ST MARTIN-IN-THE-FIELDS HIGH SCHOOL FOR GIRLS

Computer Science Framework for Learning 2021-2022

Year Group	Year 7					
Subject Intent	Computing ensures that pupils become digitally literate – able to use, and express themselves and develop their ideas through, information and communication technology – at a level suitable for the future workplace and as active participants in a digital world. High-quality computing education equips pupils to use computational thinking and creativity to understand and change the world. Computing has deep links with mathematics, science, and design and technology, and provides insights into both natural and artificial systems. The core of computing pupils are taught the principles of information and computation, how digital systems work, and how to put this knowledge to use through programming. Building on this knowledge and understanding, pupils are equipped to use information technology to create programs, systems and learn a range of content.					
Subject Implementation	**Autumn 1**	**Autumn 2**	**Spring 1**	**Spring 2**	**Summer 1**	**Summer 2**
Knowledge	★ Introduction to SMITF School VLE and School Network ★ 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 ★ Create, reuse, revise and repurpose digital artefacts for a given audience, with attention to trustworthiness, design and usability		★ Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems ★ Use 2 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		★ 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	
Skills	**Unit 1 - Impact of technology – Collaborating online respectfully** → Create a memorable and secure password for an account on the school network		**Unit 2 - Introduction to Python programming - 1** → Describe what algorithms and programs are and how they differ → Recall that a program written in a programming		**Unit 3 - Computing systems** → Recall that a general-purpose computing system is a device for executing programs → Recall that a program is a sequence of instructions	

	→ Remember the rules of the computing lab → Find personal documents and common applications → Recognise a respectful email → Construct an effective email and send it to the correct recipients → Describe how to communicate with peers online → Plan effective presentations for a given audience → Describe cyberbullying → Explain the effects of cyberbullying → Plan effective presentations for a given audience → Describe cyberbullying → Explain the effects of cyberbullying → Check who you are talking to online	language needs to be translated in order to be executed by a machine → Write simple Python programs that display messages, assign values to variables, and receive keyboard input → Locate and correct common syntax errors → Describe the semantics of assignment statements → Use simple arithmetic expressions in assignment statements to calculate values → Receive input from the keyboard and convert it to a numerical value → Use relational operators to form logical expressions → Use binary selection (if, else statements) to control the flow of program execution	that specify operations that are to be performed on data → Explain the difference between a general-purpose computing system and a purpose-built device → Describe the function of the hardware components used in computing systems → Describe how the hardware components used in computing systems work together in order to execute programs → Recall that all computing systems, regardless of form, have a similar structure ('architecture') → Analyse how the hardware components used in computing systems work together in order to execute programs → Define what an operating system is, and recall its role in controlling program execution → Describe how hardware is built out of increasingly complex logic circuits → Recall that, since hardware is built out of logic circuits, data and instructions alike need to be represented using binary digits → Provide broad definitions of 'artificial intelligence' and machine learning' → Identify examples of artificial intelligence and machine learning in the real world → Associate the use of artificial intelligence with moral dilemmas → Explain the implications of sharing program code
Subject Impact	This unit has been designed to ensure that learners are given sufficient time to familiarise themselves with the school network. It also allows the teacher to discuss appropriate use of the school network, and to	This unit introduces learners to text-based programming with Python. The lessons form a journey that starts with simple programs involving input and output, and gradually moves on through arithmetic	This unit takes learners on a tour through the different layers of computing systems: from programs and the operating system, to the physical components that store and execute these programs, to the

	update and remind learners of important online safety issues. Whilst completing this unit, learners will also learn how to use presentation software effectively. In terms of online safety, this unit focuses on respecting others online, spotting strangers, and the effects of cyberbullying.	operations, randomness, and selection. A range of pedagogical tools is employed throughout the unit, with the most prominent being pair programming, live coding, and worked examples.	fundamental binary building blocks that these components consist of. The aim is to provide a concise overview of how computing systems operate, conveying the essentials and abstracting away the technical details that might confuse or put off learners.
Assessment	★ 2 Formative Assessments, ***for e.g. Class work, Homework, Presentation, Short Recall Test, Practical Project, Quiz*** ★ 1 Summative assessment, ***for e.g. Internal School Examinations, End of Topic/Unit Tests*** **(Summative assessment covers content taught in Autumn Term)**	★ 2 Formative Assessments, ***for e.g. Class work, Homework, Presentation, Short Recall Test, Practical Project, Quiz*** ★ 1 Summative assessment, ***for e.g. Internal School Examinations, End of Topic/Unit Tests*** **(Summative assessment covers content taught in Autumn and Spring Term)**	★ 2 Formative Assessments, ***for e.g. Class work, Homework, Presentation, Short Recall Test, Practical Project, Quiz*** ★ 1 Summative assessment, ***for e.g. Internal School Examinations, End of Topic/Unit Tests*** **(Summative assessment covers content taught in the curriculum over the whole academic year)**

