Bishop Chadwick
Catholic Education Trust

Design & Technology, Food and Computer Science Department

Progression Models 2022

Computer Science

Progression Model Y7 Computer Science

PLEASE NOTE Year 7 have 2 learning modules, split between 3 assessment periods

Module Title: Collaborating Online Responsibly	Module Title: Collaborating Online Responsibly (final 3 lessons of module) Modelling Data – Spreadsheets (first 4 lessons of module)	Module Title: Modelling Data - Spreadsheets
Learning Intent for this module: • To be able to use, save and manage files on a computer • To know how to access and use cloud storage and email • To understand the risks of an online presence including risks to the self and data • To know ways to keep users physically, mentally and emotionally safe and become responsible participants in the online community • To know how to operate Teams as a remote learning tool	Learning Intent for this module: • Know the difference between hardware and software • Understand how hardware and software interact and communicate to allow users to create multisensory outcomes Learning Intent for this module: • Know how to use basic functions and formulas in Excel spreadsheets • Understand how and why these functions can be used in real life.	Learning Intent for this module: • Be able to select appropriate forms of data collection for specific purposes • Be able to analyse and present data using charts and graphs • Have the skills needed to present data in both paper and digital format using multiple applications • Understand how to evaluate data and data collection in a real-world scenario
Key Content to be learned: Computing 3.9 – Understand a range of ways to use technology safely, respectfully, responsibly and securely, including protecting their online identity and privacy; recognise inappropriate content, contact and conduct and know how to report concerns	Key Content to be learned: Computing 3.5 – Understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems. Computing 3.8 – Create, re-use, revise and repurpose digital artefacts for a given audience, with attention to trustworthiness, design and usability Computing 3.1 – Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems.	Key Content to be learned: Computing 3.7 – Undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting and analysing data and meeting the needs of known users.
Key tasks for this module: • Instructions on how to organise files and folders -successful use of 1 application (word/excel) and sending an email attachment (lesson 2) • Writing Like a Computer Scientist – The benefits and drawbacks of life online	Key tasks for this module: • Explain to new y7s how hardware and software work together (Multimedia Powerpoint) • Using basic functions in a spreadsheet quiz	Key tasks for this module: • Writing Like a Computer Scientist – Final evaluation of data analysis • End of Year Examination

Progression Model - Year 8 Computer Science

Module Title: Impact of Technology	Module Title: Introduction to Scratch	Module Title: Introduction to Python
Learning Intent for this module: This module has been designed to allow students to investigate and evaluate new and emerging technologies and their impact on our everyday life. It covers technology basics, internet safety. Scamming, hacking and phishing. Types and uses of storage, use of networks and cyber security. The fist module will allow students to be responsible, competent, confident and creative users of a range of technology	Learning Intent for this module: This module has been designed to introduce students to Scratch. Scratch uses a graphical programming language to help students learn to write code and engage in creative thinking. It provides students with the opportunity to develop an understanding of fundamental programming concepts such as variables, sequencing, selection and iteration. Learners will create their own subroutines, develop their understanding of decomposition and use lists. Build upon their problem-solving skills by working through a larger project.	Learning Intent for this Module: This module will introduce learners to text-based programming with Python which will further develop their understanding of arithmetic operations, selection, and iteration They will be investigating how data can be represented and processed using lists and strings, writing algorithms and pseudocode. Learners will grow in confidence as they work their way through the fundamentals of Python and work sequentially through tasks aimed at building programing skills
Key Content to be learned: Computing 3.5 – Understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems. Computing 3.9 – Understand a range of ways to use technology safely, respectfully, responsibly and securely, including protecting their online identity and privacy; recognise inappropriate content, contact and conduct and know how to report concerns	Key content to be learned: Computing 3.3 - use two or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures [for example, lists, tables or arrays]; design and develop modular programs that use procedures or functions. Computing 3.7 – undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting and analysing data and meeting the needs of known users.	Key Content to be learned: Computing 3.1 – Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems. Computing 3.2 - understand several key algorithms that reflect computational thinking [for example, ones for sorting and searching]; use logical reasoning to compare the utility of alternative algorithms for the same problem Computing 3.3 - use two or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures [for example, lists, tables or arrays]; design and develop modular programs that use procedures or functions.
Key tasks for this module: KT1 – Project work on keeping yourself safe online KT2 – Evaluate different types of computer storage KT3 – Explain what a network is and the pro's and con's of each	Key tasks for this module: KT1 – Using coding blocks in scratch to correctly animate a program KT2 – Collision detection skills	Key tasks for this module: KT1 – What is Python & key terms KT2 – Using the correct syntax to run a program KT3 – Create own program with variables KT4- END OF YEAR EXAMINATION

