Croxley Danes School : Key Stage _5_ Curriculum Map

Subject: Computer Science **Exam Board OCR**

Key Concepts

Components of a Computer	Systems Software & Software Development	Exchanging data & Data structures	Networks and web Technologies	Boolean Algebra & legal, moral ethical and cultural issues	Data types , Programming techniques, Computational Thinking ,Algorithms & Programming project
• Processor components • Processor performance • Types of processor • Input devices/Output devices • Storage devices	• Functions of an operating system • Types of an operating system • The nature of applications • Programming language translators • System analysis methods • Writing and following algorithms • Programming paradigms • Assembly language	• Compression,encryption and hashing • Database concepts • Relational databases and normalisation • Introduction to SQL • Defining and updating tables using SQL • Transaction processing • Arrays,tuples records • Ques, lists, linked lists • Stacks, hashtables graphs,trees	• Structure of the internet • Internet communication • Network security and threats • HTML and CSS • Web forms and Javascript • Search engine indexing • Client -server and peer to peer	• Logic gates and truth tables • Simplifying Boolean expressions • Karnaugh maps • Adders and D -type flip flops • Computing related legislation • Ethical , moral and cultural issues • Privacy and censorship	• Primitive data types, binary and hexadecimal • ASCII and UNICODE • Binary arithmetic • Floating point arithmetic • Bitwise manipulation and masks • Programming basics • Selection,iteration,subroutines and recursion • Use of IDE • Use of OOP Techniques • Analysis and design of algorithms • Bubble sort , insertion sort, merge sort and quick sort • Graph traversal algorithms • Optimisation algorithms

What is the Croxley vision for this subject at Key Stage 5 ?

At A Level, we follow the OCR H446 Computer Science specification, which builds on the foundations developed at Key Stage 3 and GCSE. This qualification helps students understand the core academic principles of Computer Science and apply them to practical, real-world contexts. Students develop a deeper understanding of key topics such as programming constructs, computational thinking, hardware and software, networks, system development life cycles, and the wider implications of computing.Classroom learning is reinforced through an independent programming project, which enables students to create a substantial piece of software based on an area of personal interest. The open-ended nature of the project encourages ambition and creativity, with students able to use familiar languages like Python or challenge themselves with alternatives such as Java or a language from the C family.

Problem-solving is at the heart of the course, fostering technical understanding and analytical thinking that benefit both Computer Science and wider academic pursuits. While those who have studied GCSE Computer Science will encounter familiar topics, they will explore them at a significantly more advanced level, gaining both depth and confidence in their application.

Key Stage 5 / Year Group: 12 - topics are taught concurrently across 2 teachers			
	Autumn Term 1	**Autumn Term 2**	**Spring Term 1**
key concept	Components of a Computer	Systems software	Exchanging data
Content: **(Know what…)**	Processor components Processor performance Types of processor Input devices/Output devices Storage devices	Functions of an operating system Types of an operating system The nature of applications Programming language translators System analysis methods Writing and following algorithms Programming paradigms Assembly language	Compression,encryption and hashing Database concepts Relational databases and normalisation Introduction to SQL Defining and updating tables using SQL Transaction processing
Skills: **(know how…)**	• List the basic internal components of the processor: ALU, Control Unit, registers and buses • Name the different registers used in the Fetch-Decode-Execute cycle • List factors which affect the performance of the CPU: clock speed, number of cores, cache • Explain what is meant by a multicore system and parallel processing • Describe the uses of RAM, ROM and virtual storage • Describe typical uses of magnetic, flash and optical storage devices • Describe the Fetch-Decode-Execute cycle and its effect on registers • Describe the role of the data, address and control buses • Describe the use of pipelining in a processor to improve efficiency	• State the function and purpose of an operating system • Describe scheduling algorithms: round robin, first come first served, multi-level feedback queues, shortest job first and shortest remaining time • Describe distributed, embedded, multi-tasking, multi-user and real-time operating systems • Describe the function of BIOS and device drivers • Distinguish between systems software and applications software • Describe what is meant by a utility program and give examples • Be able to justify a suitable application for a specific purpose • Distinguish between open source and closed source software • State the roles of an assembler, compiler and interpreter • Describe the use of libraries	• explain the difference between lossy and lossless compression and list advantages and disadvantages of each • use basic encryption to create ciphertext • encrypt and decrypt a message using the Caesar cipher • explain the weaknesses of the Caesar cipher • define the terms flat file, primary key, indexing • define the terms relational database, foreign key, secondary key, entity • draw a simple entity relationship diagram involving three or four entities • state the properties of a database in Third Normal Form • interpret a simple SQL statement • list methods of capturing data for input to a database • list problems that can arise with a multi-user database • explain run length encoding and dictionary based compression • use lossy compression methods to reduce file size