Year Group	**Year 8**					
Subject Intent	Computing ensures that pupils become digitally literate – able to use, and express themselves and develop their ideas through, information and communication technology – at a level suitable for the future workplace and as active participants in a digital world. High-quality computing education equips pupils to use computational thinking and creativity to understand and change the world. Computing has deep links with mathematics, science, and design and technology, and provides insights into both natural and artificial systems. The core of computing pupils are taught the principles of information and computation, how digital systems work, and how to put this knowledge to use through programming. Building on this knowledge and understanding, pupils are equipped to use information technology to create programs, systems and learn a range of content.					
Subject Implementation	**Autumn 1**	**Autumn 2**	**Spring 1**	**Spring 2**	**Summer 1**	**Summer 2**
Knowledge	★ Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems ★ 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, reuse, revise and repurpose digital artefacts for a given audience, with attention to trustworthiness, design and usability ★		★ 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 ★ Use 2 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	
Skills	**Unit 1 - Spreadsheets** → Identify columns, rows, cells, and cell references in spreadsheet software → Use formatting techniques in a spreadsheet		**Unit 2 - Developing for the web** → Describe what HTML is → Use HTML to structure static web pages → Modify HTML tags using inline styling to improve		**Unit 3 - Introduction to Python programming - 2** → Use multi-branch selection (if, elif, else statements) to control the flow of program execution → Describe how iteration (while statements) controls	

	→ Use basic formulas with cell references to perform calculations in a spreadsheet (+, -, *, /) → Use the autofill tool to replicate cell data → Explain the difference between data and information → Explain the difference between primary and secondary sources of data → Collect data → Analyse data → Create appropriate charts in a spreadsheet → Use the functions SUM, COUNTA, MAX, and MIN in a spreadsheet → Analyse data → Use a spreadsheet to sort and filter data → Use the functions AVERAGE, COUNTIF, and IF in a spreadsheet → Use conditional formatting in a spreadsheet → Apply all of the spreadsheet skills covered in this unit	the appearance of web pages → Display images within a web page → Apply HTML tags to construct a web page structure from a provided design → Describe what CSS is → Use CSS to style static web pages → Assess the benefits of using CSS to style pages instead of in-line formatting → Describe what a search engine is → Explain how search engines 'crawl' through the World Wide Web and how they select and rank results → Analyse how search engines select and rank results when searches are made → Use search technologies effectively → Discuss the impact of search technologies and the issues that arise by the way they function and the way they are used → Create hyperlinks to allow users to navigate between multiple web pages → Implement navigation to complete a functioning website	the flow of program execution → Use iteration (while loops) to control the flow of program execution → Use variables as counters in iterative programs → Combine iteration and selection to control the flow of program execution → Use Boolean variables as flags
Subject Impact	The unit takes learners from having very little knowledge of spreadsheets to being able to confidently model data with a spreadsheet. The unit uses engaging activities to progress learners from	In this unit, learners will explore the technologies that make up the internet and World Wide Web. Starting with an exploration of the building blocks of the World Wide Web, HTML, and CSS, learners will investigate	The **Y7 - Introduction to Python Programming - 1** unit is a prerequisite for this unit. The unit of work continues to form a journey that