KT4 – Writing Like a Computer Scientist: Impact of Technology	KT3 – Create own animated game, evaluate & Improve KT4 – Writing Like a Computer Scientist: The impact of technology	

Progression Model Y9 Computer Science

Module 1: Computational Thinking & Binary	Module 2: Combined Project	Module 3: Intermediate Python
Learning Intent for this module: Learners will further develop skills to problem solve using computational thinking techniques of abstraction and decomposition to model the state and behaviour of real-world problems and physical systems. Learner will understand the use of simple Boolean logic [for example, AND, OR and NOT] and some of its uses in circuits and programming; understand how numbers can be represented in binary, and be able to carry out simple operations on binary numbers [for example, binary addition, and conversion between binary and decimal].	Learning Intent for this module: Learners to build on skills from Year 7 improving their knowledge on hardware and software components that make up computer systems, and how they communicate with one another and with other systems. Learners will be introduced to how images, sounds and media is stored and gain an understanding how quality links to file size. Learners will undertake a creative project that involves selecting, using, and combining multiple applications that includes analysing data, presenting information and evaluating the results for a given scenario.	Learning Intent for this module: Learners to build on skills from Year 8 improving their knowledge on the text-based programming language Python. Further develop their understanding of arithmetic operations, selection, and iteration using real life problems. Learners will Investigate how data can be represented and processed using lists and strings. Explore operations on sequences of data, that range from accessing an individual element to manipulating the entire sequence of data in files. Learners will study different types of validation and build sub routines in programs.
Key Content to be learned: Computing 3.1 – Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems. Computing 3.2 - understand several key algorithms that reflect computational thinking [for example, ones for sorting and searching]; use logical reasoning to compare the utility of alternative algorithms for the same problem Computing 3.4 – understand simple Boolean logic [for example, AND, OR and NOT] and some of its uses in circuits and programming; understand how numbers can be represented in binary, and be able to carry out simple operations on binary numbers [for example, binary addition, and conversion between binary and decimal].	Key Content to be learned: Computing 3.1 – Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems. Computing 3.5 – Understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems. Computing 3.6 – understand how instructions are stored and executed within a computer system; understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally, in the form of binary digits. Computing 3.7 – undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting	Key Content to be learned: Computing 3.1 – Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems. Computing 3.2 - understand several key algorithms that reflect computational thinking [for example, ones for sorting and searching]; use logical reasoning to compare the utility of alternative algorithms for the same problem Computing 3.3 - use two or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures [for example, lists, tables or arrays]; design and develop modular programs that use procedures or functions.

	and analysing data and meeting the needs of known users.	
Key tasks for this module: KT1:Writing Algorithms KT2:Binary conversion & Boolean Logic KT3: Writing Like a Computer Scientist – The Environmental Impact of Computers	Key tasks for this module: KT1:Hardware & Software KT2:Combined Project KT3: Writing Like a Computer Scientist – The Cultural Impact of computer Science.	Key tasks for this module: KT1:Python While Statement Exercises KT2:Python Validation Exercises KT3: Writing Like a Computer Scientist – The Impact of Digital Technology on a Wider Society.

