Computing Curriculum Overview: Key Stage 3 to Key Stage 5

The Appleton School

September 2023

Next Review July 2024

Subject: Computing

Curriculum Intent

Department Vision

Computing has deep links with mathematics, science and design and technology and provides insights into both natural and artificial systems. The Computer Science Area and Computing curriculum aims to provide all learners with a high-quality computing education, equipping learners with the ability to use computational thinking and creativity to understand and change the world. The curriculum reflects high expectations, acknowledging our more able learners, while ensuring it is inclusive and providing the appropriate levels of support. The curriculum has been designed to include experiences of Computer Science, Digital Literacy, Information Technology and the Digital Media aspects of Computing, ensuring learners are prepared for their next steps, if they choose to opt for these subjects at Key Stage 4 and Key Stage 5, to progress to higher levels of study or to a professional career and for the future workplace as active participants in a digital world.

The national curriculum for computing aims to ensure that all learners:

- can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation
- can analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems
- can evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems
- are responsible, competent, confident and creative users of information and communication technology.

With a combination of teacher led delivery, a module-based approach, independent research and programming projects, student demonstration, teamwork, extension and extra-curricular activities, competition and presentation activities, we aim to support learners to find their passion for learning and computing and to complement the investigative, problem solving and employability skills that they are developing in other subject areas.

Students' Vision

- to develop you as a Computer Scientist
- to provide you the skills to analyse, deconstruct and solve real-world problems in computational terms, looking at them from a logical stance
- to enable you to evaluate and apply information technology to solve problems
- to develop you as a responsible, competent, confident and creative users of information and communication technology
- to allow you to explore the social, moral, legal and ethical impact of computers in society
- to provide you with an insight into how computing and the skills, knowledge and understanding you gain will be useful in your chosen next steps
- to complement and enhance your employability skills
- to prepare you in being an active participant in a rapidly developing digital world

What are your aims linked to the curriculum (National Curriculum and specification criteria)

At Key Stage 3, learners will be taught to:

- design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
- understand several key algorithms that reflect computational thinking; use logical reasoning to compare the utility of alternative algorithms for the same problem
- use two or more programming languages, at least one of which is textual, to solve a variety of computational problems; make use of data structures; design and develop modular programs that use procedures or functions
- understand simple Boolean logic 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
- understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems
- 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
- 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
- create, re-use, revise and re-purpose digital artefacts for a given audience, with attention to trustworthiness, design and usability
- understand a range of ways to use technology safely, respectfully, responsible and securely, including protecting their online identity and privacy; recognise inappropriate content, contact and conduct and know how to report concerns.

At Key Stage 4, learners should be taught to:

- develop their capability, creativity and knowledge in computer science, digital media and information technology
- develop and apply their analytic, problem-solving, design, and computational skills
- understand how changes in technology affect safety, including new ways to protect their online privacy and identity, and how to identify and report a range of concerns.

The OCR GCSE (9-1) Computer Science specification will encourage learners to:

- understand and apply the fundamental principles and concepts of Computer Science, including abstraction, decomposition, logic, algorithms and data representation
- analyse problems in computational terms through practical experience of solving such problems, including designing, writing and debugging programs
- think creatively, innovatively, analytically, logically and critically
- understand the components that make up digital systems, and how they communicate with one another and with other systems
- understand the impacts of digital technology to the individual and to wider society
- apply mathematical skills relevant to Computer Science.

At Key Stage 5, the OCR A-Level Computer Science is followed. The aims of this specification are to enable learners to develop:

- an understanding of and ability 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
- the capacity for thinking creatively, innovatively, analytically, logically and critically
- the capacity to see relationships between different aspects of Computer Science
- mathematical skills
- the ability to articulate the individual (moral), social (ethical), legal and cultural opportunities and risks of digital technology.

How is the curriculum delivered?

Computing at Key Stage 3 is allocated one lesson per week and the curriculum has been divided into 18 modules of work over the three years – one per each half term (approximately). Across the Computing modules, each of the above bullet points for Key Stage 3 is covered at least four times over the three years – see mapping document on page 23; therefore, skills and knowledge are revisited throughout the Key Stage and providing learners with the opportunities to apply prior learning to new contexts so that they are able to develop a depth of knowledge that is rich. The modules cover a range of theoretical and practical aspects which are balanced across each year, and also reflect an increase in challenge across each year, for example moving from block based to textual based programming.

GCSE Computer Science is usually delivered as a 2-year course (Years 10 to 11) and allocated 3 lessons per week. The specification is divided into two components. Component 01 – Computer systems covers 6 sub topics and Component 02 – Computational thinking, algorithms and programming covering 5 sub topics. The details of these sub topics are identified later in this document and within the schemes of work. The curriculum has been based on OCR's scheme of work and suggestion of delivering unit by unit, but mixing between the components with a mixture of theoretical and practical activities. The curriculum has been developed so that skills and knowledge are revisited continuously throughout Key Stage 4, through a combination of theoretical work and practical programming skills and activities, until external examinations are completed and the learners are fully prepared for their next steps in their education. This regular retrieval practice will help build confidence and allow them to connect their prior knowledge to new learning. Due to a change in the curriculum model for this academic year (2022/23) – we currently have groups in Year 10 following a 3-year course and a group in Year 11 following a 1-year course (over 11 lessons per fortnight).

A-Level Computer Science is delivered over 2 years and allocated 9 lessons per fortnight. There is a large emphasis on independent, self-learning, especially where the topics cover computer coding / learning a range programming languages. The specification is divided into three components. Component 01- Computer systems covers 5 sub topics, Component 02 – Algorithms and programming covers 3 sub topics and Component 03 – the Non-Exam Assessment (NEA) Programming Project. In Year 12, learners start with a basic introduction to cover any prior learning needed and is set as a bridging activity – providing a concrete foundation to build upon. Learners then examine data types and programming techniques alongside computational thinking. The aim is to have learners programming to an acceptable level by the end of the first term. The resulting skills, knowledge and understanding will help to underpin their work in the Programming Project (NEA). During the second term the focus changes to system architecture, encryption and software development. The final term we come back to programming with algorithms in readiness for the NEA.

In Year 13, the main focus is continuing the building of the relevant programming skills and the ability to use algorithms to describe and solve problems. This will then be used to aid learners in the completion of the independent Programming Project (NEA). Again, there will be regular retrieval practise to help build confidence and allow learners to connect their prior knowledge to new learning so they are fully prepared for the external examinations and their next steps. During the final year learners will also focus on exchanging data (networking and databases), Boolean algebra and legal, cultural and ethical issues around the use of technology.

How is the curriculum assessed?

At Key Stage 3, assessments are completed at the end of each module to assess the learner's key knowledge and / or skills that have been developed through that module – these will be either through an end of module test or the assessment of a portfolio and end product produced by the learners. At Key Stage 3, learners are issued with a Self / Teacher Assessment grid at the beginning of the module. This informs the learners of the skills, knowledge and understanding that they need to develop and demonstrate at each grading level and allows themselves and the teacher to track their progress towards their target and beyond throughout the module. It helps to inform next steps and future teaching and learning. Learners are also invited to add their comment to the assessment sheet, considering What Went Well (WWW) and Even Better If (EBI). All of the assessment data is collected and recorded in the group folder and is used to inform when making progress judgements for data drops.

As well as the module assessments, there are Baseline tests that are done at the beginning of each year and during the exam weeks, determined by the school, for each year group. These Baseline tests are done through an online, automated system and provide an analysis of performance – strengths and areas for improvement on an individual and group basis. They also test a range of skills, knowledge and understanding across all the aims of the Key Stage 3 curriculum.

At Key Stage 4, all learners will sit a Baseline test to assess their current level of skills, knowledge and understanding. These will then be revisited at key points throughout the two years of the course to monitor progress. Assessments are completed at appropriate points in relation to the delivery of a topic or group of topics, where learners' key skills and knowledge have been developed through the topic and they are assessed through the completion of an end of topic test. The topic tests will make use of a combination of past exam questions and tests provided through the exam board and paid for resources. The exam board's grade boundaries for Computer Science will be used to give learners an indication of their performance against their target on that topic. Practical activities will be used to assess their application of the theory, especially important in the assessment of Component 02. Self-review of What Went Well (WWW) and Even Better If (EBI) and teacher feedback will be used to help learners to develop their skills and knowledge – to make progress and improve, and then will be given opportunities to respond by working on the question again or to work on an extended stretch and challenge question. At key points throughout the academic year, determined by the school, mock examinations will take place. These examinations will test skills and knowledge across the topics, revisiting topics throughout the year, and will be assessed using examination mark schemes and grade boundaries, ensuring a consistency of approach. All of the assessment data is collected and recorded, on SIMS for mock examinations and topic tests, and is used to inform when making progress judgements for data drops. All of this will be in preparation for the terminal examinations that are taken at the end of the course. These examinations are:

- Two written, external papers – one for each component; consisting of multiple choice, short response and extended response questions
- Worth 50% each of the total GCSE
- 80 marks available
- 1 hour 30 minutes in length
- Non-calculator papers

The paper for Component 02 will also assess the learner's ability to write or refine algorithms in either the OCR Exam Reference Language or a high-level language they are familiar with. The practical activities done throughout the course will support the theory and give learners the experience of these types of questions.

At Key Stage 5, assessments are completed at the end of the delivery of each topic, where learners' key skills and knowledge have been developed through the topic and they are assessed through the completion of an end of topic test or presentation to the group. Each half term, an exam (mock) is set on previous and current learning. The topic tests and mock exams will make use of a combination of past exam questions and tests provided through the exam board and paid for resources. The exam board's grade boundaries for Computer Science will be used to give learners an indication of their performance against their target on that topic. The learners are also given an exemplar answer sheet for each exam to assist in the development of their responses. At the end of Year 12, an AS Level Computer Science paper is sat by learners. The results of this, the topic tests and presentations and the half termly assessments give an indication of progress in the first year. In Year 13, topics are revisited over the year and best practice and exemplar answers are made for the extended, structured response questions. The main focus is on the Programming Project (NEA) worth 20 % of the final grade and is internally assessed and externally moderated. Mock examinations are taken during the year, as per the school calendar. All of the assessment data is collected and recorded, on SIMS for mock examinations and a Google tracking sheet for the topic tests, and is used to inform when making progress judgements for data drops. All of this will be in preparation for the terminal examinations that are taken at the end of the course. These examinations are:

- Two written, external papers – one for each component; consisting of short response, long response and extended response questions
- Worth 40% each of the total A-Level
- 140 marks available
- 2 hours 30 minutes in length
- Non-calculator papers

The remaining 20% of the total A-Level comes from the Programming Project (NEA), where learners will develop their own idea by analysing the problem, designing a solution, developing a solution and evaluating the result.

