitions of: algorithm, abstraction (representational, by generalisation or categorisation, procedural, functional), information hiding, data abstractions, problem abstraction/reduction, decomposition, composition, automation Understanding: Develop Skills: Develop and check solutions to simple logic problems	Students will appreciate that computer science is about building clean abstract models (abstractions) of messy, noisy, real world objects or phenomena. Computer scientists have to choose what to include in models and what to discard, to determine the minimum amount of detail necessary to model in order to solve a given problem to the required degree of accuracy. Students will appreciate that computer science deals with putting the models into action to solve problems. This involves creating algorithms for performing actions on, and with, the data that has been modelled. When given a problem students should able to make decisions about what programming aspects, combined, will be able to created and effective solution to a problem	This unit builds on KS4 learning of programming and the previous terms's work on programming – adding new contexts for students to develop and extend their skills.	Textbook p134-149 (AS sections only) Resources – outline PowerPoints with suggested examples and scaffolding activities Practice questions (from past exams) and section assessments Skelton program – Morse Code (as part of assessment)

Teacher 2 3.8 Consequences of computing	Knowledge: Awareness of current individual (moral), social (ethical), legal and cultural opportunities and risks of computing. Awareness of some relevant laws (although specifics not required) such as the Data Protection Act and the Copyright, Designs and Patents Act. Understanding: Be able to categorise issues into the (sometimes overlapping) areas of individual, social, legal and cultural. Skills: Being able to analyse a given situation and apply typical issues (job creation/destructions, privacy, copyright, fault, access) to the specific situation being discussed. A standard approach is to 'brainstorm' the issues and then practice turning that into a longer written response.	In particular students should: Understand the (sometimes overlapping) scope of the headings: individual, social, legal, cultural. Be aware of current UK laws as pertaining to computing and the use of technology Show knowledge of current issues through wider reading Be able to write coherently on a given situation, highlighting both positives (opportunities) and negatives (risks)	Students have the opportunity to bring their own general knowledge and interests to this part of the course through situations that are discussed. Questions often involve longer written answers (around 10 marks) so writing skills (eg from English or History - especially considering all 'sides' of a situation) can be used and developed here.	Textbook p300-309 Recent news stories involving the development and use of technology (eg smart cities; use of AI/AR/VR in fields such as recruitment and medicine) Resources – outline PowerPoints with suggested examples and scaffolding activities. Practice questions (from past exams) and section assessments Black Mirror (tv series)

Computer Science - Year 12 Summer 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?**
Teacher 1 Skeleton program preparation for mock (2019 Board Game)	Knowledge: consolidation and development of all knowledge from 'Teacher1' sections above Understanding: consolidation and development of all understanding from 'Teacher 1' sections above Skills: consolidation and development of all skills from 'Teacher 1' sections above	In particular students should: Through regular revisiting, be able to quickly recall all key terms used and give examples of each where appropriate. Be able to write short programs in Python from a given description, algorithm, flow chart or similar Be very familiar with the skeleton program – be able to identify the purpose of each subroutine and line within Be able to make adjustments to the skeleton program to include error correction or enhanced functionality	Students will have worked on skeleton programs for previous assessments.	Practice questions (from past exams) for section A Programming Challenges for end of Section A (available on Notebook) Practice theory questions on the skeleton program for Section B – PowerPoint with suggestions and examples Practice practical changes to the skeleton program for Section C (can also see AQA wikibooks page)

Teacher 2 3.9 Communication and Networking	Knowledge: Serial and parallel transmission. Synchronous and asynchronous transmission. Definitions of: baud rate, bit rate, bandwidth, latency, protocol. Physical star, logical bus topology. Peer-to-peer and client-server networking. Wireless networking concepts: WiFi, NIC, WAP, security issues, CSMA/CA, RTS/CTS, SSID Understanding: Difference between bit rate and baud rate, and the relationship to bandwidth. How a topologys physical layout can be different to it's logical functionality. Skills: Calculate bit rate from baud rate and bits per signal. Draw and identify bus and star networks.	In particular students should: Through regular revisiting, be able to quickly recall all key terms used in this section and give examples of each where appropriate. Be able to compare ideas (eg synchronous and asynchronous transmission) in depth, giving advantages and disadvantages of each Become more confident in applying their knowledge to a range of contexts (eg in programming, and harder exam questions)	Some of this builds directly on work covered in GCSE Computing, however note that not all students will have done GCSE Computing. Students tend to know the gist of topics from GCSE (eg bus topology) but usually struggle to answer questions in the required depth without clear modelling. Using the practice exam questions throughout helps students to see the standard required.	Textbook p310-325 Resources – outline PowerPoints with suggested examples and scaffolding activities Practice questions (from past exams) and section assessments

Computer Science - Year 12 Summer 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?**
Teacher 1 4.10 Databases and SQL	Knowledge: Relational databases and entity-relationship (ER) diagrams. Definitions of: entity, attribute, primary key, composite primary key, foreign key, normalisation. Use of SQL to retrieve, update, insert and delete data. Use of SQL to define a database table. Know how concurrent access to a database can be controlled to preserve the integrity of the database. Understanding: Why databases are normalised. Why concurrent access can cause issues and how to they can be prevented/resolved. Skills: Produce a data model from given data requirements for a simple scenario involving multiple entities. Be able to identify why given relations are not normalised, and be able to normalise given relations to third normal form	In particular students should: Through regular revisiting, be able to quickly recall all key terms used in this section and give examples of each where appropriate. 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 Become more confident in applying their knowledge to a range of contexts (eg in programming, and harder exam questions)	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.	Textbook p364-393 Resources – outline PowerPoints with suggested examples and scaffolding activities Practice questions (from past exams) and section assessments For practising SQL: https://www.w3schools.com/ Python files: - Using sqlite - Using sqlite advanced

Teacher 2 4.1 Fundementals of Programming NEA - Introduction	Knowledge: Be familiar with the key concepts: class, object, instantiation, encapsulation, inheritance, aggregation, composition, polymorphism, overriding. Use of 'ADITE' for a project. Understanding: The object-oriented paradigm and why it is used. Be aware of the principles: encapsulate what varies, favour composition over inheritance, program to interfaces, not implementation. Skills: Experience of programming in object-oriented programming paradigm involving: abstract, virtual and static methods, inheritance, aggregation, polymorphism, public/private/protected specifiers. Be able to draw and interpret class diagrams.	In particular students should: Through regular revisiting, be able to quickly recall all key terms used in this section and give examples of each where appropriate. Confidently program using classes Confidently draw class diagrams showing the relationships between classes Understand private/public/protected despite Python not explicitly using them Become more confident in applying their knowledge to a range of contexts (eg in programming, and harder exam questions)	Most students will not be familiar with classes (unless they have experience programming in Java or similar). However this builds on the concepts of a structured programming as covered in Y12 Autumn 1 (Teacher 1).	Textbook p25-49 Resources – outline PowerPoints with suggested examples and scaffolding activities Practice questions (from past exams) and section assessments Many Python files available for introducing the various concepts and giving examples (eg of composition vs inheritance)