Progression Model – Y10

Computer Science Y10 Module 1 (1.2 & 2.1)	Computer Science Y10 Module 2 (2.2 & 2.5)	Computer Science Y10 Module 3 (2.3 & 2.4)
Learning Intent for this module: To build on students understanding from KS3 of memory, ASCII, binary and denary with new learning covering virtual memory, character sets, Unicode, hexadecimal and compression. For students to build on prior knowledge on algorithmic thinking, develop skills in writing pseudocode and writing high level language programs. To understand how key sorting (bubble, merge & insertion) and searching (linear & binary) algorithms work prior to programming fundamentals unit.	Learning Intent for this module: To build on students programming skills from KS3. These form the basis for new learning covering nested IF's, data structures, sub programs, reading and writing to files. Links well with SQL learning to program in low level languages. While developing and strengthening their knowledge in learning to code, students will also look at the need for IDE's and independently tackle challenges on coding skills through a series of programming challenges.	Learning Intent for this module: Students will use a series of coding activities, tutorials and challenges to fully appreciate the need for testing computer programs using normal, boundary and erroneous values. Students will be able to build on their understanding of boolean operators and apply this to logic diagrams and truth tables.
Key Content to be learned: (1.2)The need for secondary storage including common types of storage and virtual memory: suitable storage devices and storage media for a given application and the advantages and disadvantages of these using characteristics: capacity, speed, portability, durability, reliability and cost. The units of data storage: Bit, Nibble (4 bits), Byte (8 bits), Kilobyte (1,000 bytes or 1 KB), Megabyte (1,000 KB), Gigabyte (1,000 MB), Terabyte (1,000 GB), Petabyte (1,000 TB). Investigate how data needs to be converted into a binary format to be processed by a computer. Look at data capacity and calculation of data capacity requirements. How to add two binary integers together (up to and	Key content to be learned: (2.2) To create and interpret the use of variables, constants, operators, inputs, outputs and assignments in computer programs. To use the three basic programming constructs used to control the flow of a program: sequence, selection and iteration (count- and condition-controlled loops) To use the common arithmetic operators in the creation of computer programs. Including the use of the common Boolean operators AND, OR and NOT. To create computer programs using the data types: integer, real, Boolean, character and string and casting	Key content to be learned: (2.3) To analyse defensive design considerations: o anticipating misuse o authentication To produce programs using Input validation To investigate Maintainability: Use of sub programs, Naming conventions, Indentation, Commenting. To study the purpose of testing and explain the different types of testing: iterative, final/terminal To identify syntax and logic errors in programs To select and use suitable test data: Normal, Boundary, Invalid/Erroneous

including 8 bits) and explain overflow errors which may occur. How to convert positive denary whole numbers into 2-digit hexadecimal numbers and vice versa How to convert binary integers to their hexadecimal equivalents and vice versa. To carry out binary shifts Define the term 'character-set' and the relationship between the number of bits per character in a character set, and the number of characters which can be represented, e.g.: ASCII, Unicode. Investigate how an image is represented as a series of pixels, represented in binary. Look at the relationship between the number of bits per character in a character set, and the number of characters which can be represented, e.g.: ASCII, Unicode. Investigate how an image is represented as a series of pixels, represented in binary, metadata and the effect of colour depth and resolution on the the quality of the image and the size of an image file. Investigate how sound can be sampled and stored in digital form. The effect of sample rate, duration and bit depth on: The playback quality and the size of a sound file. Investigate a need for compression, the different types of compression: lossy and lossless. (2.1) Computational thinking: Looking at the Principals of computational thinking: abstraction, decomposition and algorithmic thinking. To Identify the inputs, processes, and outputs for a problem, looking at the complexity of computer system examples. Creating structure diagrams To create, interpret, correct, complete, and refine algorithms using: pseudocode, flowcharts, reference language / high-level programming languages. To Identify common errors in Trace tables. To create and interpret standard searching algorithms: binary search and linear search. To	Be able to read and interpret the use of basic string manipulation and the use of basic file handling operations: open, read, write, close. To interpret and use basic records to store data and write basic SQL to search for data. To use arrays (or equivalent) when solving problems, including both one-dimensional (1D) and two-dimensional (2D) arrays. To be able to use sub programs (functions and procedures) to produce structured code and Random number generation (2.5) Characteristics and purpose of different levels of programming language: High-level languages & Low-level languages To recognise the purpose of translators and identify the characteristics of a compiler and an interpreter. To be able to use common tools and facilities available in an integrated development environment (IDE): o editors, error diagnostics, run-time environment, translators.	To refine algorithms. (2.4) Boolean Logic, to use a simple logic diagrams using the operators AND, OR and NOT, use truth tables and combine Boolean operators using AND, OR and NOT Gates. Gain knowledge on applying logical operators in truth tables to solve problems.