At all key stages, a variety of questioning styles will be used to help check and assess the learners' understanding so that gaps can be addressed before moving on. Especially process style questioning at later key stages, which will help deepen their knowledge and develop a higher level of thinking so that learners can think through what they know and apply it to a question, especially those that require a structured response.

How is the curriculum enriched (through speakers/visits/clubs) to generate a love of learning?

At all key stages, the curriculum is enriched predominately through the application of their skills, knowledge and understanding to real-world problems and to help solve problems that often don't exist. At Key Stage 3, there are a number of extra-curricular clubs / activities, including a Code Club, CyberFirst Girls' Competition and a KS3 Club. These include project based activities, including mini-builds for micro:bits and robots. At Key Stage 4, extra-curricular clubs are offered by staff to support revision and development of knowledge. Students are encouraged to support the activities at Key Stage 3 clubs. At Key Stage 5, videos and online lectures are watched and discussed, including those offered by Isaac Computer Science, the Richard Dimbleby Lectures and the IP / Digital Transformation Expo. Year 12s participate in a work experience programme; these placements can be linked to Computing / Computer Science to support their interest in working in the field or advancing their education. Year 13s are encouraged to help run clubs and assist younger years with coding. Students that are keen programmers can be linked with community groups to help design solutions for them such as websites, bespoke software and databases.

The aim for the future is:

- to promote national Computer Science competitions to age-appropriate groups
- to continue to nominate learners for the Royal Institute Computer Science Master Classes
- to facilitate a visit to Bletchley Park and / or the National Museum for Computing
- to facilitate visits to Computing / Computer Science / Coding and Programming related venues / experiences
- to organise clubs / extra-curricular projects based on code breaking, Raspberry Pi and STEM activities linked to real-world examples.

What skills and knowledge do we expect learners bring with them from Key Stage 2 to Year 7?

Learners should have been taught to:

- design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts
- use sequence, selection and repetition in programs; work with variables and various forms of input and output
- use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs
- understand computer networks including the internet; how they can provide multiple services such as the world wide web; and the opportunities they offer for communication and collaboration
- use search technologies effectively, appreciate how results are selected and ranked, and be discerning in evaluating digital content
- select, use and combine a variety of software (including internet services) on a range of digital devices to design and create a range of programs, systems and content that accomplish given goals, including collecting, analysing, evaluating and presenting data and information
- use technology safely, respectfully and responsibly; recognise acceptable / unacceptable behaviour; identify a range of ways to report concerns about content and contact.

What skills and knowledge do learners bring with them from Year 7 to Year 8?

Learners will cover all the key criteria from the Key Stage 3 National Curriculum, as identified previously. More specifically in Year 7:

- File management and security
- Appropriate conduct whilst operating online
- E-Safety including cyber-bullying phishing and how to action concerns
- Online profiles and social media including dangers and drawbacks, keeping data safe
- Functionality and operation of email and search engines, including their effective use
- Basic computer architecture, including input – process – output and the Fetch – Decode – Execute Cycle
- Binary – data representation of numbers, binary to denary conversion and vice versa, binary addition
- Binary / ASCII – data representation of characters
- Storage devices and data representation
- History of communication devices and new and emerging technologies
- Identify everyday situations where computer control is used and the sensors used by control systems
- Break down problems in the form of flowcharts
- Design algorithms to solve problems by producing flowchart-based solutions for control systems that include sequences and loops, subroutines and variables
- Design algorithms to solve problems by planning and developing a game using a block programming language that includes a range of programming constructs
- Develop, test and refine solutions for a given purpose
- Reverse engineer existing solutions to help facilitate their own planning and development
- Write and develop programs in a high-level programming language, using a range of programming constructs and taking into account the importance of the process, syntax, testing and debugging (using both emulators and physical hardware to run the programs)
- Binary – data representation of bitmap and vector images
- Skills in design, photo-editing and image manipulation to create graphics for a given audience and purpose
- Collaborative working in order to give and receive feedback on work done by themselves and others.

What skills and knowledge do learners bring with them from Year 8 to Year 9?

Learners will cover all the key criteria from the Key Stage 3 National Curriculum, as identified previously. More specifically in Year 8:

- Name the major Acts concerning computer use – Computer Misuse Act, Data Protection Act and GDPR, Copyright Law, Health and Safety
- Recognise and describe a range of threats to data – phishing, email frauds, hacking, data harvesting, identity theft
- Recognise and describe the dangers of putting personal data on social networking sites
- Describe and implement a range of ways of protecting personal data and computer systems
- Environmental concerns relating to the increased use of technology
- The basic principles and architecture of local and wide area networks, including examples
- Network topologies including their benefits and constraints
- Client-server networks, structure / hardware, advantages and disadvantages
- Describe the Internet and the World Wide Web and the difference between them
- Communication over networks and security of communications – encryption
- Skills in HTML and CSS to create a basic but responsive website in a text editor, designing the website
- Design and create a website for a given audience and purpose
- How websites use forms to collect data and the privacy issues relating to data collection and use
- Develop, test and refine solutions for a given purpose
- Write and develop programs in a high-level programming language, using a range of programming constructs and taking into account the importance of the process, syntax, testing and debugging
- Recognise and analyse code being able to describe its function
- Show discrimination in selecting assets such as still images, sound effects and background music for a video product for a given audience and purpose
- Combine music and sound effects with moving and still images, transitions and video effects into a video product for a given audience and purpose.
- Binary – data representation of sound
- Skills in sound editing, mixing and sound manipulation to create a radio advert / podcast for a given audience and purpose
- How sound effects are created
- Collaborative working in order to give and receive feedback on work done by themselves and others.

What skills and knowledge do learners bring with them from Year 9 to Key Stage 4?

Learners will cover all the key criteria from the Key Stage 3 National Curriculum, as identified previously. More specifically in Year 9:

- Understand the origin and uses of AI, how rules are used in AI decision making and how intelligence can be measured in humans and computers
- Ethics – understanding what ethics is and consider some simple ethical hypothetical problems and discuss them as they relate to AI
- Know what the Turing Test is and how it works
- Use training data to develop facts and rules to classify / categorise data to solve problems
- The impact of AI and automation on jobs, both the loss, change and creation of jobs
- Apply theory to practice in the completion of three projects – image detection program, chat bot, rating a text review
- Investigating and giving examples of databases used by organisations
- Design and create a flat-file or two-table relational database, using suitable field types and adding appropriate validations
- Design and create an input form with user friendly features to support the entry of data
- Design and run queries and reports using data from one or both tables in the database
- Develop a front-end menu for their application linking to the database input form and report
- Write and develop programs in a high-level programming language, using a wide range of programming constructs and taking into account the importance of the process, syntax, testing and debugging
- Design and create a simple quiz game
- Write and develop programs in a high-level programming language, using procedures and functions with parameters
- Concept of modular programming including its benefits and drawbacks
- Recognise and analyse code being able to describe its function
- Recognise and understand the power of problem solving and the different methods that Computer Scientists use to tackle problems – algorithmic thinking, abstraction, decomposition.
- Ask logical questions to solve problems
- Understand what an algorithm is and create sequences of instructions to achieve goals
- Know the basic logic gates: AND, OR, NOT
- Complete truth tables for logic gates and circuits with up to three inputs
- Write and develop programs in a high-level programming language to solve a problem, for a given audience and purpose
- Collaborative working in order to give and receive feedback on work done by themselves and others.

The modules at Key Stage 3 will enable learners to meet the following core Key Stage 4 GCSE course content:

- Fundamentals of algorithms
- Programming
- Fundamentals of data representation
- Systems architecture
- Computer networks and connections
- Cyber security
- Relational databases and SQL
- Ethical, legal and environmental impacts
- Languages and IDEs

What skills and knowledge do learners bring with them from Key Stage 4 Year 1 to Key Stage 4 Year 2?

Learners will cover the key criteria from the Key Stage 4 National Curriculum, as identified previously. Year 1 will focus on topics 1.1, 1.2, 1.3, 2.2, 2.4 and part of 2.1. More specifically:

- The purpose of a CPU, the fetch-execute cycle and what occurs at each stage
- The role and purpose of each component of the CPU: ALU, CU, cache, registers
- Von Neumann Architecture – the purpose of each register and what it stores: MAR, MDR, PC, ACC
- The difference between storing data and an address
- Common characteristics that effect the performance of CPUs: clock speed, cache size, number of cores
- The purpose and characteristics of embedded systems, including a range of examples
- The need for primary storage and the purpose of RAM and ROM
- The difference between RAM and ROM
- Why virtual memory may be needed and how it works
- The need for secondary storage and recognise a range of secondary storage devices and media
- Common types of storage: optical, magnetic, solid state
- Identify suitable storage devices and storage media for given applications
- Advantages and disadvantages of different storage devices and storage media relating to capacity, speed, portability, durability, reliability, cost, ; applying the knowledge to a given scenario
- Why data must be stored in binary format, converted to be processed by a computer

