AS and A LEVEL
# COMPUTER SCIENCE

OCR
Oxford Cambridge and RSA

| Component 1 | | | Year 12 | Year 13 |
|---|---|---|---|---|
| **Topic** | **Sub Topic** | Teacher | Term | Term |
| 1.1.1 Structure and Function of Processor | 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: | DFE | Autumn 1 | |
| | The Fetch-Decode-Execute Cycle; including its effects on registers. | | | |
| | The factors affecting the performance of the CPU: clock speed, number of cores, cache. | | | |
| | The use of pipelining in a processor to improve efficiency | | | |
| 1.1.2 Types of Processor | The differences between and uses of CISC and RISC processors. | | | |
| | GPUs and their uses (including those not related to graphics). | | | |
| | Multicore and Parallel systems. | | | |
| 1.1.3 Input, Output and storage | How different input, output and storage devices can be applied to the solution of different problems. | | | |
| | The uses of magnetic, flash and optical storage devices. | | | |
| | RAM and ROM. | | | |
| | Virtual storage. | | | |
| 1.2.1 Systems Software | The need for, function and purpose of operating systems. | DFE | Autumn 2 | |
| | Memory Management (paging, segmentation and virtual memory). | | | |
| | Interrupts, the role of interrupts and Interrupt Service Routines (ISR), role within the Fetch-Decode-Execute Cycle. | | | |
| | Scheduling: round robin, first come first served, multi-level feedback queues, shortest job first and shortest remaining time. | | | |
| | Distributed, embedded, multi-tasking, multi-user and Real Time operating systems. | | | |
| | BIOS. | | | |
| | Device drivers. | | | |
| | Virtual machines, any instance where software is used to take on the function of a machine, including executing intermediate code or running an operating system within another. | | | |
| 1.2.2 Applications Generation | The nature of applications, justifying suitable applications for a specific purpose. | | | |
| | Utilities. | | | |
| | Open source vs. closed source. | | | |
| | Translators: Interpreters, compilers and assemblers. | | | |
| | Stages of compilation (lexical analysis, syntax analysis, code generation and optimisation). | | | |
| | Linkers and loaders and use of libraries. | | | |
| 1.2.3 Software Development | Understand the waterfall lifecycle, agile methodologies, extreme programming, the spiral model and rapid application development. | CHE | Spring 2 | Autumn 1 |
| | The relative merits and drawbacks of different methodologies and when they might be used. | CHE | | |
| | Writing and following algorithms. | DFE | | |
| 1.2.4 Types of Programming Language | Need for and characteristics of a variety of programming paradigms. | DFE | | Autumn 1 |
| | Procedural languages:<br>• program flow | CHE | Autumn 1 | |
| | Assembly language (including following and writing simple programs with the Little Man Computer instruction set). | DFE | | Autumn 1 |
| | Modes of addressing memory (immediate, direct, indirect and indexed). | DFE | | |
| | Object-oriented languages with an understanding of classes, objects, methods, attributes, inheritance, encapsulation and polymorphism. | CHE | Autumn 2 | |
| 1.3.1 Compression, Encryption and Hashing | Lossy vs. Lossless compression. | DFE | | Autumn 2 |
| | Run length encoding and dictionary coding for lossless compression. | | | |
| | Symmetric and asymmetric encryption. | | | |
| | Different uses of hashing. | | | |
| 1.3.2 Databases | Relational database, flat file, primary key, foreign key, secondary key, entity relationship modelling, normalisation and indexing. | CHE | Spring 1 | |
| | Methods of capturing, selecting, managing and exchanging data. | | | |
| | Normalisation to 3NF. | | | |
| | SQL – Interpret and modify. | | | |
| | Referential integrity. | | | |
| | Transaction processing, ACID (Atomicity, Consistency, Isolation, Durability), record locking and redundancy. | | | |
| 1.3.3 Networks | Characteristics of networks and the importance of protocols and standards. | DFE | | Spring 1 |
| | The internet structure:<br>• The TCP/IP Stack. | | | |
| | Network security and threats, use of firewalls, proxies and encryption. | | | |
| | Network hardware. | | | |

| | | | | |
|---|---|---|---|---|