create and interpret standard sorting algorithms: bubble sort, merge sort and insertion sort		
Key tasks for this module: KT1:Secondary storage KT2:Binary conversion and addition KT3: Pseudocode and flowcharts KT4: Computational thinking and algorithms	Key tasks for this module: KT1:Simple programming task KT2:Complex program task KT3:Boolean operators KT4:Compiler and interpreters	Key tasks for this module: KT1:Boolean operators KT2:Logic diagrams KT3:Truth tables KT4:Problem solving task

Progression Model – Y11

Computer Science Y11 Module 1 (1.1 & 1.3)	Computer Science Y11 Module 2 (1.4 & 1.5)	Computer Science Y11 Module 3 (1.6)
Learning Intent for this module: At KS3 students were taught some basic knowledge on the internal working of a computer system focusing on the CPU and its components alongside external peripherals, RAM & ROM. For GCSE students require a deeper knowledge on how a computer works and what common characteristics affect performance. Students have studied at KS3 the use of technology in the home and school looking at embedded systems. Students have a basic understanding from KS3 of networks, new learning covering the factors that affect the performance of networks and the different roles of computers in network topologies, along with encryption, IP & MAC addressing and layers.	Learning Intent for this module: Students have an understanding from KS3 of hardware and software with new learning covering file management, utility software, defragmentation, file management and data compression. Students at KS3 have an awareness of forms of attack on computers and networks, new learning covering brute force attacks, denial of service attacks, data interception and theft, and also the concept of SQL injection. Students have an understanding from KS3 of passwords, encryption and physical security. Introduce students to more network security methods, operating systems and utility software.	Learning Intent for this Module: Use students existing knowledge of data protection, copyright law and computer misuse act to deliver new learning more in depth knowledge needed for GCSE covering data protection, computer misuse and software licensing. Students have studied some ethical and environmental impacts of digital technology on society now need to understand the wider implication of legal, cultural and privacy issues that computers have had on society.
Key Content to be learned: (1.1) The purpose of the CPU, the common CPU components and their function including : ALU, Arithmetic Logic Unit) , CU (Control Unit) , Cache, Registers Von Neumann architecture: MAR (Memory Address Register), MDR (Memory Data Register), Program Counter, Accumulator.	Key Content to be learned: (1.4)Investigate forms of attack on computers: Malware, Social engineering, e.g. phishing, people as the 'weak point'. Brute-force attacks, Denial of service attacks, Data interception and theft, The concept of SQL injection. Investigate Common prevention methods: Penetration testing, Anti-malware software, Firewalls	Key Content to be learned: (1.6) Investigate the impacts of digital technology on wider society including: o Ethical issues o Legal issues o Cultural issues o Environmental issues o Privacy issues