- Familiarity with data units and moving between each – Bit, Nibble, Byte, Kilobyte, Megabyte, Gigabyte, Terabyte, Petabyte
- Data capacity and calculation of data capacity requirements
- Convert positive denary whole numbers to binary numbers (up to and including 8 bits) and vice versa
- Addition – two binary integers (up to and including 8 bits) and explain overflow errors which may occur
- Convert positive denary whole numbers into 2-digit hexadecimal numbers and vice versa
- Convert binary integers to their hexadecimal equivalents and vice versa
- Binary shifts
- How characters are represented in binary
- 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 – ASCII, Unicode
- How images are represented as a series of pixels, represented in binary
- Metadata stores additional image information, e.g. height, width, colour depth, etc.
- The effect of colour depth and resolution on the quality of an image and the size of an image file
- How analogue sound must be stored in binary and therefore needs to be sampled to be stored in a digital form
- The effect of sample rate, duration and bit depth on the playback quality and the size of the sound file
- The need for compression and the types of compression: lossy and lossless
- Advantages and disadvantages of each type of compression
- The effects on the file for each type of compression
- The characteristics of LANs and WANS including common examples of each
- The different factors that can affect the performance of networks
- The hardware needed to connect to a network and the tasks performed by each piece of hardware
- The different roles of computers in a client-server and a peer-to-peer network
- The concept of the Internet as a network of computer networks
- The role of the DNS in the conversion of a URL to an IP address
- Concepts of servers providing services and clients requesting / using services from a server
- Cloud computing and its advantages and disadvantages
- Advantages and disadvantages of star and mesh topologies
- Wired and wireless networks and being able to compare the benefits and drawbacks of the modes of connection
- Principle of encryption to secure data across network connections
- IP addressing and the format of an IP address
- MAC addressing and MAC address being assigned to a device

- Standards allowing hardware / software to interact across different manufacturers and producers
- Principle of a protocol as a set of rules for transferring data
- Different protocols, their purpose and key features: TCP/IP, HTTP, HTTPS, FTP, POP, IMAP, SMTP
- Concept of layers, how they are used in protocols and the benefits of using layers
- Principles of computational thinking: abstraction, decomposition, algorithmic thinking
- Designing, creating and refining algorithms
- Identify inputs, processes and outputs for a problem
- Structure diagrams
- Create, interpret, correct, complete and refine algorithms using pseudocode, flowcharts, reference language / high-level programming language
- Identify common errors and suggest fixes
- Create and use trace tables to follow an algorithm
- Programming fundamentals – understanding and the practical application of each technique in a high-level programming language
- Use variables, constants, operators, inputs, outputs and assignments
- Use of the three basic programming constructs used to control the flow of a program: sequence, selection, iteration
- Common arithmetic operators: addition, subtraction, multiplication, division, modulus, quotient, exponentiation, comparison operators
- Common Boolean operators AND, OR and NOT
- Understand and the practical use of data types in a high-level programming language: integer, real, Boolean, character, string, casting
- Choose suitable data types for data in a given scenario
- Understand data types may be temporarily changed through casting, and where this might be useful
- Practical use of additional programming techniques in a high-level language: string manipulation, basic file handling operations, use of records to store data, use of SQL to search for data, use of arrays both one-dimensional and two-dimensional, sub programs (functions and procedures) to produce structured code, random number generation
- Simple logic diagrams using the operators AND, OR and NOT, recognising each gate symbol
- Truth tables for each logic gate
- Combining Boolean operators using AND, OR and NOT and applying logical operators in truth tables to solve problems

What skills and knowledge do learners bring with them from Key Stage 4 Year 2 to Key Stage 5 / beyond?

Learners will cover the key criteria from the Key Stage 4 National Curriculum, as identified previously. Year 2 will focus on topics 1.4, 1.5, 1.6, 2.3, 2.5 and the rest of 2.1. More specifically:

- Skills and knowledge from Year 1
- Threats to computer devices, systems and networks, understanding the principles of each form of attack, including why and how it is used: malware, social engineering, brute-force attacks, denial of service attacks, data interception and theft, the concept of SQL injection
- Identify and prevent vulnerabilities to the threats posed, what the method limits / prevents and how: penetration testing, anti-malware, firewalls, user access levels, passwords, encryption, physical security
- Purpose and functionality of operating systems including the features of an OS user interface, purpose of memory management and how it allows for multitasking, peripheral management and drivers, user management, file management
- Purpose and functionality of utility software and how it performs housekeeping tasks
- Purpose of utility software and why it is required: encryption software, defragmentation, data compression
- Recognise the impact of digital technology on society: ethical, legal cultural, environmental, privacy
- Knowledge and the ability to discuss of a variety of examples of digital technology and how these impacts on society – relating to the aforementioned issues
- The purpose of each piece of legislation and the specific actions it allows or prohibits: DPA, CMA, CDPA
- Features of open sources and proprietary software and benefits and drawbacks of each, being able to recommend a type of licence for a given scenario
- Principles of computational thinking: abstraction, decomposition, algorithmic thinking
- Understanding the main steps of each standard searching algorithm: binary, linear
- Understand the main steps of each standard sorting algorithm: bubble, merge, insertion
- Understand any pre-requisites of an algorithm and apply the algorithm to a data set
- Identify an algorithm if given the code for it
- Defensive design considerations, anticipating misuse and authentication
- Understanding of the issues a programmer should consider to ensure that program caters for all likely input values, including practical experience of input validation in a high-level programming language
- Deal with invalid data in a program
- Maintainability of programs, including the use of sub programs, naming conventions, indentations and commenting
- Purpose of testing and the types of testing: iterative, final / terminal
- Identify syntax and logic errors

- Selecting and using suitable test data: normal, boundary, invalid, erroneous
- Create / complete test plans
- Refining algorithms as a result of testing
- Characteristics and purpose of different levels of programming language: high-level, low-level
- Purpose / need for translators
- Characteristics of a compiler and an interpreter; differences, benefits and drawback of using a compiler or an interpreter
- Knowledge of the tools that an IDE provides, how each can be used to help develop a program, including practical experience of using a range of these tools within at least one IDE: editors, error diagnostics, run-time environment, translators

What skills and knowledge do learners bring with them from Year 12 to Year 13?

Year 12 will focus on topics 1.1, 1.2, 1.4, 2.1, 2.2 and parts of 1.3, 1.5, 2.3, more specifically:

- ALU, CU and Registers (PC, ACC, MAR, MDR, CIR), buses – data, address and control, and how it relates to assembly language programs (Little Man Computer)
- Fetch-Decode-Execute Cycle; including its effects on registers
- Factors affecting the performance of the CPU – clock speed, number of cores, cache
- Use of pipelining in a processor to improve efficiency
- Von Neumann, Harvard and contemporary processor architecture
- Differences between and uses of CISC and RISC processors
- GPUs and their uses (including those not related to graphics)
- Multicore and Parallel systems
- 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
- The need for, function and purpose of operating systems
- Memory management (paging, segmentation and virtual memory)
- Interrupts, the role of interrupts and Interrupt Service Routines (ISR), role within the FDE 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
- Nature if 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
- 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
- Need for and characteristics of a variety of programming paradigms
- Procedural languages
- Assembly language (including following and writing simple programs with the Little Man Computer
- Modes of addressing memory (immediate, direct, indirect and indexed)
- Object-oriented languages with and understanding of classes, objects, methods, attributes, inheritance, encapsulation and polymorphism
- Lossy vs Lossless compression
- Run-length encoding and dictionary coding for lossless compression
- Symmetric and asymmetric encryption
- Different uses of hashing
- Relational database, flat file, primary key, foreign key, secondary key, entity relationship modelling, normalisation and indexing
- Normalisation to 3NF
- SQL – interpret and modify
- HTML, CSS and JavaScript
- Search engine indexing
- PageRank algorithm
- Server and client-side processing
- Primitive data types, integer, real / floating point, character, string and Boolean
- Representing positive integers in binary
- The use of sign and magnitude and two’s complement to represent negative numbers in binary

- Addition and subtraction of binary integers
- Representing positive integers in hexadecimal
- Converting positive integers between binary, hexadecimal and denary
- Representation and normalisation of floating-point numbers in binary
- Floating point arithmetic, positive and negative numbers addition and subtraction
- Bitwise manipulation and masks: shifts, combining AND, OR, and XOR
- How character sets (ASCII and Unicode) are used to represent text
- Arrays (of up to 3 dimensions), records, lists, tuples
- The following structures to store data: linked-list, graph (directed and undirected), stack, queue, tree, binary search tree, hash table
- How to create, traverse, add data to and remove data from data structures mentioned (either using arrays and procedural programming or an object-oriented approach
- Computing related legislation: The Data Protection 1998, The Computer Misuse Act 1990, The Copyright, Design and Patents Act 1988, The Regulation of Investigatory Powers Act 2000
- 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
- The nature of and need for abstraction
- Differences between an abstraction and reality
- Devise and abstract model for a variety of situations
- Identify the inputs and outputs for a given situation
- Determine the preconditions for devising a solution to a problem
- Nature, benefits and drawbacks of caching
- Need for reusable program components
- Identify the components of a problem
- Identify the components of a solution to a problem
- Determine the order of the steps needed to solve a problem
- Identify the sub-procedures necessary to solve a problem
- 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
- Determine the parts of a problem that can be tackled at the same time

- Outline benefits and trade-offs that might result from concurrent processing in a particular situation
- Programming constructs: sequence, selection, iteration, branching
- Recursion, how it can be used and compares to an iterative approach
- Global and local variables
- Modularity, functions and procedures, parameter passing by value and by reference
- Use of IDE to develop / debug a program
- Use of object-oriented techniques
- Features that make a problem solvable by computational methods
- Problem recognition
- Problem decomposition
- Use of divide and conquer
- Use of abstraction
- Learners should apply their knowledge of: backtracking, data mining, heuristics, performance modelling, pipelining, visualisation to solve problems
- Analysis and design of algorithms for a given situation
- Suitability of different algorithms for a given task and data set, in terms of execution time and space
- Programming Project (NEA): analysis of the problem

What skills and knowledge do learners bring with them from Year 13 to further study / employment?