	using basic formulas to writing their own COUNTIF statements.	how websites are catalogued and organised for effective retrieval using search engines. By the end of the unit, learners will have a functioning website.	starts with simple programs involving input and output, and gradually moves on through arithmetic operations, randomness, selection, and now iteration. Emphasis is placed on tackling common misconceptions and elucidating the mechanics of program execution. A range of pedagogical tools is employed throughout the unit, with the most prominent being pair programming, live coding, and worked examples.
Assessment	★ 2 Formative Assessments, ***for e.g. Class work, Homework, Presentation, Short Recall Test, Practical Project, Quiz*** ★ 1 Summative assessment, ***for e.g. Internal School Examinations, End of Topic/Unit Tests*** **(Summative assessment covers content taught in Autumn Term)**	★ 2 Formative Assessments, ***for e.g. Class work, Homework, Presentation, Short Recall Test, Practical Project, Quiz*** ★ 1 Summative assessment, ***for e.g. Internal School Examinations, End of Topic/Unit Tests*** **(Summative assessment covers content taught in Autumn and Spring Term)**	★ 2 Formative Assessments, ***for e.g. Class work, Homework, Presentation, Short Recall Test, Practical Project, Quiz*** ★ 1 Summative assessment, ***for e.g. Internal School Examinations, End of Topic/Unit Tests*** **(Summative assessment covers content taught in the curriculum over the whole academic year)**

Year Group	**Year 9**					
Subject Intent	Computing ensures that pupils become digitally literate – able to use, and express themselves and develop their ideas through, information and communication technology – at a level suitable for the future workplace and as active participants in a digital world. High-quality computing education equips pupils to use computational thinking and creativity to understand and change the world. Computing has deep links with mathematics, science, and design and technology, and provides insights into both natural and artificial systems. The core of computing pupils are taught the principles of information and computation, how digital systems work, and how to put this knowledge to use through programming. Building on this knowledge and understanding, pupils are equipped to use information technology to create programs, systems and learn a range of content.					
Subject Implementation	**Autumn 1**	**Autumn 2**	**Spring 1**	**Spring 2**	**Summer 1**	**Summer 2**
Knowledge	★ 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		★ 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 [for example, ones for sorting and searching]; use logical reasoning to compare the utility of alternative algorithms for the same problem ★ Use 2 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		★ 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	
Skills	**Unit 1 - Representations – from clay to silicon** → List examples of representations → Recall that representations are used to store, communicate, and process information		**Unit 2 - Python programming with sequences of data** → Write programs that display messages, receive keyboard input, and use simple arithmetic expressions in assignment statements		**Unit 3 - Cyber Security** → Explain the difference between data and information → Critique online services in relation to data privacy	

	→ Provide examples of how different representations are appropriate for different tasks → Recall that characters can be represented as sequences of symbols and list examples of character coding schemes → Measure the length of a representation as the number of symbols that it contains → Provide examples of how symbols are carried on physical media → Explain what binary digits (bits) are, in terms of familiar symbols such as digits or letters → Measure the size or length of a sequence of bits as the number of binary digits that it contains → Describe how natural numbers are represented as sequences of binary digits → Convert a decimal number to binary and vice versa → Convert between different units and multiples of representation size → Provide examples of the different ways that binary digits are physically represented in digital devices → Describe the NOT, AND, and OR logical operators, and how they are used to form logical expressions → Use logic gates to construct logic circuits, and associate these with logical operators and expressions → Create complex logic circuits with 2 or more logic gates. → Describe how hardware is built out of increasingly complex logic circuits → Recall that, since hardware is built out of logic circuits, data and instructions alike need to be represented using binary digits → Apply all of the skills covered in this unit	→ Locate and correct common syntax errors → Create lists and access individual list items → Use selection (**if-elif-else* statements) to control the flow of program execution → Perform common operations on lists or individual items → Use iteration (while statements) to control the flow of program execution → Perform common operations on lists or individual items → Perform common operations on strings or individual characters → Use iteration (for statements) to iterate over list items → Perform common operations on lists or strings → Use iteration (for loops) to iterate over lists and strings → Use variables to keep track of counts and sums → Combine key programming language features to develop solutions to meaningful problems → Apply all of the skills covered in this unit	→ Identify what happens to data entered online → Explain the need for the Data Protection Act → Recognise how human errors pose security risks to data → Implement strategies to minimise the risk of data being compromised through human error → Define hacking in the context of cyber security → Explain how a DDoS attack can impact users of online services → Identify strategies to reduce the chance of a brute force attack being successful → Explain the need for the Computer Misuse Act → List the common malware threats → Examine how different types of malware causes problems for computer systems → Question how malicious bots can have an impact on societal issues → Compare security threats against probability and the potential impact to organisations → Explain how networks can be protected from common security threats → Identify the most effective methods to prevent cyberattacks
Subject Impact	This unit conveys essential knowledge relating to binary representations and . The activities gradually introduce learners to binary digits and how they can be used to represent text and numbers. The concepts	This unit introduces learners to how data can be represented and processed in sequences, such as lists and strings. The lessons cover a spectrum of operations on sequences of data, that range from	This unit takes the learners on an eye-opening journey of discovery about techniques used by cybercriminals to steal data, disrupt systems, and infiltrate networks. The learners will start by considering the value of