How common characteristics of CPUs affect their performance: clock speed, cache size, number of cores. The purpose and characteristics of embedded systems including examples of embedded systems. The need for primary storage The difference between RAM and ROM The purpose of ROM in a computer system The purpose of RAM in a computer system Virtual memory (1.3) The types of computer networks: LAN (Local Area Network), WAN (Wide Area Network). The factors that affect the performance of networks and the different roles of computers in a client-server and a peer-to-peer network. The hardware needed to connect stand-alone computers into a Local Area Network: wireless access points, routers, switches, NIC (Network Interface Controller/Card), transmission media. Investigate the internet as a worldwide collection of computer networks: DNS (Domain Name Server), hosting, the cloud, web servers and clients. Learn about star and mesh network topologies, their modes of connection: Wired, Ethernet, Wireless, Wi-Fi, Bluetooth. Develop knowledge on Encryption, IP addressing and MAC addressing, Network standards and Common protocols.	User access levels, Passwords, Encryption, Physical security. (1.5)The purpose and functionality of operating systems including: user interface, memory management and multitasking, peripheral management and drivers, user management, file management. The purpose and functionality of utility software including: encryption software, defragmentation and data compression.	Investigate the current legislation relevant to Computer Science: o The Data Protection Act 2018 o Computer Misuse Act 1990 o Copyright Designs and Patents Act 1988 o Software licences (i.e. open source and proprietary)
Key tasks for this module: KT1:Central processing unit and factors KT2:Computer networks KT3:The Internet KT4:Network topologies	Key tasks for this module: KT1:Forms of attack on computers KT2: Common prevention methods KT3:Operating systems KT4:Utility software and encryption	Key tasks for this module: KT1: Ethical issues & Legal issues KT2: Cultural issues & Environmental issues KT3: Computer Laws KT4:Open V Closed software

Progression Model Y12

Computer Science Year 12 Module 1	Computer Science Year 12 Module 2	Computer Science Year 12 Module 3)
Learning Intent for this module: Build on students understanding of programming from KS3/4, introduce students to Defold & Python and Assembly Language. To learn basic programming techniques in Python. How computers can be used to solve problems and programs can be written to solve them Investigate the functions of a processor. What are the components of a CPU, and what do they do? Explore the differences between and uses of CISC and RISC processors. Learn how to use two's complement to represent negative numbers in binary. Extend student knowledge of binary number to include subtraction.	Learning Intent for this module: Understand how to represent positive and negative real numbers using normalised floating-point representation. Explore memory management (paging, segmentation and virtual memory). Study a range of SDLC's. Study a range of testing strategies, including black and white box testing and alpha and beta testing. Gain a understanding of database, including normalisation. Build on students understanding of programming in Python to an advanced level, introduce students to HTML & LMC.	Learning Intent for this Module: Explore the individual (moral), social (ethical) and cultural opportunities and risks of digital technology. Introduce stacks and queues. Investigate assembly language (including following and writing programs with Little Man Computer). Build on students understanding of algorithms by coding a bubble sort, insertion sort, binary search and linear search. Look at software development in relation to the software development life cycle and identify different methods used in industry.
Key Content to be learned: **1.1.1 Structure and function of the processor** The Arithmetic and Logic Unit; ALU, Control Unit and Registers (PC, ACC, MAR, MDR, CIR). Buses: data, address and control: how this relates to assembly language programs. (b) The Fetch-Decode-Execute Cycle; including its effects on registers. (c) The factors affecting the performance of the CPU: clock speed, number of cores, cache. **1.1.2 Types of processor** (a) The differences between and uses of CISC and RISC processors. (b) GPUs and their uses (including those not related to graphics). (c) Multicore and Parallel systems. **1.1.3 Input, output and storage**	Key content to be learned: **1.4.3 Boolean Algebra** Define problems using Boolean logic. b) Manipulate Boolean expressions, including the use of Karnaugh maps. **1.2.1 Systems Software** (a) The need for, function and purpose of operating systems. (b) Memory Management (paging, segmentation and virtual memory). (c) Interrupts, the role of interrupts and Interrupt Service Routines (ISR), role within the Fetch-Decode-Execute Cycle. (d) Scheduling. (e) Distributed, embedded, multi-tasking, multi-user and Real Time operating systems. (f) BIOS. (g) Device drivers. (h) 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.	Key Content to be learned: **1.4.2 Data Structures** (a) Arrays (of up to 3 dimensions), records, lists, tuples. (b) The following structures to store data: linked-list, graph (directed and undirected), stack, queue, tree, binary search tree, hash tables. **1.2.3 Software Development** a) Understand the waterfall lifecycle, agile methodologies, extreme programming, the spiral model and rapid application development. (b) The relative merits and drawbacks of different methodologies and when they might be used. (c) Writing and following algorithms. **Project Analysis** 2.1.1 Thinking abstractly (a) The nature of abstraction. (b) The need for abstraction. (c) The differences between an abstraction and