Year 13 will focus on the Programming Project (NEA) and rest of 1.3, 1.5 and 2.3 more specifically:

- Skills and knowledge from Year 12
- Relational database, flat file, primary key, foreign key, secondary key, entity relationship modelling, normalisation and indexing
- 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
- Characteristics of networks and the importance of protocols and standards
- The internet structure: TCP / IP Stack, DNS, protocol layering, LANs and WANs, packet and circuit switching
- Network security and threats, use of firewalls, proxies and encryption
- Network hardware

- Client-server and peer-to-peer
- Define problems using Boolean logic
- Manipulate Boolean expressions, including the use of Karnaugh maps to simplify Boolean expressions
- Use the following rules to derive or simplify statements in Boolean algebra: De Morgan's Laws, distribution, association, commutation, double negation
- Using logic gates and truth tables
- The logic associated with D type flip flops, half and full adders
- Computing related legislation: The Data Protection 1998, The Computer Misuse Act 1990, The Copyright, Design and Patents Act 1988, The Regulation of Investigatory Powers Act 2000
- 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
- Measures and methods to determine the efficiency of different algorithms, Big O notation (constant, linear, polynomial, exponential and logarithmic complexity)
- Comparison of the complexity of algorithms
- Algorithms for the main data structures, (stacks, queues, trees, linked lists, depth-first (post-order) and breadth-first traversal of trees)
- Standard algorithms (bubble sort, insertion sort, merge sort, quick sort, Dijkstra's shortest path algorithm, A* algorithm, binary search and linear search)
- Use and function of Python, SQL, HTML
- 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 solution
- Programming Project (NEA): analysis of the problem, design of the solution, developing the solution, testing and evaluating the solution

What will learners study, be expected to know and remember?

Learners will study the following modules / sub-topics:

	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
Year 7	Using computers safely, effectively and responsibly	Understanding computers	Control systems with Flowol	Games programming in Scratch	micro:bit	Graphics
Year 8	Computer crime and cyber security	Networks	HTML and website development	Introduction to Python	Creating a video	Sound manipulation in Audacity
Year 9	AI and machine learning	Database development	First steps in Small Basic	Python: Next Steps	Computational thinking and logic	Python project
KS4 Year 1	2.4.1 Boolean logic 1.2.3 Units 1.2.4 Data storage – numbers 2.1.2 Designing, creating and refining algorithms	2.1.2 Designing, creating and refining algorithms 2.2.1 Programming fundamentals 2.2.2 Data types Practical Programming Skills including 2.1.1 Computational thinking	2.2.3 Additional Programming Techniques Practical Programming Skills including 2.1.1 Computational thinking	1.2.4 Data Storage – Characters, Images, Sound, Compression 1.1.1 Architecture of the CPU 1.1.2 CPU performance	1.1.3 Embedded systems 1.2.1 Primary storage 1.2.2 Secondary storage 1.3.1 Networks and topologies	1.3.1 Networks and topologies 1.3.2 Wired and wireless networks, protocols and layers Practical Programming Skills including 2.1.1 Computational thinking
KS4 Year 2 & Year 3 2024 Cohort	1.4.1 Threats to computer systems and networks 1.4.2 Identifying and preventing vulnerabilities 1.5.1 Operating systems 1.5.2 Utility software 1.6.1 Ethical, legal, cultural and environmental impact	1.6.1 Ethical, legal, cultural and environmental impact 2.3.1 Defensive design 2.3.2 Testing 2.5.1 Languages 2.5.2 IDEs	2.1.3 Searching and sorting algorithms Practical Programming Skills including 2.1.1 Computational thinking and 2.1.3 Searching and sorting algorithms	Theory Revision Practical Programming Revision	Theory Revision Practical Programming Revision	x

	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
Year 12	1.4.1 a-h Data types 2.1.1 Thinking abstractly 2.1.2 Thinking ahead 2.1.3 Thinking procedurally 2.1.4 Thinking logically 2.1.5 Thinking concurrently 2.2.1 a-d Programming techniques	1.4.1 i-j Data types 1.4.2 a-c Data structures 2.2.1 e-f Programming techniques 2.2.2 Computational methods	1.1.1 Structure and function of the processor 1.1.2 Types of processor 1.1.3 Input, output and storage 1.3.1 Compression, encryption and hashing 1.2.1 Systems software 1.2.2 Applications generation 1.2.3 Software development	1.2.4 Types of programming language 1.3.2 a, c, d Databases	1.3.4 Web technologies 1.4.2 a-c Data structures	1.5.1 Computing related legislation 1.5.2 Moral and ethical issues 2.3.1 a-b Algorithms NEA Project 3.1 Analysis of the problem - Project Proposal
Year 13	1.4.3 Boolean Algebra 1.3.2 Databases 1.3.3 Networks NEA Project 3.1 Analysis of the problem - Project Proposal 3.2 Design of the solution - Project Design / Write-up	1.5.1 Computing related legislation 1.5.2 Moral and ethical issues 2.3.1 c-e Algorithms NEA Project 3.2 Design of the solution - Project Design / Write-up 3.3 Developing the solution - Project Coding and Testing / Write-up	2.3.1 f Algorithms NEA Project 3.3 Developing the solution - Project Coding and Testing / Write-up	NEA Project 3.4 Evaluation – Project Evaluation / Completion / Write-up	Theory Revision Exam Technique Remedial work on project – if needed	Theory Revision Exam Technique

The details of what learners are expected to know for each module / half term, follow on the next series of pages. How the modules covered in Years 7, 8 and 9 map against the National Curriculum Programme of Study for Key Stage 3 are represented in the table on the next page

KS3 National Curriculum Map	Year 7 - Unit 1: Using computers safely, effectively and responsibly	Year 7 - Unit 2: Understanding Computers	Year 7 - Unit 3: Control System with Flowol	Year 7 - Unit 4: Games Programming in Scratch	Year 7 - Unit 5: micro:bit	Year 7 - Unit 6: Graphics	Year 8 - Unit 1: Computer crime and cyber security	Year 8 - Unit 2: Networks	Year 8 - Unit 3: HTML and Website Development	Year 8 - Unit 4: Introduction to Python*	Year 8 - Unit 5: Creating a Video	Year 8 - Unit 6: Sound Manipulation in Audacity	Year 9 - Unit 1: AI and Machine Learning	Year 9 - Unit 2: Database Development	Year 9 - Unit 3: First Steps in Small Basic	Year 9 - Unit 4: Python Next Steps**	Year 9 - Unit 5: Computational Thinking and Logic	Year 9 - Unit 6: Python Project	
Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems			x	x	x								x			x	x	x	7
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										x					x	x	x	x	5
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			x	x	x				x	x			x	x	x	x	x	x	11
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]		x		x	x												x	x	5
Understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems	x	x	x	x				x	x					x					7
Understand how instructions are stored and executed within a computer system		x		x				x		x					x	x			6
Understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally, in the form of binary digits		x				x						x						x	4
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						x			x		x	x						x	5
Create, re-use, revise and re-purpose digital artefacts for a given audience, with attention to trustworthiness, designs and usability						x			x		x	x						x	5
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	x						x	x					x					x	5

	Year 7	Year 8	Year 9
Autumn 1	**Using computers safely, effectively and responsibly** A theoretical module covering the necessary basic knowledge to use computers safely, effectively and responsibly. At the end of the module all learners should be able to: • use basic file management techniques to create folders, save, copy, move, rename and delete files and folders and make backup copies • recognise extensions for common file types • keep their files in well organised and appropriately named folders • explain what constitutes a 'strong' password for an online account • describe a code of conduct • list some dangers and drawbacks of social networking sites • list some of the possible responses to cyberbullying • send and reply to emails, send attachments • use a search engine to find information Most learners will be able to: • describe guidelines for keeping their identity secure on the Internet • describe what is meant by identity theft and how to minimise the risks • identify a probable phishing email and deal with it appropriately • describe how to minimise the danger of having their computer infected by a virus. • resize images before attaching to emails • explain the advantages and disadvantages of email as a method of communication Some learners will be able to: • manage a Contacts list efficiently for email • use an email signature • use the advanced features of a search engine • describe why the information they find may not be accurate	**Computer crime and cyber security** A theoretical module covering some of the legal safeguards regarding computer use. Phishing scams and other email frauds, hacking, 'data harvesting' identity theft and safe use of social media are discussed together with ways of protecting online identity and privacy. Health and Safety law and environmental issues are also discussed. At the end of the module all learners should be able to: • name the major Acts concerning computer use • describe some of the dangers of putting personal data on social networking sites • describe ways of protecting online identity • identify some of signs of fraudulent emails and respond appropriately • adhere to Copyright Law • list some of the Health and Safety hazards associated with computer use • describe how to safely dispose of an old computer Most learners will be able to: • briefly describe the content of the major Acts • find out what data is held about them by companies • recognise fraudulent emails and protect themselves • protect their online identity using Privacy settings and by not uploading personal details • use computers sensibly and safely with regard to physical hazards Some learners will be able to: • respond effectively and appropriately to emails • describe the effects on individuals and companies of illegally downloading copyright material	**AI and machine learning** A theoretical and practical module covering the history and developments behind Artificial Intelligence (AI). It looks out how AI uses machine learning and learners consider the ethical issues surrounding the use of AI. Learners will then apply the theory to three projects – an image detection program, a chatbot and a program that rates text reviews. At the end of the module all learners should be able to: • understand the origin and uses of AI • understand how rules are used in decision making • understand what ethics is • understand how intelligence can be measured in humans and computers • know the Turing test and how it works Most learners will be able to: • investigate the rules needed to solve problems • understand the difference between facts and rules • describe uses of machine learning • use training data to create rules that solve problems of categorising data • understand what ethics is, discussing ethical issues as they relate to AI • understand how jobs can be affected by AI • understand challenges around facial recognition • understand how images are stored as binary data • describe a technique for detecting patterns in a grid • review program code, adapting it to detect given shapes • program a chatbot • understand the analysis of text to rate an attitude or opinion • review the program, identifying areas for improvement Some learners will be able to: • discuss strengths and weaknesses of machine learning • understand how bias can be introduced • describe the opportunities and problems of using AI for sentiment analysis • understand why interpreting patterns is not as useful a skill as 'thinking'