	are linked to practical applications and problems that the learners are familiar with.	accessing an individual element to manipulating the entire sequence. ***N.B. The Year 7 and 8 Programming units are prerequisites for this unit. It is assumed that learners are already able to write Python programs that display messages, receive keyboard input, use simple arithmetic expressions, and control the flow of program execution through selection and iteration structures.***	their data to organisations and what they might use it for. They will then look at social engineering techniques used by cybercriminals to try to trick users into giving away their personal data. The unit will look at the more common cybercrimes such as hacking, DDoS attacks, and malware, as well as looking at methods to protect ourselves and our networks against these attacks
Assessment	★ 2 Formative Assessments, ***for e.g. Class work, Homework, Presentation, Short Recall Test, Practical Project, Quiz*** ★ 1 Summative assessment, ***for e.g. Internal School Examinations, End of Topic/Unit Tests*** **(Summative assessment covers content taught in Autumn Term)**	★ 2 Formative Assessments, ***for e.g. Class work, Homework, Presentation, Short Recall Test, Practical Project, Quiz*** ★ 1 Summative assessment, ***for e.g. Internal School Examinations, End of Topic/Unit Tests*** **(Summative assessment covers content taught in Autumn and Spring Term)**	★ 2 Formative Assessments, ***for e.g. Class work, Homework, Presentation, Short Recall Test, Practical Project, Quiz*** ★ 1 Summative assessment, ***for e.g. Internal School Examinations, End of Topic/Unit Tests*** **(Summative assessment covers content taught in the curriculum over the whole academic year)**

Year Group	**Year 10 [OCR GCSE COMPUTER SCIENCE - J277]**					
Subject Intent	Computer Science encourages students 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					
Subject Implementation	**Autumn 1**	**Autumn 2**	**Spring 1**	**Spring 2**	**Summer 1**	**Summer 2**
Knowledge	Student will develop an understanding of: **Component 1:** ➔ Architecture of the CPU ➔ CPU Performance ➔ Embedded systems ➔ Primary storage (Memory) ➔ Secondary storage **Component 2:** ➔ Computational thinking ➔ Designing, creating and refining algorithms ➔ Programming fundamentals ➔ Defensive design		Student will develop an understanding of: **Component 1:** ➔ Units - Data storage ◆ Numbers ◆ Characters ◆ Images ◆ Sound ➔ Compression **Component 2:** ➔ Data types ➔ Additional programming techniques ➔ Boolean logic		Student will develop an understanding of: **Component 1:** ➔ Threats to computer systems and networks ➔ Identifying and preventing vulnerabilities ➔ Operating systems ➔ Utility software **Component 2:** ➔ Testing	
	Practical Programming - All students must be given the opportunity to undertake a programming task(s), either to a specification or to solve a problem (or problems), during their course of study. Students may draw on some of the content in **both components** when engaged in Practical Programming.					
Skills	**COMPONENT 1** **Students will study:** **1.1.1 Architecture of the CPU**		**COMPONENT 1** **Students will study:** **1.2.3 Units**		**COMPONENT 1** **Students will study:** **1.4.1 Threats to computer systems and networks**	