a) How different input, output and storage devices can be applied to the solution of different problems. (b) The uses of magnetic, flash and optical storage devices. (c) RAM and ROM. (d) Virtual storage **1.4.1 Data Types** (a) Primitive data types, integer, real/floating point, character, string and Boolean. (b) Represent positive integers in binary. (c) Use of sign and magnitude and two's complement to represent negative numbers in binary. (d) Addition and subtraction of binary integers. (e) Represent positive integers in hexadecimal. (f) Convert positive integers between binary hexadecimal and denary. (g) Representation and normalisation of floating point numbers in binary. (h) Floating point arithmetic, positive and negative numbers, addition and subtraction. **2.2.1 Programming techniques** (a) Programming constructs: sequence, iteration, branching. (b) Recursion, how it can be used and compares to an iterative approach. (c) Global and local variables.	**2.2Programming Techniques** (a) Features that make a problem solvable by computational methods. (b) Problem recognition. (c) Problem decomposition. (d) Use of divide and conquer. (e) Use of abstraction. (f) Learners should apply their knowledge of: • backtracking • data mining • heuristics • performance modelling • pipelining • visualisation to solve problems. **1.5.1 Computing related legislation** (a) The Data Protection Act 1998. (b) The Computer Misuse Act 1990. (c) The Copyright Design and Patents Act 1988. (d) The Regulation of Investigatory Powers Act 2000. 1**.2.2 Applications Generation** (a) The nature of applications, justifying suitable applications for a specific purpose. (b) Utilities. (c) Open source vs closed source. (d) Translators: Interpreters, compilers and assemblers. **2.2.2 Computational methods** (a) Features that make a problem solvable by computational methods. (b) Problem recognition. (c) Problem decomposition. (d) Use of divide and conquer. (e) Use of abstraction. **1.3.1 Compression, Encryption and Hashing** (a) Lossy vs Lossless compression. (b) Run length encoding and dictionary coding for lossless compression **1.3.2 Databases** (a) Relational database, flat file, primary key, foreign key, secondary key, entity relationship modelling, normalisation and indexing. See appendix 5f. (b) Methods of capturing, selecting, managing and exchanging data. (c) Normalisation to 3NF. (d) SQL – Interpret and modify. See appendix 5d. (e) Referential integrity. (f) Transaction processing, ACID (Atomicity, Consistency, Isolation, Durability), record locking and redundancy. 1**.3.3 Networks** (a) Characteristics of networks and the importance of protocols and standards. (b) The internet structure: • The TCP/IP Stack. • DNS • Protocol layering. • LANs and WANs. • Packet and circuit switching. (c) Network security and threats, use of firewalls.	reality. (d) Devise an abstract model for a variety of situations. 2.1.2 Thinking ahead (a) Identify the inputs and outputs for a given situation. (b) Determine the preconditions for devising a solution to a problem. (c) The nature, benefits and drawbacks of caching. (d) The need for reusable program components. 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 the steps needed to solve a problem. (d) Identify sub-procedures necessary to solve a problem. 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. **2.3.1 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. (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. (e) Algorithms for the main data structures, (stacks, queues, trees, linked lists, depth-first (post-order) and breadth-first traversal of trees).

	1.5.2 Moral and ethical Issues The individual moral, social, ethical and cultural opportunities and risks of digital technology: • Computers in the workforce. • Automated decision making. • Artificial intelligence. • Environmental effects. • Censorship and the Internet. • Monitor behaviour. • Analyse personal information. • Piracy and offensive communications. • Layout, colour paradigms and character sets.	
Key tasks for this module: KT1: System Software KT2: Programme task KT3: Types of processors KT4: Input, Output and Storage Device (Ext W) KT5: Binary and data types	Key tasks for this module: KT1: Boolean Algebra KT2:Programming task KT3: Data Types KT4:Databases(Ext W) KT5:Networks	Key tasks for this module: KT1: Data structures KT2: Software development (Ext W) KT3: Algorithmic Thinking KT4:Programming task KT5:Producing algorithms