	Year 7	Year 8	Year 9
Autumn 2	**Understanding computers** A theoretical module covering the basic principles of computer architecture and use of binary, including the history of communication devices and new and emerging technologies. At the end of the module all learners should be able to: • distinguish between hardware and software, giving examples • draw a block diagram of a basic computer system – input, process, output and storage • name different types of permanent storage device • suggest appropriate input and output devices for a simple scenario • explain what RAM and ROM are used for • show how numbers and text can be represented in binary • explain the impact of future technologies Most learners will be able to: • perform simple binary arithmetic • state the strengths and weaknesses of different storage devices • describe briefly how data is stored on a CD Some learners will be able to: • identify input and output devices for more complex scenarios • explain how characters are encoded using the ASCII system • use an ASCII reference chart to convert a character into binary and its denary equivalent	**Networks** A theoretical module covering the basic principles and architecture of networks and how data is communicated over networks. At the end of the module all learners should be able to: • state that the Internet is a wide area network and the World Wide Web is part of the Internet • define the meaning of the terms 'domain name', http, protocol • explain the basic principle of packet switching • give examples of LANs and WANs • state three different network topologies • describe what is meant by a client-server network and state some of its advantages • state why some transmissions are encrypted and use a simple algorithm to encrypt and decrypt a message Most learners will be able to: • explain the meaning and significance of bandwidth • explain what is meant by buffering and why it is used • state the advantages and disadvantages of different network topologies • design a simple network layout • identify some of the extra hardware components used in a LAN • compare the uses of peer-to-peer networks and client-server networks Some learners will be able to: • design a network layout for the school, using icons to represent the server, hub, switch, router, Internet, workstation, printer • describe the concept of cloud computing and some of the benefits it brings to individuals and organisations	**Database development** A practical module covering the basic theory, creation and use of a single-table database and a simple relational database involving two tables in a one-to-many relationship. Learners will start by looking at an existing single-table database, learning how to add records and make queries. The module will then move onto more complex skills making use of a database of their own creation. At the end of the module all learners should be able to: • give examples of databases used by organisations which are accessible to the public via the Internet • create a database table using several fields with different data types • state the purpose of a primary key in a database • create a basic input form to input data • query the database using more than one criterion to find answers to user queries • create a basic report with suitable headings • create a front-end application menu with buttons linking to a form and a report Most learners will be able to: • add features to an input form to make it more user-friendly • fully customise their input forms and reports Some learners will be able to: • create the relationship between two linked tables • create a complex query which uses two tables in a relational database • create a report which uses data from linked tables • edit a report structure and add subtotals and / or a total to the report

	Year 7	Year 8	Year 9
Spring 1	**Control systems with Flowol** A practical module covering the principles of producing control and monitoring solutions using a flowchart based interface. At the end of the module all learners should be able to: • identify everyday situation where computer control is used • identify common types of sensors used by control systems • identify control flowchart symbols and understand how they are used to break down problems • produce flowchart-based solutions for control systems that include sequences and loops Most learners will be able to: • explain why control systems might fail and how this might impact on safety • produce control solutions for problems that include subroutines Some learners will be able to: • produce control solutions for problems that include variables	**HTML and website development** A practical module developing the basics of HTML and CSS in order to create a working multi-page website using a text editor, which meets the needs of a specific purpose and audience. At the end of the module all learners should be able to: • write HTML code to create a simple webs page and display it in a browser • write CSS to define styles used in a web page • create a simple navigation system using HTML • use a design to create a template for a web page using HTML • create their own multi-page website • insert text, images and links on their web pages Most learners will be able to: • use a range of HTML tags to create well laid out web pages • write CSS code to define the styles of different parts of a web page • use HTML and CSS to create their web page template • use the template to design a multi-page website with a consistent look and feel to each page • use responsive design techniques in creating their website so that the web pages will adapt to any size of screen • create a simple web form to collect user data Some learners will be able to: • add enhancements or additional features to the original basic design • construct a good-looking, well-formatted interactive website that is suitable for its intended audience	**First steps in Small Basic** A practical module introducing a textual language called Small Basic, designing to make programming easy and approachable. It introduces Turtle graphics, leading to the use of variables and loops and inputs, outputs and selection. It improves the use and familiarity of these programming statements whilst producing a quiz game, coloured graphics and a simple screensaver. At the end of the module all learners should be able to: • write and run programs in Small Basic using For… EndFor loops, variables, input, output and selection statements • create a simple quiz game • identify and correct syntax errors in a program Most learners will be able to: • use a While… EndWhile loop in a program • find and correct logic errors in a program • use the graphics window to draw different shapes in random colours Some learners will be able to: • use variables effectively to create repeating patterns • add scoring to their quiz game • create an effective screensaver which runs until the user stops it

	Year 7	Year 8	Year 9
Spring 2	**Games programming in Scratch** A practical module involving the planning, development, testing and refinement of a computer game using block programming, incorporating variables, procedures, lists and operators. At the end of the module all learners should be able to: • relate computational abstractions and simple programming code to on-screen actions • design simple algorithms to solve problems • sequence instructions in order to make things happen • use variables in programming structures • assemble code in procedural blocks • use simple Boolean operators in programming code • identify and use screen objects in their own Scratch game • carry out simple tests to debug their project Most learners will be able to: • write their own instructions to create and use a simple list • use the broadcast function in Scratch at a simple level • make good use of operators • incorporate a range of sprites which can be controlled in different ways • improve the game / project based on peer feedback • systematically test the game / project to ensure that few errors remain Some learners will be able to: • use the broadcast function in Scratch effectively • use a range of 'event handlers' effectively to create a complex game / project • effectively design, implement and refine their own algorithms • compare the effectiveness of their algorithms with those of peers • critically analyse the limitations of their projects	**Introduction to Python** A practical module introducing a high-level, but easy-to-use, programming language, in order to understand the process of developing programs, the importance of writing the correct syntax, being able to formulate algorithms for simple programs and debugging the programs. At the end of the module all learners should be able to: • run simple Python programs in Interactive and Script mode • write pseudocode to outline the steps in an algorithm prior to coding • write programs using different types of data • correctly used different variable types, assignment statements, arithmetic operators • distinguish between syntax and logic errors, finding and correcting both types of error • describe the purpose of pseudocode in designing algorithms • use comments to document their programs and explain how they work • write an error-free, well-documented program involving sequence, selection and iteration, with some help given Most learners will be able to: • write an error-free, well-documented program involving selection and iteration • describe how a binary search is carried out • explain the advantages of a binary search over a linear search for an ordered list Some learners will be able to: • devise their own algorithms to solve reasonable complex problems • test and debug their programs, and correct both syntax and logic errors • make allowances in their programs for user input errors, ensuring that the program still runs to a successful conclusion	**Python: Next Steps** A practical module developing skills and experience in a high-level, but easy-to-use, programming language further in preparation for GCSE. Covering a range of more advanced constructs – iterations, arrays, procedures, functions, and understanding the concept and benefits of modular programming. At the end of the module all learners should be able to: • use data types correctly and convert between them when necessary • write programs that use a loop to repeat a section of code • write programs that use lists • create and use a function with or without parameters • find and debug syntax errors • look at a given section of code and describe its function Most learners will be able to: • select the most suitable type of loop for a given problem • use counters correctly in conjunction with for loops • create a list and append or change elements of the list • explain the advantage of functions for reusable sections of program code Some learners will be able to: • use loops to populate, interrogate and print lists, using a counter as an index to an array element • devise their own functions to create a modular program • create a program that is easy to use, caters for user input errors, has explicit error messages telling the user what the correct form of entry is and produces output with suitable headings or explanation

	Year 7	Year 8	Year 9
Summer 1	**micro:bit** A practical module to introduce programming using the BBC micro:bit. Learners understand the purpose of the device and what it can do and will learn to program it, initially using block programming and moving to programming it in a high-level textual based language. The use of the micro:bit encourages learners to learn in an exploratory fashion, extending their knowledge through experimentation. At the end of the module all learners should be able to: • create a basic image on the micro:bit • create a 'Guess the Number' game • annotate the code • create a safe lock / 3 number safe lock system • state the purpose of the safe lock system • use feedback to correct any errors Most learners will be able to: • create multiple images on the micro:bit, using the buttons to select different images • create a number guessing game which engages the user with suspense • write up the code • make use of a development log to annotate their code, to explain what is going on • effectively explain how the safe lock system works • use feedback to improve their work • write a 3 number safe lock system in both Blocks Editor and a scripting editor • compare the differences between a Blocks Editor and a scripting editor Some learners will be able to: • effectively explain and discuss examples of different Blocks Editor code and explain what it is doing • using IF statements • make use of a development log to effectively annotate their code, to explain what is going on • demonstrate effective annotation showing the improvements made from feedback • clearly understand the limitations of the Blocks Editor • extend the safe lock system using the scripting editor, taking advantage of the inputs available	**Creating a video** A practical module where learners will undertake a creative project to analyse, plan, shoot and edit a short advertisement for TV, a short movie on a topic such as Cyber Crime or a short film trailer. At the end of the module all learners should be able to: • work as part of a team to complete an appropriate advertisement or movie • work collaboratively on editing and giving feedback on the work of others • show discrimination in selecting accompanying material such as still images, sound effects and background music • use a range of digital devices • use video transitions and effects to improve their movie Most learners will be able to: • select appropriate material for a specific audience • combine music and sound effects with moving and still images from different packages and sources into one end product • add introductory and final pages with appropriate text Some learners will be able to: • plan and share the elements of a team project taking into account the strengths of each team member	**Computational thinking and logic** A theoretical module introducing learners to the world of computational thinking and logic, allowing them to understand the power of problem solving and the different methods that Computer Scientists use to tackle problems – algorithmic thinking, abstraction and decomposition. At the end of the module all learners should be able to: • ask logical questions to solve problems • know the common Boolean operators: AND, OR, NOT • know the different logic gates: AND, OR, NOT • understand what an algorithm is • create a sequence of instructions to achieve a goal Most learners will be able to: • understand how Boolean operators can be represented in written expressions & Venn diagrams • understand how logic is used in different situations • complete truth tables for logic gates and circuits with up to three inputs • understand how loops can be used to reduce the amount of code required for a solution • refine algorithms to reduce the number of instructions required • understand the difference between lossy and lossless compression, and why it is needed for video transmission and photo storage • use an algorithm to communicate data • understand how abstractions are used in real life • create abstractions for different purposes • understand how networks are used to make an abstraction of a maze • understand how decomposition can be used to solve problems • break down a large computing problem into its parts Some learners will be able to: • understand how nested loops can be used to improve solutions further • understand network theory terms (nodes, edges) • understand packet switching

