Cardinal Newman Catholic School
Department of Computer Science and Creative iMedia

INTENT: Curriculum Overview Year 12 OCR A-Level Computer Science

A learner in Year 12 will know: The knowledge and understanding to apply the fundamental principles and concepts of computer science, including: abstraction, decomposition, logic, algorithms and data representation; The ability to analyse problems in computational terms through practical experience of solving such problems, including writing programs to do so; and the capacity to think creatively, innovatively, analytically, logically and critically	A learner in Year 12 will be able to: Apply the academic principles learned in the classroom to real-world systems, preparing them for the end of year 12 as they begin their 20% NEA, the practical programming project.

Term 1	**12.1 Topics** 1.1 The characteristics of contemporary processors, input, output and storage devices 1.2 Software and software development 2.1 Elements of computational thinking 2.2 Problem solving and programming	**12.2 Topics** 1.4 Data types, data structures and algorithms 2.1 Elements of computational thinking 2.2 Problem solving and programming	Autumn Summative Assessment
	Knowledge: **1.1.1 Structure and function of the processor** a) Control unit and registers 1.1.1 b) The FDE cycle including its effects on registers. 1.1.1 c) Factors affecting the performance of a CPU: clock speed, number of cores, cache, d) The use of pipelining in a processor to improve efficiency, e) CPU architectures **1.1.2 Types of processor** a) Differences between and uses of CISC and RISC processors 1.1.2 b) GPUs and their uses (including those not related to graphics), c) Multicore and Parallel systems **1.1.3 Input, output and storage** a) Application for different problems 1.1.3 b) Uses of magnetic, flash and optical storage 1.1.3 c) RAM and ROM, d) Virtual storage **1.2.1 Systems software** a) The need for, function and purpose of operating systems 1.2.1 e) Distributed, embedded, multi-tasking, multi-user and Real Time operating systems 1.2.1 f) BIOS, g) Device drivers **1.2.2 Applications Generation** a) The nature of applications, justifying suitable applications for a specific purpose 1.2.2 b) Utilities 1.2.2 c) Open vs closed source 1.2.2 d) Translators: Interpreters, compilers and assemblers, e) Stages of compilation (lexical analysis, syntax, analysis, code generation and optimisation), f) Linkers and loaders and use of libraries **2.1.2 Thinking ahead** a) Identifying inputs and outputs	Knowledge: **1.4.1 Data Types** a) Primitive data types, integer, real/floating point, character, string and Boolean 1.4.1 b) Represent positive integers in binary 1.4.1 c) Use of sign and magnitude and two's complement to represent negative numbers in binary 1.4.1 d) Addition and subtraction of binary integers 1.4.1 e) Represent positive integers in hexadecimal 1.4.1 f) Convert positive integers between binary, hexadecimal and denary 1.4.1 g) Representation and normalisation of floating-point numbers in binary 1.4.1 h) Floating point arithmetic, positive and negative numbers, addition and subtraction **2.1.1 Thinking abstractly** a) The nature of abstraction, b) The need for abstraction, c) The differences between an abstraction and reality, d) Devise an abstract model for a variety of situations **2.1.3 Thinking Procedurally** a) Identify the components of a problem, b) Identify the components of a solution to a problem, c) Determine the order of steps needed to solve a problem, d) Identify sub-procedures necessary to solve a problem **2.2.1** c) Global and local variables. Programming tasks 2.2.1 d) Modularity, functions and procedures, parameter passing by value and by reference 2.2.1 e) Use of an IDE to develop/debug a program **2.2.2 Computational Methods** a) Features that make a problem solvable by computational methods, b) Problem recognition	Assessment follows the policy of the school, students will have a piece of work marked every 4 weeks. This will be based on practise questions around the topics covered during that period. All questions will follow the structure of those they will answer in the real exams. **Summative assessment** During the assessment period a mock exam paper will be used to cover real exam past-paper questions that cover the content students have covered.