Progression Model Y13

Computer Science Year 13 Module 1	Computer Science Year 13 Module 2	Computer Science Year 13 Module 3
Learning Intent for this module: Students start their project which they will work on throughout the year beginning with Analysis of the project. Students will recap AS syllabus of the course and then further develop knowledge in particular areas of the syllabus. Student will explore new areas of the curriculum in OOP and gain an understanding of classes, objects, methods, attributes, inheritance, encapsulation and polymorphism. Further develop a student's knowledge of compression techniques. Study symmetric and asymmetric encryption. Explore databases including normalisation and SQL to modify and interpret data. Explore search engine indexing PageRank algorithms.	Learning Intent for this module: Students complete the design of their project incorporating theory knowledge learnt in other parts of the syllabus. Students will code and write up their development story, with the use of testing to inform their development of a solution at various sections of their project. New knowledge learnt will be further developing their own understanding of using structures to store data: linked list, graph (directed and undirected), tree, binary search tree, hash tables. Applying their knowledge of abstraction to backtracking, data mining and heuristics. Use methods to determine the efficiency of different algorithms e.g. Big O notation. Investigate the logic associated with D type flip flops, half and full adders.	Learning Intent for this Module: Students complete the evaluation of their project incorporating theory knowledge learnt in other parts of the syllabus. Students to study concurrent processing. To investigate the use of algorithms to describe problems and standard algorithms Investigate recursion and how it can be used and compares to an iterative approach. Explore standard algorithms (Bubble sort, insertion sort, merge sort, quick sort, Dijkstra's shortest path algorithm, A* algorithms, binary search and linear search).

Key Content to be learned: **3.1.1 Project Analysis**(a) Describe and justify the features that make the problem solvable by computational methods. (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). 3.1.3 Research the problem (a) Research the problem and solutions to similar problems to identify and justify suitable approaches to a solution. (b) Describe the essential features of a computational solution explaining these choices. (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). (b) Identify and justify measurable success criteria for the proposed solution **1.1.1 Structure and function of the processor** Recap (a) to (c) then (d) The use of pipelining in a processor to improve efficiency. (e) Von Neumann, Harvard and contemporary processor architecture. **1.1.2 Types of processor** Recap (a) to (b) (c) Multicore and Parallel systems. **1.2.2 Applications Generation** Recap a to d. (e) Stages of compilation (lexical analysis, syntax analysis, code generation and optimisation). (f) Linkers and loaders and use of libraries **1.2.4 Types of Programming Language** (a) Need for and characteristics of a variety of programming paradigms. (b) Procedural languages. (c) Assembly language (including following and writing simple programs with the Little Man Computer instruction set). See appendix 5d. (d) Modes of addressing memory (immediate, direct, indirect and indexed). (e) Object-oriented languages (see appendix 5d for pseudocode style) with an understanding of classes,	Key content to be learned: **3.3 Developing the solution** (25 marks) 3.3.1 Iterative development process (a) Provide annotated evidence of each stage of the iterative development process justifying any decision made. (b) Provide annotated evidence of prototype solutions justifying any 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. **1.4.2 Data Structures** Recap (a) and (b) then (c) How to create, traverse, add data to and remove data from the data structures mentioned above. (NB this can be either using arrays and procedural programming or an object-oriented approach). **1.4.3 Boolean Algebra** Recap (a) and (b) then (c) Use the following rules to derive or simplify statements in Boolean algebra: De Morgan's Laws, distribution, association, commutation, double negation. (d) Using logic gate diagrams and truth tables. See appendix 5d. (e) The logic associated with D type flip flops, half and full adders. **1.5.1 Computing related legislation** Recap (a) to (d) **1.5.2 Moral and ethical Issues** The individual moral, social, ethical and cultural opportunities and risks of digital technology: • Computers in the workforce. • Automated decision making. • Artificial intelligence. • Environmental effects. • Censorship and the Internet. • Monitor behaviour. • Analyse personal information. • Piracy and offensive communications. • Layout, colour paradigms and character sets. **2.1.1 Thinking abstractly** Recap (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.	Key Content to be learned: **3.4 Evaluation** (20 marks) 3.4.1 Testing to inform evaluation (a) Provide annotated evidence of testing the solution of robustness at the end of the development process. (b) Provide annotated evidence of usability testing (user feedback). 3.4.2 Success of the solution (a) Use the test evidence from the development and post development process to evaluate the solution against the success criteria from the analysis. 3.4.3 Describe the final product (a) Provide annotated evidence of the usability features from the design, commenting on their effectiveness. 3.4.4 Maintenance and development (a) Discuss the maintainability of the solution. (b) Discuss potential further development of the solution. **2.3.1 Algorithms** Recap (a) and (b) then look at the suitability of different algorithms for a given task and data set, in terms of execution time and space. (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. (e) Algorithms for the main data structures, (stacks, queues, trees, linked lists, depth-first (post-order) and breadth-first traversal of trees). (f) Standard algorithms (bubble sort, insertion sort, merge sort, quick sort, Dijkstra's shortest path algorithm, A* algorithm, binary search and linear search). Revision: Past exam papers. Quizzes and Videos.