	• Describe von Neumann and Harvard architectures and the advantages and use of each • Describe the characteristics and use of a Graphics Processing Unit (GPU) • Describe the benefits and potential problems of virtual storage • Describe the uses of magnetic, flash and optical storage devices • Describe how different I/O and storage devices can be applied to the solution of different problems	• Describe memory management (paging, segmentation and virtual memory) • Describe the role of interrupts • Describe the need for processor scheduling algorithms • Explain the difference between compilation and interpretation, and describe situations when both would be appropriate • Describe what is meant by a virtual machine • Describe the stages of compilation: lexical analysis, syntax analysis, code generation and optimisation • Describe the function of linkers and loaders	• explain the differences between asymmetric and symmetric encryption • explain the use of hashing to encrypt data • draw a complex entity relationship diagram involving several entities • normalise a database to third normal form • list the advantages of a normalised database • use SQL to select data from related tables in a database • describe methods of capturing, selecting, managing and exchanging data • describe what is meant by record locking and why it is necessary in a multi-user database • Describe what is meant by redundancy
Key vocabulary (5- 10 words)	Arithmetic and Logic Unit (ALU), Control Unit, bus, registers, program counter (PC), accumulator (ACC), memory address register (MAR), memory data register (MDR), current instruction register (CIR), Fetch-Decode-Execute cycle, clock speed, core, cache, pipelining, von Neumann architecture, Harvard architecture, CISC, RISC, multicore, parallel systems, graphics processing unit (GPU) magnetic, flash and optical storage devices, RAM, ROM, virtual storage	Operating System (OS), Read Only Memory (ROM), paging, segmentation, Random Access Memory (RAM), contiguous, virtual memory, disk thrashing, interrupt, scheduling algorithm, stack, time slice, distributed, embedded, failsafe, redundancy, BIOS, driver, virtual machine, utility software, disk defragmenter, heuristics, library, translator, bespoke, proprietary assembler, compiler, interpreter, bytecode, lexical analysis, syntax analysis, parsing, symbol table, semantic analysis, linker, loader.	lossy, lossless, compression, run length encoding, dictionary coding, hashing, relational database, flat file, primary key, foreign key, secondary key, entity relationship, modelling, normalisation, indexing 3NF, SQL, referential integrity transaction processing, ACID (Atomicity, Consistency, Isolation, Durability), record locking, deadlock, redundancy
End of Half term assessment	Baseline assessment at start of year 12 End of topic tests	UCAS Assessment	End of topic tests
Planned trips / Clubs / links			
Key Stage 5/ Year Group: 12-topics are taught concurrently across 2 teachers			