	Year 7	Year 8	Year 9
Summer 2	**Graphics** A practical module about how images are represented and stored on computers, undertaking a creative project to analyse, plan, design, photo edit and manipulate images to produce a film poster for a specific audience. At the end of the module all learners should be able to: • explain that bitmap images are made up of individual pixels • explain that in the case of vector graphics, properties such as position, fill, stroke colour and dimensions are stored • create and manipulate a simple group of objects to form a logo design • change the saturation, brightness and contrast in an image • add text to a graphic • use a graphics package to create an artwork Most learners will be able to: • describe the characteristics of bitmap and vector graphics, state the advantages of each, giving examples of situations where they would be appropriate to use • use fonts consistently and carefully to convey a particular message or image • use white space effectively use layers in the creation of an artwork Some learners will be able to: • use the advanced facilities of a graphics package • create a series of two or more posters in the same style, using a combination of layered images and fonts effectively to convey a message	**Sound manipulation in Audacity** A practical module about how sound is digitised and stored on computers, undertaking a creative project to analyse, plan, record and edit a short sound file for a specific audience and purpose. At the end of the module all learners should be able to: • explain how sound is digitised • use input and output devices to record and play sounds • select suitable materials for a project • use basic editing techniques to produce a sound file • work collaboratively to give and receive feedback on work done by themselves and others Most learners will be able to: • select appropriate material for a specific audience • combine speech, music and sound effects from different sources into one end product • use more sophisticated editing techniques • explain how their product meets the given brief Some learners will be able to: • plan and create a project with the minimum of assistance • include a range of suitable techniques and effects to produce an effective product that meets the brief / specification	**Python project** A practical module that brings together the skills, knowledge and understanding across Key Stage 3 into a programming project. Preparing learners for study at KS4. The learners will select one of five challenges and will decompose and design a solution for the problem, program the solution – making use of annotations to explain the program, test the program and evaluate the solution, identifying areas for improvement. The challenges are: • Pythagoras Solver • Rock, Paper, Scissors • Login Server • Battle of the Bands • Password Reminder The projects will cover: • Using variables, operators, inputs, outputs and assignments • Using sequences, selection and iteration • Using count-controlled loops (for) and condition-controlled loops (while) • Using different types of data, i.e. integer, string, float and Boolean • Basic string manipulation • Basic file handling operations • Using lists • Using subroutines Some learners will be able to achieve the extension and further extension tasks identified for project they have selected.

	KS4 Year 1	KS4 Year 2
Autumn 1	**At the end of this half term, learners will know / have done:** **2.4.1 Boolean logic** • Simple logic diagrams using the operators AND, OR and NOT, recognising each gate symbol • Truth tables for each logic gate • Combining Boolean operators using AND, OR and NOT and applying logical operators in truth tables to solve problems **1.2.3 Units** • Why data must be stored in binary format, converted to be processed by a computer • Familiarity with data units and moving between each – Bit, Nibble, Byte, Kilobyte, Megabyte, Gigabyte, Terabyte, Petabyte • Data capacity and calculation of data capacity requirements **1.2.4 Data storage – numbers** • Convert positive denary whole numbers to binary numbers (up to and including 8 bits) and vice versa • Addition – two binary integers (up to and including 8 bits) and explain overflow errors which may occur • Convert positive denary whole numbers into 2-digit hexadecimal numbers and vice versa • Convert binary integers to their hexadecimal equivalents and vice versa • Binary shifts **2.1.2 Designing, creating and refining algorithms** • Identify inputs, processes and outputs for a problem • Structure diagrams • Create, interpret, correct, complete and refine algorithms using pseudocode, flowcharts, reference language / high-level programming language • Identify common errors and suggest fixes • Create and use trace tables to follow an algorithm	**At the end of this half term, learners will know / have done:** **1.4.1 Threats to computer systems and networks** • Threats to computer devices, systems and networks, understanding the principles of each form of attack, including why and how it is used: malware, social engineering, brute-force attacks, denial of service attacks, data interception and theft, the concept of SQL injection **1.4.2 Identifying and preventing vulnerabilities** • Identify and prevent vulnerabilities to the threats posed, what the method limits / prevents and how: penetration testing, anti-malware, firewalls, user access levels, passwords, encryption, physical security **1.5.1 Operating systems** • Purpose and functionality of operating systems including the features of an OS user interface, purpose of memory management and how it allows for multitasking, peripheral management and drivers, user management, file management **1.5.2 Utility software** • Purpose and functionality of utility software and how it performs housekeeping tasks • Purpose of utility software and why it is required: encryption software, defragmentation, data compression **1.6.1 Ethical, legal, cultural and environmental impact** • Recognise the impact of digital technology on society: ethical, legal cultural, environmental, privacy • Knowledge and the ability to discuss of a variety of examples of digital technology and how this impacts on society – relating to the aforementioned issues • The purpose of each piece of legislation and the specific actions it allows or prohibits: DPA, CMA, CDPA

	KS4 Year 1	KS4 Year 2
Autumn 2	**At the end of this half term, learners will know / have done:** **2.1.2 Designing, creating and refining algorithms** • Identify inputs, processes and outputs for a problem • Structure diagrams • Create, interpret, correct, complete and refine algorithms using pseudocode, flowcharts, reference language / high-level programming language • Identify common errors and suggest fixes • Create and use trace tables to follow an algorithm **2.2.1 Programming fundamentals** • Understanding and the practical application of each technique in a high-level programming language • Use variables, constants, operators, inputs, outputs and assignments • Use of the three basic programming constructs used to control the flow of a program: sequence, selection, iteration • Common arithmetic operators: addition, subtraction, multiplication, division, modulus, quotient, exponentiation, comparison operators • Common Boolean operators AND, OR and NOT **2.2.2 Data types** • Understand and the practical use of data types in a high-level programming language: integer, real, Boolean, character, string, casting • Choose suitable data types for data in a given scenario • Understand data types may be temporarily changed through casting, and where this might be useful **2.1.1 Computational thinking** • Principles of computational thinking: abstraction, decomposition, algorithmic thinking **Practical Programming Skills**	**At the end of this half term, learners will know / have done:** **1.6.1 Ethical, legal, cultural and environmental impact** • Features of open sources and proprietary software and benefits and drawbacks of each, being able to recommend a type of licence for a given scenario **2.3.1 Defensive design** • Defensive design considerations, anticipating misuse and authentication • Understanding of the issues a programmer should consider to ensure that program caters for all likely input values, including practical experience of input validation in a high-level programming language • Deal with invalid data in a program • Maintainability of programs, including the use of sub programs, naming conventions, indentations and commenting **2.3.2 Testing** • Purpose of testing and the types of testing: iterative, final / terminal • Identify syntax and logic errors • Selecting and using suitable test data: normal, boundary, invalid, erroneous • Create / complete test plans • Refining algorithms as a result of testing **2.5.1 Languages & 2.5.2 IDEs** • Characteristics and purpose of different levels of programming language: high-level, low-level • Purpose / need for translators • Characteristics of a compiler and an interpreter; differences, benefits and drawback of using a compiler or an interpreter • Knowledge of the tools that an IDE provides, how each can be used to help develop a program, including practical experience of using a range of these tools within at least one IDE: editors, error diagnostics, run-time environment, translators

	KS4 Year 1	KS4 Year 2
Spring 1	**At the end of this half term, learners will know / have done:** **2.2.3 Additional programming techniques** • Practical use of additional programming techniques in a high-level language: string manipulation, basic file handling operations, use of records to store data, use of SQL to search for data, use of arrays both one-dimensional and two-dimensional, sub programs (functions and procedures) to produce structured code, random number generation **2.1.1 Computational thinking** • Principles of computational thinking: abstraction, decomposition, algorithmic thinking **Practical Programming Skills**	**At the end of this half term, learners know / have done:** **2.1.1 Computational thinking** • Principles of computational thinking: abstraction, decomposition, algorithmic thinking **2.1.3 Searching and sorting algorithms** • Understanding the main steps of each standard searching algorithm: binary, linear • Understand the main steps of each standard sorting algorithm: bubble, merge, insertion • Understand any pre-requisites of an algorithm and apply the algorithm to a data set • Identify an algorithm if given the code for it **Practical Programming Skills**

	KS4 Year 1	KS4 Year 2
Spring 2	**At the end of this half term, learners know / have done:** **1.2.4 Data storage – characters, images, sound, compression** • How characters are represented in binary • 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 – ASCII, Unicode • How images are represented as a series of pixels, represented in binary • Metadata stores additional image information, e.g. height, width, colour depth, etc. • The effect of colour depth and resolution on the quality of an image and the size of an image file • How analogue sound must be stored in binary and therefore needs to be sampled to be stored in a digital form • The effect of sample rate, duration and bit depth on the playback quality and the size of the sound file • The need for compression and the types of compression: lossy and lossless • Advantages and disadvantages of each type of compression • The effects on the file for each type of compression **1.1.1 Architecture of the CPU** • The purpose of a CPU, the fetch-execute cycle and what occurs at each stage • The role and purpose of each component of the CPU: ALU, CU, cache, registers • Von Neumann Architecture – the purpose of each register and what it stores: MAR, MDR, PC, ACC • The difference between storing data and an address **1.1.2 CPU performance** • Common characteristics that effect the performance of CPUs: clock speed, cache size, number of cores	**At the end of this half term, learners know / have done:** Theory and Practical Programming Revision

	KS4 Year 1	KS4 Year 2
Summer 1	**At the end of this half term, learners will know:** **1.1.3 Embedded systems** • The purpose and characteristics of embedded systems, including a range of examples **1.2.1 Primary storage** • The need for primary storage and the purpose of RAM and ROM • The difference between RAM and ROM • Why virtual memory may be needed and how it works **1.2.2 Secondary storage** • The need for secondary storage and recognise a range of secondary storage devices and media • Common types of storage: optical, magnetic, solid state • Identify suitable storage devices and storage media for given applications • Advantages and disadvantages of different storage devices and storage media relating to capacity, speed, portability, durability, reliability, cost, ; applying the knowledge to a given scenario **1.3.1 Networks and topologies** • The characteristics of LANs and WANS including common examples of each • The different factors that can affect the performance of networks • The hardware needed to connect to a network and the tasks performed by each piece of hardware • The different roles of computers in a client-server and a peer-to-peer network • The concept of the Internet as a network of computer networks • The role of the DNS in the conversion of a URL to an IP address • Concepts of servers providing services and clients requesting / using services from a server • Cloud computing and its advantages and disadvantages • Advantages and disadvantages of star and mesh topologies	**At the end of this half term, learners know / have done:** Theory and Practical Programming Revision