objects, methods, attributes, inheritance, encapsulation and polymorphism **1.3.1 Compression, Encryption and Hashing** (a) Lossy vs Lossless compression. Recap. (b) Run length encoding and dictionary coding for lossless compression. (c) Symmetric and asymmetric encryption. (d) Different uses of hashing. **3.2 Design of the solution** (15 marks) 3.2.1 Decompose the problem (a) Break down the problem into smaller parts suitable for computational solutions justifying any decisions made. 3.2.2 Describe the solution (a) Explain and justify the structure of the solution. (b) Describe the parts of the solution using algorithms justifying how these algorithms form a complete solution to the problem. (c) Describe usability features to be included in the solution. (d) Identify key variables / data structures / classes justifying choices and any necessary validation. 3.2.3 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 test data. **1.3.2 Databases** Recap (a) and (b) then (c) Normalisation to 3NF. (d) SQL – Interpret and modify. See appendix 5d. (e) Referential integrity. (f) Transaction processing, ACID (Atomicity, Consistency, Isolation, Durability), record locking and redundancy. **1.3.3 Networks** Recap (a) and (b) then (c) Network security and threats, use of firewalls, proxies and encryption. (d) Network hardware. (e) Client-server and peer to peer. **1.3.4 Web Technologies** (a) and (b) then (c) PageRank algorithm. (d) Server and client side processing. **1.4.1 Data Types** Recap (a) to (f) then (g) Representation and normalisation of floating point numbers in binary. (h) Floating point arithmetic, positive and negative numbers, addition and	**2.1.2 Thinking ahead** (a) Identify the inputs and outputs for a given situation. (b) Determine the preconditions for devising a solution to a problem. (c) The nature, benefits and drawbacks of caching. (d) The need for reusable program components **2.1.3 Thinking procedurally** Recap (a) Identify the components of a problem. (b) Identify the components of a solution to a problem. (c) Determine the order of the steps needed to solve a problem. (d) Identify sub-procedures necessary to solve a problem. **2.1.4 Thinking logically** Recap (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. **2.1.5 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. **2.2.2 Computational methods** Recap (a) to (d) then (e) Use of abstraction. (f) Learners should apply their knowledge of: • backtracking • data mining • heuristics • performance modelling • pipelining • visualisation to solve problem.	

subtraction. (i) Bitwise manipulation and masks: shifts, combining with AND, OR, and XOR. (j) How character sets (ASCII and UNICODE) are used to represent text.		
Key tasks for this module: KT1:CISC, RISC & Pipelining KT2:Types of programming languages (Ext W) KT3:Analysis of Project.(CW) KT4:Databases KT5:Networks	Key tasks for this module: KT1:Design of Project KT2:Implementation story KT3:DeMorgans Law KT4:Computer Laws (Ext W) KT5:Implentation of Project hand in	Key tasks for this module: KT1:Project Evaluation KT2: Project code KT3: Final project hand in