	Spring 2	Summer 1	Summer 2
Key Concept	Data structures	Networks and Web technologies	Boolean Algebra
Content: **(Know what…)**	Arrays,tuples records Ques, lists, linked lists Stacks, hashtables graphs,trees	Structure of the internet Internet communication Network security and threats HTML and CSS Web forms and Javascript Search engine indexing Client -server and peer to peer	Logic gates and truth tables Simplifying Boolean expressions Karnaugh maps Adders and D -type flip flops
Skills: **(Know how…)**	• describe the concept and uses of a queue, stack, graph, tree, binary search tree and hash table • list typical uses of each of these data structures • know how an adjacency matrix and an adjacency list may be used to represent a graph • traverse a binary tree in pre-order, in-order and post-order • create a binary search tree • be able to apply a simple hashing algorithm • state what is meant by a collision and describe how collisions may be handled • distinguish between an array, list and tuple • describe the addition, deletion and maintenance of data within queues, stacks, hash tables and trees • describe the characteristics of an array-based queue, circular queue and priority queue • write an algorithm for traversing a linked list • be able to compare the use of adjacency matrices and adjacency lists for representing graphs	• State the importance of protocols and standards • Describe the structure of the Internet • Explain the protocols used within the TCP/IP stack • Demonstrate DNS in action using an IP address within a web browser • Describe and identify examples of LANs and WANs • Explain packet switching • Provide examples of network threats and state methods to overcome these • Explain the function of a firewall • State the functions of a proxy server • Create a basic webpage using HTML and some CSS • Use JavaScript to make web form elements interactive and add validation • Describe the characteristics of the PageRank algorithm and state the factors that influence page ranking • Describe the processes at each layer of the TCP/IP stack • Explain the DNS resolution process • Explain packet switching in contrast to circuit switching	• construct truth tables for a variety of logic gates • draw and interpret logic gate circuit diagrams involving multiple gates • write a Boolean expression for a given logic gate circuit • draw an equivalent logic gate circuit for a given Boolean expression • Draw a Karnaugh map corresponding to a given expression or truth table • Draw a truth table for a half adder • State what a D type flip flop is and what it is used for • Complete a truth table for a given logic gate circuit • Represent and solve a problem using Boolean logic • Use de Morgan's laws to manipulate and simplify Boolean expressions • Simplify an expression using a Karnaugh map • Draw the logic circuit for a half adder • Give the output from a series of connected D type flip flops

	• be able to apply a number of different hashing algorithms • describe and apply the following operations to a linear, circular and priority queue: add an item, remove an item, test for empty queue, test for full queue • write algorithms for adding and deleting elements to/from a linked list • write algorithms for pre-order, in-order and post-order tree traversals	• State the advantages of layering protocols in the TCP/IP stack • Explain, by use of example, the difference between client and server side processing • Use sequence and selection statements in JavaScript with a range of data types including arrays	
Key vocabulary (5- 10 words)	array, record, tuple, list, linked list, queue, circular queue, priority queue, First In, First Out (FIFO), enqueue, dequeue Append, push, pop, stack, Last In, First Out (LIFO), call stack, stack frame, parameter, return address, Hashing, hash table, collision, mid-square method, folding method, dictionary Graph, edge, arc, vertex, node, directed graph, digraph, undirected graph, weighted edge, adjacency matrix, adjacency list, Page Rank algorithm Tree, root, child, parent, subtree, leaf node, binary search tree, pre-order, in-order and post-order traversal	URL, Internet registry, registrar, DNS, IP address, WAN, LAN, topology, bus, star, Wi-Fi, WAP, packet switching, router, gateway, MAC address, TCP/IP stack, protocol, layer, FTP, POP, IMAP, SMTP, packet, filtering, firewall, proxy, worm, Trojan, virus, malware, social engineering, phishing, HTML, CSS, JavaScript, tag, PageRank, index, meta tag, client-server, API, client side processing, server side processing	logic gate, truth table, Boolean algebra, distribution, association, commutation, double negation, Karnaugh map, D type flip flop, half adder, full adder
End of Half term assessment	End of topic tests Year 12 Mock Exams	End of topic tests	End of topic test
Planned trips / Clubs / links		Trip to the National museum of Computing in Bletchley	