	- the purpose of the CPU o the fetch-execute cycle - common CPU components and their function: o ALU (Arithmetic Logic Unit), CU (Control Unit), Cache, Registers - Von Neumann architecture: o MAR (Memory Address Register), MDR (Memory Data Register), Program Counter, Accumulator **1.1.2 CPU Performance** - how common characteristics of CPUs affect their performance: o Clock speed, Cache size, Number of Cores **1.1.3 Embedded systems** - the purpose and characteristics of embedded systems - Examples of embedded systems **1.2.1 Primary storage (Memory)** - The need for primary storage - The difference between RAM and ROM - The purpose of ROM in a computer system - The purpose of RAM in a computer system - Virtual memory **1.2.2 Secondary storage** - The need for secondary storage - Common types of storage: o Optical, Magnetic, Solid state - Suitable storage devices and storage media for a given application - The advantages and disadvantages of different storage devices and storage media relating to these characteristics: o Capacity, Speed, Portability, Durability, Reliability, Cost **COMPONENT 2**	- The units of data storage: o Bit, Nibble (4 bits), Byte (8 bits), Kilobyte (1000, (bytes or 1 KB), Megabyte (1,000 KB), Gigabyte (1,000 MB), Terabyte (1,000 GB), Petabyte (1,000 TB) - How data needs to be converted into a binary format to be processed by a computer. - Data capacity and calculation of data capacity requirements **1.2.4 Data storage** **Numbers** - How to convert positive denary whole numbers to binary numbers (up to and including 8 bits) and vice versa - How to add two binary integers together (up to and including 8 bits) and explain overflow errors which may occur - How to convert positive denary whole numbers into 2-digit hexadecimal numbers and vice versa - How to convert from binary to hexadecimal equivalents and vice versa - Binary shifts **Characters** - The use of binary codes to represent characters - The term 'character-set' - The relationship between the number of bits per character in a character set, and the number of characters which can be represented, e.g.: o ASCII, Unicode **Images** - How an image is represented as a series of pixels, represented in binary - Metadata - The effect of colour depth and resolution on: o The quality of the image, The size of an image file	- Forms of attack o Malware, Social engineering, e.g. phishing, people as the 'weak point', Brute-force attacks, Denial of service attacks, Data interception and theft, The concept of SQL injection **1.4.2 Identifying and preventing vulnerabilities** - Common prevention methods: o Penetration Testing, Anti-malware software, Firewalls, User access levels, Passwords, Encryption, Physical Security **1.5.1 Operating systems** - The purpose and functionality of operating systems: o User interface, Memory management and multitasking, Peripheral management and drivers, User management, File management **1.5.2 Utility software** - The purpose and functionality of utility software - Utility system software: o Encryption software, Defragmentation, Data Compression **COMPONENT 2** **Students will study:** **2.3.2 Testing** - The purpose of testing - Types of testing: o Iterative, Final/terminal - Identify syntax and logic errors, Selecting and using suitable test data o Normal, Boundary, Invalid, Erroneous - Refining algorithms

	Students will study: **2.1.1 Computational thinking** - Principles of computational thinking o Abstraction, Decomposition, Algorithmic Thinking. **2.1.2 Designing, creating and refining algorithms** - Identify the inputs, processes, and outputs for a problem - Structure diagrams - Create, interpret, correct, complete, and refine algorithms using: o Pseudocode, Flowcharts, Reference language/high-level programming language - Identify common errors - Trace tables - Standard searching algorithms: o Binary search, Linear search - Standard sorting algorithms: o Bubble sort, Merge sort, Insertion sort **2.2.1 Programming fundamentals** - The use of variables, constants, operators, inputs, outputs and assignments - The use of the three basic programming constructs used to control the flow of a program: o Sequence, Selection, Iteration (count- and condition- controlled loops) - The common arithmetic operators - The common Boolean operators AND, OR, NOT **2.3.1 Defensive design** - Defensive design considerations: o Anticipating misuse, Authentication - Input validation - Maintainability: o Use of sub programs, Naming conventions,	**Sound** - How sound can be sampled and stored in digital form - The effect of sample rate, duration and bit depth on: o The playback quality, The size of a sound file **1.2.5 Compression** - The need for compression - Types of compression: o Lossy, Lossless **COMPONENT 2** **Students will study:** **2.2.2 Data types** - The use of data types: o Integer, Real, Boolean, Character and string, Casting **2.2.3 Additional programming techniques** - The use of basic string manipulation - The use of basic file handling operations: o Open, Read, Write, Close - The use of records to store data - The use of SQL to search for data - The use of arrays (or equivalent) when solving problems, including both one-dimensional (1D) and two-dimensional (2D) arrays - How to use subprograms (functions and procedures) to produce structured code - Random number generation - Defensive design considerations: o Anticipating misuse, Authentication - Input validation - Maintainability: o Use of sub programs, Naming conventions, Indentation, Commenting	