| | Client-server and peer to peer. | | | |
| 1.3.4 Web Technologies | HTML, CSS and JavaScript.<br><br>Search engine indexing.<br><br>PageRank algorithm.<br><br>Server and client side processing. | DFE | | Spring 2 |
| 1.4.1 Data Types | Primitive data types, integer, real/floating point, character, string and Boolean.<br><br>Represent positive integers in binary.<br><br>Use of sign and magnitude and two's complement to represent negative numbers in binary.<br><br>Addition and subtraction of binary integers.<br><br>Represent positive integers in hexadecimal.<br><br>Convert positive integers between binary hexadecimal and denary.<br><br>Representation and normalisation of floating point numbers in binary.<br><br>Floating point arithmetic, positive and negative numbers, addition and subtraction.<br><br>Bitwise manipulation and masks: shifts, combining with AND, OR, and XOR.<br><br>How character sets (ASCII and UNICODE) are used to represent text. | DFE | Autumn 1/ Autumn | |
| 1.4.2 Data Structures | Arrays (up to 3 dimensions), records, lists, tuples<br><br>Following structures to store data: linked-list, graph (directed and undirected), stack, queue, tree, binary search tree, hash table.<br>How to create, traverse, add data to and remove data from the above data structures | DFE | Spring 2/ Summer 1 | |
| 1.4.3 Boolean Algebra | Define problems using boolean logic.<br><br>Manipulate Boolean expressions, including the use of Karnaugh maps to simplify Boolean expressions<br><br>Use the following rules to derive or simplify statements in Boolean algebra: De Morgan's Laws, distribution, association, commutation, double negation.<br>Using logic gate diagrams and truth tables.<br><br>The logic associated with D type flip flops, half and full adders. | DFE | Spring 1 | |
| 1.5.1 Computing related legislation | The Data Protection Act 1998.<br><br>The Computer Misuse Act 1990.<br><br>The Copyright Design and Patents Act 1988.<br><br>The Regulation of Investigatory Powers Act 2000. | DFE | Summer 2 | Autumn 1 |
| 1.5.2 Moral and Ethical Issues | The individual moral, social, ethical and cultural opportunities and risks of digital technology: | | | |
| Component 2 | | | | |
| 2.1.1 Thinking Abstractly | The nature of abstraction.<br><br>The need for abstraction.<br><br>The differences between an abstraction and reality.<br><br>Devise an abstract model for a variety of situations. | | Spring 2 | |
| 2.1.2 Thinking Ahead | Identify the inputs and outputs for a given situation.<br><br>Determine the preconditions for devising a solution to a problem.<br><br>The nature, benefits and drawbacks of caching.<br><br>The need for reusable program components. | | | |
| 2.1.3 Thinking Procedurally | Identify the components of a problem.<br><br>Identify the components of a solution to a problem.<br><br>Determine the order of the steps needed to solve a problem.<br><br>Identify sub-procedures necessary to solve a problem. | | | |
| 2.1.4 Thinking Logically | Identify the points in a solution where a decision has to be taken.<br><br>Determine the logical conditions that affect the outcome of a decision.<br><br>Determine how decisions affect flow through a program. | | | |
| 2.1.5 Thinking Concurrently | Determine the parts of a problem that can be tackled at the same time.<br><br>Outline the benefits and trade offs that might result from concurrent processing in a particular situation. | | | |
| 2.2.1 Programming Techniques | Programming constructs: sequence, iteration, branching.<br><br>Recursion, how it can be used and compares to an iterative approach.<br>Global and local variables.<br><br>Modularity, functions and procedures, parameter passing by value and by reference.<br>Use of an IDE to develop/debug a program.<br><br>Use of object oriented techniques. | DFE | Summer 1 | |
| 2.2.2 Computational Methods | Features that make a problem solvable by computational methods.<br><br>Problem recognition.<br><br>Problem decomposition. | | | |

| | | | | |
|---|---|---|---|---|
| | Use of divide and conquer.<br><br>Use of abstraction.<br><br>Learners should apply their knowledge of:<br>• backtracking | | | |