Key Stage 5 / Year Group: 13 -topics are taught concurrently across 2 teachers			
	Autumn Term 1	**Autumn Term 2**	**Spring Term 1**
key concept	Legal, moral,ethical and cultural issues	Data types	Software development
Content: **(Know what…)**	Computing related legislation Ethical , moral and cultural issues Privacy and censorship	Primitive data types, binary and hexadecimal ASCII and UNICODE Binary arithmetic Floating point arithmetic Bitwise manipulation and masks	System analysis methods Writing and following algorithms Programming paradigms Assembly language
Skills: **(know how…)**	• Give examples of organisations that amass and analyse personal information • Give examples of some of the moral and ethical choices which arise when digital technology is used. • Explain, with examples, how some software applications have resulted in great benefits but also caused great harm • Give examples from the range of laws which relate to the use, and misuse, of digital technology to gather, store, process and distribute digital data. • Explain that there are cultural influences on the way information is understood. • Explain the difference between data processing and artificial intelligence • Comment on the current capacity to distribute, publish, communicate and disseminate personal information and the benefits and drawbacks of this capability • Discuss some of the issues involved in regard to the collection and analysing of personal information by security agencies and other organisations, and relate them to relevant laws.	• list primitive data types • represent positive integers in binary • use sign and magnitude and two's complement to represent negative numbers in binary • add two unsigned binary numbers • represent positive numbers in hexadecimal • convert between denary, binary and hexadecimal number systems • define bits and bytes, and use names, symbols and prefixes appropriately • know how to use the ASCII table to represent text as binary • explain why Unicode was introduced, and its advantages • use arithmetic operations and Boolean operations AND, OR and XOR • show the effect of a logical shift left and shift right of a number of bits • use fixed point binary to represent numbers with a fractional part • convert a positive floating point number to denary and vice versa • normalise a positive floating point number	• list the stages in the waterfall lifecycle model • name two other systems development models • name and describe different types of testing • write a pseudocode algorithm to solve a simple problem • use a trace table to trace through an algorithm • interpret simple algorithms to describe their purpose • list two features of a good algorithm • Define the term "programming paradigm" and give an example of two paradigms • define the terms object, class, method, attribute, inheritance • draw a simple inheritance diagram for a set of classes in an object-oriented approach • follow through a simple program using the LMC instruction set • briefly describe agile methodologies, extreme programming, the spiral model and rapid application development • write pseudocode algorithms to solve problems • describe different programming paradigms, including procedural, and object oriented paradigms • explain the terms encapsulation and polymorphism

	• Identify some of the ethical issues arising from the use of digital technology • Describe some environmental effects of digital technology • Describe both positive and negative impacts of the use of computers on the workforce • Describe some of the opportunities and risks of artificial intelligence and automated decision making	• use arithmetic, logical and circular shifts • differentiate between the character code for a digit and its pure binary representation	• write simple assembly code programs using the LMC instruction set • distinguish between immediate, direct and indirect addressing modes in assembly language
Key vocabulary (5- 10 words)	Legislation, moral, ethical, cultural, computer misuse, copyright, patent, investigatory powers, surveillance, interception, privacy, piracy, environment, recycling, censorship, paradigm	Primitive data types, integer, real/floating point, character, string, Boolean, denary, binary, signed and unsigned, sign and magnitude, fixed point, floating point, two's complement, kibi, mebi, gibi, hexadecimal, ASCII, Unicode, character set, bitwise manipulation, mask, arithmetic shift, logical shift and circular shift	analysis, design, implementations, black box testing, functional testing, white box testing, structural testing, alpha testing, beta testing, waterfall lifecycle model, spiral model, agile modelling, extreme programming, rapid application development algorithm, trace table, programming paradigm, procedural language, functional programming language, declarative language, logic programming, object-oriented paradigm, class, object, method, attribute, inheritance, encapsulation, polymorphism Little Man Computer (LMC), assembly language, machine code instruction, addressing modes (immediate, direct, indirect and indexed), accumulator
End of Half term assessment	Baseline assessment at start of year 13 End of topic test	End of topic tests	End of topic tests
Planned trips / Clubs / links			