	2.1.2 b) Determining pre-conditions for devising a solution to a problem 2.1.2 c) The nature, benefits and drawbacks of caching 2.1.2 d) The need for reusable program components **2.2.1 Programming techniques** a) Sequence, iteration and branching 2.2.1 b) Recursion, how it can be used and compares to an iterative approach **2.1.4 Thinking logically** a) Identify the points in a solution where a decision has to be taken, b) Determine the logical conditions that affect the outcome of a decision, c) Determine how decisions affect flow through a program	2.2.2 c) Problem decomposition, d) Use of divide and conquer, e) Use of abstraction 2.2.2 f) Learners should apply their knowledge of; backtracking, data mining, heuristics, performance modelling, pipelining, visualisation to solve problems.	
Term 2	**12.3 Topics** 1.4 Data types, data structures and algorithms 2.1 Elements of computational thinking 2.3 Algorithms	**12.4 Topics** 1.2 Software and software development 1.3 Exchanging data 3.1 Analysis of the problem	Spring Summative Assessment
	Knowledge: **1.4.2 Data Structures** a) Arrays (of up to 3 dimensions), records, lists, tuples. 1.4.2 b) The following structures to store data: linked-list, graph (directed and un-directed), stack, queue, tree, binary search tree, hash table. 1.4.2 c) How to create, traverse, add data to and remove data from the data structures mentioned above **2.1.5 Thinking concurrently** a) Determine the parts of a problem that can be tackled at the same time, b) Outline the benefits and trade-offs that might result from concurrent processing in a particular situation. **2.3.1 Algorithms** a) Analysis and design of algorithms for a given situation, b) The suitability of different algorithms for a given task and data set, in terms of execution time and space 2.3.1 c) Measures and methods to determine the efficiency of different algorithms, Big O notation (constant, linear, polynomial, exponential and logarithmic complexity), d) Comparison of the complexity of algorithms 2.3.1 e) Algorithms for the main data structures, (stacks, queues, trees, linked lists, depth-first (post-order) and breadth-first traversal of trees) 2.3.1 f) Standard algorithms (bubble sort, insertion sort, merge sort, quick sort, Dijkstra's shortest path algorithm, A* algorithm, binary search and linear search)	Knowledge: **1.2.3 Software Development** a) Understand the waterfall cycle, agile methodologies, extreme programming, the spiral model and rapid application development 1.2.3 b) The relative merits and drawbacks of different methodologies and when they might be used 1.2.3 c) Writing and following algorithms **1.3.2 Databases** a) Relational database, flat file, primary key, foreign key, secondary key, entity relationship modelling, normalisation and indexing 1.3.2 b) Methods of capturing, selecting, managing and exchanging data 1.3.2 c) Normalisation to 3NF 1.3.2 d) SQL - Interpret and modify 1.3.2 e) Referential integrity 1.3.2 f) Transactional processing, ACID (Atomicity, Consistency, Isolation, Durability, record locking and redundancy **3.1 Analysis of the problem, 3.1.1 Problem identification** a) Describe and justify the features that make the problem solvable by computational methods 3.1.1 b) Explain why the problem is amenable to a computational approach. **3.1.2 Stakeholders** a) 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)	Assessment follows the policy of the school, students will have a piece of work marked every 4 weeks. This will be based on practise questions around the topics covered during that period. All questions will follow the structure of those they will answer in the real exams. **Summative assessment** During the assessment period a mock exam paper will be used to cover real exam past-paper questions that cover the content students have covered.

		3.1.3 Research the problem a) Research the problem and solutions to similar problems to identify and justify suitable approaches to a solution. 3.1.3 b) Describe the essential features of a computational solution explaining these choices. 3.1.3 c) Explain the limitations of the proposed solution **3.1.4 Specify the proposed solution** a) Specify and justify the solution requirements including hardware and software configuration (if appropriate) 3.1.4 b) Identify and justify measurable success criteria for the proposed solution	
Term 3	**12.5 Topics** 1.3 Exchanging data 1.5 Legal, moral, cultural and ethical issues 3.2 Design of the solution	**12.6 Topics/Themes** 1.2 Software and software development 1.5 Legal, moral, cultural and ethical issues 3.2 Design of the solution 3.3 Developing the solution	Summer Summative Assessment
	Knowledge: **1.3.1 Compression, Encryption and Hashing** a) Lossy vs Lossless compression 1.3.1 b) Run length encoding and dictionary coding for lossless compression 1.3.1 c) Symmetric and asymmetric encryption, d) Different uses of hashing **1.3.3 Networks** a) Characteristics of networks and the importance of protocols and standards 1.3.3 b) The Internet structure: The TCP/IP Stack, DNS, Protocol layering, LANs and WANs, Packet and circuit switching 1.3.3 c) Network security and threats, use of firewalls, proxies and encryption 1.3.3.d) Network hardware 1.3.3 e) Server and client-side processing **1.3.4 Web Technologies** a) HTML, CSS and JavaScript 1.3.4 b) Search engine indexing 1.3.4 c) PageRank algorithm, d) Server and client side processing **1.5.1 Computing related legislation** a) The Data Protection Act 1998, b) The Computer Misuse Act 1990 1.5.1 c) The Copyright Designs and Patents Act 1988, d) The Regulatory of Investigatory Powers Act 2000 **3.2 Design of the solution, 3.2.1 Decompose the problem** a) Break down the problem into smaller parts suitable for computational solutions justifying any decisions made	Knowledge: **1.5.2 Moral and ethical Issues** The individual moral, social, ethical and cultural opportunities and risks of digital technology: Computers in the workforce **(**1.2.1 h) Virtual machines**)** 1.5.2 Automated decision making 1.5.2 Artificial Intelligence 1.5.2 Environmental effects 1.5.2 Censorship and the Internet 1.5.1 Monitor behaviour 1.5.1 Analyse personal information 1.5.1 Piracy and offensive communications 1.5.1 Layout, colour paradigms and character sets **1.2.1** b) Memory management (paging, segmentation and virtual memory) 1.2.1 c) Interrupts, the role of interrupts and Interrupt Service Routines (ISR), role within the FDE cycle 1.2.1 d) Scheduling: round robin, first come first served, multi-level feedback queues, shortest job first and shortest time remaining 1.4.3 e) The logic associated with D type flip flops, half and full adders **3.2.2 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 data **3.3 Developing the solution, 3.3.1 Iterative development process** a) Provide annotated evidence of each stage of the iterative	Assessment follows the policy of the school, students will have a piece of work marked every 4 weeks. This will be based on practise questions around the topics covered during that period. All questions will follow the structure of those they will answer in the real exams. **Summative assessment** During the assessment period a mock exam paper will be used to cover real exam past-paper questions that cover the content students have covered.

	3.2.2 Describe the solution a) Explain and justify the structure of the solution 3.2.2 b) Describe the parts of the solution using algorithms justifying how these algorithms form a complete solution to the problem 3.2.2 c) Describe usability features to be included in the solution 3.2.2 d) Identify key variables / data structures / classes justifying choices and any necessary validation	development process justifying any decision made, b) Provide annotated evidence of prototype solutions justifying and decision made. **3.3.2 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.	