| 2.2.1 Algorithms | Analysis and design of algorithms for a given situation.<br><br>The suitability of different algorithms for a given task and data set, in terms of execution time and space.<br><br>Standard algorithms (bubble sort, insertion sort, binary search and linear search).<br><br>Standard algorithms (quick sort, Dijkstra's shortest path algorithm, A* algorithm, binary search).<br><br>Implement bubble sort, insertion sort.<br><br>Implement binary and linear search.<br><br>Measures and methods to determine the efficiency of different algorithms, Big O notation (constant, linear, polynomial, exponential and logarithmic complexity).<br>Algorithms for the main data structures, (stacks, queues, trees, linked lists, depth-first (post-order) and breadth-first traversal of trees).<br>Comparison of the complexity of algorithms.<br><br>Compare the suitability of different algorithms for a given task and data set. | DFE | | Autumn 2/ Spring 1 |
| Component 3 | | | | |
| **Analysis of the problem** | | | | |
| Problem Identification | Describe and justify the features that make the problem solvable by computational methods. | CHE | Spring 2 | |
| | Explain why the problem is amenable to a computational approach. | | | |
| Stakeholders | Identify and describe those who will have an interest in the solution explaining how the solution is appropriate to their needs (this may be named individuals, groups or persona that describes the target end user). | | | |
| Research the Problem | Research the problem and solutions to similar problems to identify and justify suitable approaches to a solution. | | | |
| | Describe the essential features of a computational solution explaining these choices. | | | |
| | Explain the limitations of the proposed solution. | | | |
| Specify the Proposed Solution | Identify the points in a solution where a decision has to be taken. | | | |
| | Determine the logical conditions that affect the outcome of a decision | | | |
| | Determine how decisions affect flow through a program. | | | |
| **Design of the solution** | | | | |
| Decompose the Problem | Break down the problem into smaller parts suitable for computational solutions justifying any decisions made. | CHE | Summer 2 | |
| Describe the solution | Explain and justify the structure of the solution | | | |
| | Describe the parts of the solution using algorithms justifying how these algorithms form a complete solution to the problem. | | | |
| | Describe usability features to be included in the solution. | | | |
| | Identify key variables / data structures / classes justifying choices and any necessary validation. | | | |
| Describe the approach to testing | Understand the waterfall lifecycle, agile methodologies, extreme programming, the spiral model and rapid application development. | | | |
| | The relative merits and drawbacks of different methodologies and when they might be used. | | | |
| | Writing and following algorithms. | | | |
| | Different test strategies, including black and white box testing and alpha and beta testing. | | | |
| | Test programs that solve problems using suitable test data and end user feedback, justify a test strategy for a given situation. | | | |
| **Developing the solution** | | | | Autumn 1&2 |
| Iterative Development Process | Provide annotated evidence of each stage of the iterative development process justifying any decision made. | CHE | | |
| | Provide annotated evidence of prototype solutions justifying any decision made. | | | |
| Testing to inform development | Provide annotated evidence for testing at each stage justifying the reason for the test. | | | |
| | Provide annotated evidence of any remedial actions taken justifying the decision made. | | | |
| **Evaluation** | | | | |
| Testing to inform evaluation | Provide annotated evidence of testing the solution of robustness at the end of the development process. | CHE | | Spring 1&2 |
| | Provide annotated evidence of usability testing (user feedback). | | | |
| Success of the solution | 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 | Provide annotated evidence of the usability features from the design, commenting on their effectiveness. | | | |
| Maintenance and development | Discuss the maintainability of the solution.<br><br>Discuss potential further development of the solution. | | | |