Key Stage 5 / Year Group: 13-topics are taught concurrently across 2 teachers			
	Autumn Term 1	**Autumn Term 2**	**Spring Term 1**
Key Concept	Programming techniques	Computational Thinking	Algorithms
Content: **(Know what…)**	Programming basics Selection,iteration,subroutines and recursion	Thinking abstractly Thinking ahead Thinking procedurally Thinking logically, thinking concurrently Problem recognition Problem solving	Analysis and design of algorithms Bubble sort , insertion sort, merge sort and quick sort Graph traversal algorithms Optimisation algorithms
Skills: **(Know how…)**	• use an IDE to develop and debug a program • describe the use of an IDE to check for syntax errors • explain the difference between a variable and a constant • write a pseudocode solution for a simple problem involving iteration and selection (branching) • use nested selection and iteration statements • use arithmetic operations and Boolean operations NOT, AND and OR • use functions and library subroutines including random number generation • know how to define and call a subroutine (procedure or function) with parameters • construct algorithms using one-dimensional arrays • describe what is meant by recursion • define the OOP terms class, object, method, attribute, inheritance, encapsulation and polymorphism • draw an inheritance diagram	• explain the differences between an abstraction and reality • describe the need for reusable program components • identify the inputs and outputs for a given situation • interpret simple algorithms to describe their purpose • give an example of how caching is used in a computer system • Identify the components of a problem • identify the points in a solution where a decision has to be taken • give an example of a Divide and Conquer algorithm • give examples of backtracking, data mining, heuristics, performance modelling, pipelining and visualisation • determine the preconditions for devising a solution to a problem • describe the nature, benefits and drawbacks of caching • identify the components of a problem and its solution • determine the order of steps needed to solve a problem	• state the order in which nodes are visited in pre-order and post-order tree traversals • give examples of linear, polynomial, exponential and logarithmic functions • compare two algorithms in terms of efficiency • explain the principles of a linear and binary search • explain how an insertion sort works • state the two basic steps in a merge sort • show the result of merging two sorted lists • state a possible order in which nodes are visited in depth first and breadth first graph traversals • state applications of each graph traversal • state the purpose and applications of Dijkstra's shortest path algorithm • Explain what is meant by a tractable or intractable problem • state the time complexity of an algorithm • write an algorithm for a linear search • trace through a bubble sort algorithm • trace through an insertion sort algorithm

	• describe features of an IDE which are useful in developing and debugging a program • write a pseudocode solution for a problem involving iteration and selection (branching) • determine the output from a pseudocode program • use structured programming techniques and write their own subroutines with parameters • construct algorithms using two-dimensional arrays • use local and global variables in subroutines • trace through a recursive algorithm • compare iterative and recursive algorithms for solving a problem • complete given pseudocode for an object oriented program	• determine the logical conditions that affect the outcome of a decision • determine how decisions affect flow through a program • identify sub-procedures needed to solve a problem • explain how a Divide and Conquer algorithm works • explain what is meant by backtracking, data mining, heuristics, performance modelling, pipelining and visualisation	• explain how the merge sort works and analyse its time complexity • explain how the quicksort works • describe applications of each graph traversal • be able to trace Dijkstra's shortest path algorithm • Give examples of intractable problems • Describe briefly the A* algorithm and its purpose
Key vocabulary (5- 10 words)	Integrated Development Environment (IDE), syntax errors, logic errors, debug, watch, breakpoint, trace algorithm, structured programming, data type, variables, constants, assignment, arithmetic operations, Boolean operators, sequence, selection, branching, definite and indefinite iteration, top down design, modular programming, subroutine, procedure, function, parameter, argument, pass by value, pass by reference, global and local variables, recursion, object oriented programming, class, object, method, attribute, inheritance, encapsulation, polymorphism	computational thinking, abstraction, algorithm, preconditions, greatest common divisor, modelling, simulation, preconditions, hierarchy charts, modularisation, trace table, concurrent processing, parallel processing, graph data structure, shortest path, exhaustive search, backtracking, data mining, Big Data, heuristics, rule of thumb, intractable problem, Travelling Salesman problem, performance modelling, pipelining, visualisation	algorithm, brute force method, divide and conquer, decrease and conquer recursion, recursive subroutine Big-O notation, linear, polynomial, exponential, logarithmic functions, permutation, time complexity linear search, binary search, binary tree search, bubble sort, insertion sort, merge sort, quick sort depth-first traversal, breadth-first traversal, pre-order and post-order tree traversal optimisation problem, Dijkstra's shortest path algorithm, A* algorithm
End of Half term assessment	End of topic Assessment AP1 assessment	Mock Exams	End of topic tests
Planned trips / Clubs / links			