	KS4 Year 1	KS4 Year 2
Summer 2	**At the end of this half term, learners will know:** **1.3.1 Networks and topologies** • The characteristics of LANs and WANS including common examples of each • The different factors that can affect the performance of networks • The hardware needed to connect to a network and the tasks performed by each piece of hardware • The different roles of computers in a client-server and a peer-to-peer network • The concept of the Internet as a network of computer networks • The role of the DNS in the conversion of a URL to an IP address • Concepts of servers providing services and clients requesting / using services from a server • Cloud computing and its advantages and disadvantages • Advantages and disadvantages of star and mesh topologies **1.3.2 Wired and wireless networks, protocols and layers** • Wired and wireless networks and being able to compare the benefits and drawbacks of the modes of connection • Principle of encryption to secure data across network connections • IP addressing and the format of an IP address • MAC addressing and MAC address being assigned to a device • Standards allowing hardware / software to interact across different manufacturers and producers • Principle of a protocol as a set of rules for transferring data • Different protocols, their purpose and key features: TCP/IP, HTTP, HTTPS, FTP, POP, IMAP, SMTP • Concept of layers, how they are used in protocols and the benefits of using layers **2.1.1 Computational thinking** • Principles of computational thinking: abstraction, decomposition, algorithmic thinking **Practical Programming Skills**	

<table>
<tr><th></th><th>Year 12</th><th>Year 13</th></tr>
<tr><td>All Terms</td><td colspan="2">Private Study to Promote Independence at KS5:
Read the BBC Technology page weekly
Work through the PMT revision material
Use Wingspan to complete the data structures and algorithms activities
Watch the Craig and Dave video series
Practise Python programming (at least 3 hours weekly)
Udemy courses – Python, OOP, Java Programming
MOOC – Databases, HTML, CSS, JavaScript, Algorithms and Graph Theory</td></tr>
<tr><td>Autumn 1</td><td>At the end of this half term, learners will know / have done:
1.4.1 Data types
• Primitive data types, integer, real / floating point, character, string and Boolean
• Representing positive integers in binary
• The use of sign and magnitude and two’s complement to represent negative numbers in binary
• Addition and subtraction of binary integers
• Representing positive integers in hexadecimal
• Converting positive integers between binary, hexadecimal and denary
• Representation and normalisation of floating-point numbers in binary
• Floating point arithmetic, positive and negative numbers addition and subtraction

2.1.1 Thinking abstractly
• The nature of and need for abstraction
• Differences between an abstraction and reality
• Devise and abstract model for a variety of situations

2.1.2 Thinking ahead
• Identify the inputs and outputs for a given situation
• Determine the preconditions for devising a solution to a problem
• Nature, benefits and drawbacks of caching
• Need for reusable program components</td><td>At the end of this half term, learners will know / have done:
1.3.2 Databases
• Relational database, flat file, primary key, foreign key, secondary key, entity relationship modelling, normalisation and indexing
• 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
• The internet structure: TCP / IP Stack, DNS, protocol layering, LANs and WANs, packet and circuit switching
• Network security and threats, use of firewalls, proxies and encryption
• Network hardware
• Client-server and peer-to-peer</td></tr>
</table>

	Year 12	Year 13
Autumn 1 Cont.	**2.1.3 Thinking procedurally** • Identify the components of a problem • Identify the components of a solution to a problem • Determine the order of the steps needed to solve a problem • Identify the 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 • Determine the logical conditions that affect the outcome of a decision • 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 • Outline benefits and trade-offs that might result from concurrent processing in a particular situation **2.2.1 Programming techniques** • Programming constructs: sequence, selection, iteration, branching • Recursion, how it can be used and compares to an iterative approach • Global and local variables • Modularity, functions and procedures, parameter passing by value and by reference	**1.4.3 Boolean algebra** • Define problems using Boolean logic • Manipulate Boolean expressions, including the use of Karnaugh maps to simplify Boolean expressions • Use the following rules to derive or simplify statements in Boolean algebra: De Morgan's Laws, distribution, association, commutation, double negation • Using logic gates and truth tables • The logic associated with D type flip flops, half and full adders **3.1 Analysis of the problem** • Problem identification • Stakeholders • Research the problem • Specify the proposed solution **3.2 Design of the solution** • Decompose the problem • Describe the solution • Describe the approach to testing Programming Project (NEA)

	Year 12	Year 13
Autumn 2	**At the end of this half term, learners will know / have done:** **1.4.1 Data types** • Bitwise manipulation and masks: shifts, combining AND, OR, and XOR • How character sets (ASCII and Unicode) are used to represent text **1.4.2 Data structures** • Arrays (of up to 3 dimensions), records, lists, tuples • The following structures to store data: linked-list, graph (directed and undirected), stack, queue, tree, binary search tree, hash table • How to create, traverse, add data to and remove data from data structures mentioned (either using arrays and procedural programming or an object-oriented approach **2.2.1 Programming techniques** • Use of IDE to develop / debug a program • Use of object-oriented techniques **2.2.2 Computational methods** • Features that make a problem solvable by computational methods • Problem recognition • Problem decomposition • Use of divide and conquer • Use of abstraction • Learners should apply their knowledge of: backtracking, data mining, heuristics, performance modelling, pipelining, visualisation to solve problems	**At the end of this half term, learners will know / have done:** **1.5.1 Computing related legislation** • The Data Protection 1998 • The Computer Misuse Act 1990 • The Copyright, Design and Patents Act 1988 • The Regulation of Investigatory Powers Act 2000 **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 **2.3.1 Algorithms** • Measures and methods to determine the efficiency of different algorithms, Big O notation (constant, linear, polynomial, exponential and logarithmic complexity) • Comparison of the complexity of algorithms • Algorithms for the main data structures, (stacks, queues, trees, linked lists, depth-first (post-order) and breadth-first traversal of trees) **3.2 Design of the solution** • As previous half term **3.3 Developing the solution** • Iterative development process • Testing to inform development Programming Project (NEA)

	Year 12	Year 13
Spring 1	**At the end of this half term, learners will know / have done:** **1.1.1 Structure and function of the processor** • ALU, CU and Registers (PC, ACC, MAR, MDR, CIR), buses – data, address and control, and how it relates to assembly language programs (Little Man Computer) • Fetch-Decode-Execute Cycle; including its effects on registers • Factors affecting the performance of the CPU – clock speed, number of cores, cache • Use of pipelining in a processor to improve efficiency • Von Neumann, Harvard and contemporary processor architecture **1.1.2 Types of processor** • 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 **1.3.1 Compression, encryption and hashing** • Lossy vs Lossless compression • Run-length encoding and dictionary coding for lossless compression • Symmetric and asymmetric encryption • Different uses of hashing	**At the end of this half term, learners will know / have done:** **2.3.1 Algorithms** • Standard algorithms (bubble sort, insertion sort, merge sort, quick sort, Dijkstra's shortest path algorithm, A* algorithm, binary search and linear search) **3.3 Developing the solution** • Iterative development process • Testing to inform development Theory Revision Exam Technique Programming Project (NEA)

	Year 12	Year 13
Spring 1 Cont.	**1.2.1 Systems software** • The need for, function and purpose of operating systems • Memory management (paging, segmentation and virtual memory) • Interrupts, the role of interrupts and ISR, role within the FDE Cycle • Scheduling: round robin, first come first served, multi-level feedback queues, shortest job first and shortest remaining time **1.2.2 Applications generation** • Nature if 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** • 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	

	Year 12	Year 13
Spring 2	**At the end of this half term, learners will know / have done:** **1.2.4 Types of programming language** • Need for and characteristics of a variety of programming paradigms • Procedural languages • Assembly language (including following and writing simple programs with the Little Man Computer • Modes of addressing memory (immediate, direct, indirect and indexed) • Object-oriented languages with and understanding of classes, objects, methods, attributes, inheritance, encapsulation and polymorphism **1.3.2 Databases** • Relational database, flat file, primary key, foreign key, secondary key, entity relationship modelling, normalisation and indexing • Normalisation to 3NF • SQL – interpret and modify	**At the end of this half term, learners will know / have done:** **3.4 Evaluation** • Testing to inform evaluation • Success of the solution • Describe the final product • Maintenance and development Theory Revision Exam Technique Programming Project (NEA) Submission
Summer 1	**At the end of this half term, learners will know / have done:** **1.3.4 Web technologies** • HTML, CSS and JavaScript • Search engine indexing • PageRank algorithm • Server and client-side processing **1.4.2 Data structures** • Arrays (of up to 3 dimensions), records, lists, tuples • The following structures to store data: linked-list, graph (directed and undirected), stack, queue, tree, binary search tree, hash table • How to create, traverse, add data to and remove data from data structures mentioned (either using arrays and procedural programming or an object-oriented approach	**At the end of this half term, learners know / have done:** Theory Revision Exam Technique Past Papers / Questions

	Year 12	Year 13
Summer 2	**At the end of this half term, learners will know / have done:** **1.5.1 Computing related legislation** • The Data Protection 1998 • The Computer Misuse Act 1990 • The Copyright, Design and Patents Act 1988 • The Regulation of Investigatory Powers Act 2000 **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 **2.3.1 Algorithms** • Analysis and design of algorithms for a given situation • Suitability of different algorithms for a given task and data set, in terms of execution time and space **3.1 Analysis of the problem** • Problem identification • Stakeholders • Research the problem • Specify the proposed solution Programming Project (NEA) – proposal	

Why has learning been sequenced in this way?

Key knowledge and understanding have been placed early on in Year 7, as they provide a grounding / foundation skills and knowledge required for many of the other modules being covered. It is important to check prior knowledge and address any gaps so that there is that strong foundation for the new skills, knowledge and learning. Topics are presented using small steps, providing models to allow learners to develop their knowledge and providing scaffolds for more challenging tasks, especially at Key Stage 4 and 5 where structured responses are a requirement. This allows learners to move the learning into their long-term memory making it easier to retrieve and apply in the future.

Across the Computing modules at Key Stage 3, each of the above National Curriculum criteria is covered at least four over the three years; therefore, skills and knowledge are revisited throughout the Key Stage and provide learners with opportunities to apply prior learning to new contexts. The modules cover a range of theoretical and practical aspects which are balanced across each year, so the focus is not to heavily-weighted to one or the other. The modules also reflect an increase in challenge across each year, for example moving from block based to textual based programming.

At Key Stage 4, the topics / sub-topics have been sequenced according to the scheme of work published by the exam board. It starts off with foundation skills and knowledge that will support the learning and progress in other topics and sub-topics and then the practical programming skills, gradually building up to the more complex topics and programming skills and knowledge. The interleaving provides learners the opportunity to continually use their skills and knowledge throughout the activities and practical programming scenarios – allowing learners to build confidence in the application to a range of real-world examples and examination style responses.

At Key Stage 5, Computing education equips pupils to understand and contribute to the ever-evolving world through logical thinking and creativity. The core of computing is computer science, in which pupils are taught the principles of information and computation, and how digital systems work. Computing equips pupils to use information technology to create programs, systems and a range of media. It also ensures that pupils become digitally literate. This includes the ability to use, express themselves and develop their ideas through information and communication technology. We are conscious of our responsibility to prepare pupils for the future workspace including jobs that may not yet exist. We do not merely teach pupils to use specific software packages; rather we teach fundamental principles and relate these to the evolving needs of the world. As pupils increasingly live their lives seamlessly on and offline, we teach them about the challenges, dangers and exciting opportunities offered by the online world.

Each module taught throughout KS5 has clearly outlined ‘key knowledge’ and ‘practical application’ skills. These are shared with pupils at the start of each module and are reviewed by the pupils again at the end of the module so they are able to self-assess. Our curriculum is coherently planned and sequenced such that work in each new module builds upon what has been taught previously. Modules are strategically interspersed to aid long-term recall based upon research theories such as spaced learning.

The KS5 Computer Science curriculum is exclusively taught by two subject specialists, both of whom have over a two decades of experience in successfully teaching Computing and ICT. The Computer Science curriculum is taught via one-hour lessons, nine times a fortnight. Teachers use marking, assessment

and class discussions to identify and correct misunderstandings and ensure that all pupils embed key concepts in their long-term memory. Pupils are provided with numerous opportunities to connect new knowledge and skills and build on prior learning.

Year 12 builds on the knowledge gained during KS4 with a review of topics which underpin the whole of the A-level. Once a baseline has been established the delivery is holistic where topics from both papers are taught together. This allows the pupils to see the connection between topics and allows for better understanding. Programming and data structures along with Computational think are taught during the first term and a half. This decision is based on these areas are the foundation of the subject and pupils must fully get to grips with these topics before the more advanced programming elements are introduced. The 2nd term focus on System Architecture, Software Development and Databases. The final term comes back to Data Structures as well as Encryption, Legal and Web Technologies Algorithms and the NEA.

Year 13 we revisit many of the topics taught during year 12 but now focus on the A-level material rather than the AS. Algorithms takes most of the first two terms along with Networking, Boolean Algebra as well as the NEA. Pupil must master the ability to program all algorithms and this takes time. We embed numerous programming challenges to stretch pupils so they may reach their full potential.

What cross-curricular themes have been identified?

Thinking:

- Problem solving – non-routine, expert /computational thinking, metacognition, creativity
- Systems thinking – decision making and reasoning
- Critical thinking

Numeracy:

- Data types – integer, real / floating point, character, string, Boolean
- Binary – representation of numbers in a Base 2, conversion of binary to denary and vice versa, binary addition and subtraction, shifts
- Hexadecimal – representation of numbers in Base 16, conversion of hexadecimal to denary and vice versa
- Binary / Hexadecimal – conversion of hexadecimal to binary and vice versa
- Sign and magnitude and two's complement to represent negative numbers in binary
- Representation and normalisation of floating-point numbers in binary
- Floating point arithmetic, positive and negative numbers, addition and subtraction
- Calculation of file size and capacity requirements
- Computational logic, logic diagrams and truth tables
- Bitwise manipulation and masks
- Manipulate Boolean expressions and Karnaugh maps
- Boolean algebra
- Identifying and using suitable test data
- Programming – mathematical reasoning, logical operators, arithmetic operators, comparison operators
- Flowcharts – sequencing instructions, symbols
- Graphs – digitising sound, measuring frequency, sample rate and sample size
- Non-calculator papers at GCSE and A-Level

Literacy:

- Communicating information for a given purpose and audience
- Using search engines - identifying and using key words, selecting information that is valid, reliable, trustworthy for use
- Programming – using appropriate and correct syntax to structure and sequence instructions, spelling of variable names
- Use of appropriate conventions
- Key words and computing specific terminology – used appropriately in written work and when verbalising responses
- Promotion of speaking and listening skills within questioning, class discussion, group work
- Proofreading, spelling and grammar checks on work
- Proofreading and debugging algorithms / programs
- Encryption
- Extended response questions – ability to construct and develop a sustained line of reasoning

SMSC:

- Student work is monitored through LAN School
- Appropriate use policy is reinforced with learners
- Using a range of technology safely, respectfully, responsibly and securely
- Protecting their online identity and privacy
- Recognising inappropriate content, contact and conduct
- Knowing how to report concerns
- Understanding and discussing the impact of technology – social, ethical, moral, environmental, legal
- Teamwork, negotiation, trust, conflict resolution
- Digital divide – impact on individuals and groups in society – benefits, power, drawbacks / disadvantages
- Gender – consideration of case studies to reflect the positive contribution made by female practitioners
- Inclusivity / Diversity – consideration of case studies to reflect the positive contribution made by those that represent the diverse nature of our community and society

How will this be assessed to show that learners have learnt and remembered what they have been taught?

	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
Year 7	Module Test	Module Test	Portfolio Assessment	Portfolio Assessment	Year 7 Exam Module Test	Portfolio Assessment
Year 8	Module Test	Module Test	Portfolio Assessment	Baseline Test Portfolio Assessment	Year 8 Exam Portfolio Assessment	Portfolio Assessment
Year 9	Portfolio Assessment	Portfolio Assessment	Year 9 Exam Portfolio Assessment	Portfolio Assessment	Module Test	Portfolio Assessment
KS4 Year 1	Topic Tests for 2.4.1, 1.2.3, 1.2.4	Topic Tests for 2.1.2, 2.2.1, 2.2.2 Practical Programming Assessment covering 2.1.1, 2.1.2, 2.2.1, 2.2.2	Topic Tests for 2.2.3 Practical Programming Assessment covering 2.1.1, 2.1.2, 2.2.1, 2.2.2, 2.2.3	Topic Tests for 1.2.4, 1.1.1, 1.1.2 Baseline Test	Topic Tests for 1.1.3, 1.2.1, 1.2.2	Topic Tests for 1.3.1, 1.3.2 Practical Programming Assessment covering 2.1.1, 2.1.2, 2.2.1, 2.2.2, 2.2.3 Exam Week
KS4 Year 2	Topic Tests 1.4.1, 1.4.2, 1.5.1, 1.5.2	Topic Tests 1.6.1, 2.3.1, 2.3.2, 2.5.1, 2.5.2 Mock Examination	Topic Tests 2.1.3 Practical Programming Assessment covering 2.1.1, 2.1.2, 2.1.3, 2.2.1, 2.2.2, 2.2.3, 2.3.2, 2.5.1, 2.5.2 Mock Examination	Practice Exam Questions Practical Programming Assessments	Practice Exam Questions Practical Programming Assessments Live / External Exams	

	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
Year 12	Topic Tests 2.2.1 a, 1.4.1 a-h Exam Week	Topic Tests 2.2.1 b-d, 2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.1.5, 2.2.2	Topic Tests 1.1.1, 1.1.2, 1.1.3, 1.2.1, 1.2.2	Topic Tests 1.2.3, 1.2.4, 1.3.1, 1.3.2 a, c, d	Topic Tests 1.3.4, 1.4.2	Topic Tests 1.5.1, 1.5.2, 2.3.1 a-b Exam Week – AS Mock Paper NEA Programming Project - Proposal
Year 13	Topic Tests 1.3.2, 1.3.3, 1.4.3 NEA Programming Project - Analysis	Mock A-Level Papers NEA Programming Project - Design	Topic Tests 2.3.1, 1.5.2 NEA Programming Project - Implementation	Mock A-Level Papers NEA Programming Project – Implementation / Evaluation Project Submission	Exam Practice Past Papers	Live / External Exams

Key Stage 4 – GCSE Computer Science 2022-2023

Due to changes to the curriculum model, current Year 11 are now in the third year of a three-year course rather than a two-year. This has an impact on the aforementioned sequencing.

Year 11 Option Groups (Year 3 of 3)

Year 1 for this group followed KS4 Year 1 and Year 3 are following KS4 Year 2 – see page 21. Year 2 will focussed on programming skills and followed the sequence below.

	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
Year 10 (KS4 Year 2 of 3)	Theory and Practical Programming – 2.2.1, 2.2.2 and 2.2.3 (basic skills) Baseline Test	Theory and Practical Programming – 2.2.1, 2.2.2 and 2.2.3 (basic skills) Programming Challenges linked to topics covered	Theory and Practical Programming – 2.2.3, 2.3.1, 2.3.2 (advanced skills) Programming Challenges linked to topics covered	Programming Projects Exam Question Practice Baseline Test	Review of Year 1 Knowledge – 1.2.1, 1.2.2, 1.2.3, 1.2.4, 1.2.5, 2.1.1, 2.1.2, 2.4.1	Review of Year 1 Knowledge – 1.1.1, 1.1.2, 1.1.3, 1.3.1, 1.3.2 Exam Week