	Indentation, Commenting	**2.4.1 Boolean logic** - Simple logic diagrams using the operations AND, OR and NOT - Truth tables - Combining Boolean operators using AND, OR and NOT - Applying logical operators in truth tables to solve problems	
Subject Impact	The GCSE Computer Science course will enable students to develop a real, in-depth understanding of how computer technology works, giving them an insight into what goes on 'under the lid' of a computer. You will need to think creatively, innovatively and logically to design and program solutions to real-world problems. Students will investigate the components that make up digital systems and how they communicate with one another and with other systems. They will also develop an understanding of the impacts of digital technology to the individual and to the wider society.		
Assessment	★ Formative Assessments at the end of each lesson topic, ***for e.g. Class work, Homework, Presentation, Short Recall Test, Practical Project, Quiz*** ★ 1 Summative assessment, ***for e.g. Internal School Examinations, End of Topic/Unit Tests*** **(Summative assessment covers content taught in Autumn Term)**	★ Formative Assessments at the end of each lesson topic, ***for e.g. Class work, Homework, Presentation, Short Recall Test, Practical Project, Quiz*** ★ 1 Summative assessment, ***for e.g. Internal School Examinations, End of Topic/Unit Tests*** **(Summative assessment covers content taught in Autumn and Spring Term)**	★ Formative Assessments at the end of each lesson topic, ***for e.g. Class work, Homework, Presentation, Short Recall Test, Practical Project, Quiz*** ★ 1 Summative assessment, ***for e.g. Internal School Examinations, End of Topic/Unit Tests*** **(Summative assessment covers content taught in the curriculum over the whole academic year)**

Year Group	Year 11 [OCR GCSE COMPUTER SCIENCE - J277]					
Subject Intent	Computer Science encourages students 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					
Subject Implementation	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
Knowledge	Student will develop an understanding of: **Component 1:** ➔ Networks and topologies ➔ Wired and wireless networks, protocols and layers **Component 2:** ➔ Languages ➔ The Integrated Development Environment (IDE)		Student will develop an understanding of: **Component 1:** ➔ Ethical, legal, cultural and environmental impact **Component 2:** ➔ Final Practical Programming Skills Tasks and Revision/Intervention		GCSE EXAMS	
	Practical Programming - All students must be given the opportunity to undertake a programming task(s), either to a specification or to solve a problem (or problems), during their course of study. Students may draw on some of the content in **both components** when engaged in Practical Programming.					
Skills	**COMPONENT 1** **Students will study:** **1.3.1 Networks and topologies** - Types of networks: o LAN (Local Area Network), WAN (Wide Area		**COMPONENT 1** **Students will study:** **1.6.1 Ethical, legal, cultural and environmental impact**		GCSE EXAMS	

	Network) - Factors that affect the performance of networks - The different roles of computers in a client-server and a peer-to-peer network - The hardware needed to connect stand-alone computers into a Local Area Network: o Wireless access points, Routers, Switches, NIC (Network Interface Controller/Card), Transmission media - The Internet as a worldwide collection of computer networks: o DNS (Domain Name Server), Hosting, The Cloud, Web Servers and Clients, - Star and Mesh network topologies **1.3.2 Wired and wireless networks, protocols and layers** - Modes of connection: o Wired - Ethernet o Wireless - Wi-Fi and Bluetooth - Encryption - IP addressing and MAC addressing - Standards - Common protocols including: o TCP/IP (Transmission Control Protocol/Internet Protocol), HTTP (HyperText Transfer Protocol), HTTPS (HyperText Transfer Protocol Secure), FTP (File Transfer Protocol), POP (Post Office Protocol), IMAP (Internet Message Access Protocol), SMTP (Simple Mail Transfer Protocol), - The concept of layers **COMPONENT 2** **Students will study:** **2.5.1 Languages** - Characteristics and purpose of different levels of	- Impacts of digital technology on wider society including: o Ethical issues o Legal issues o Cultural issues o Environmental issues o Privacy issues - Legislation relevant to Computer Science: o The Data Protection Act 2018 o Computer Misuse Act 1990 o Copyright Designs and Patents Act 1988 o Software licences (i.e. open source and proprietary) **COMPONENT 2** **Students will study:** **Final Practical Programming Skills Tasks and all units Revision/Intervention** All students must be given the opportunity to undertake a programming task or tasks during their course of study. The programming task(s) will allow students to develop skills within the following areas when programming: • Design, Write, Test, Refine Each task(s) will require a high-level text based solution using the programming language **PYTHON**. Practical Programming skills will be assessed in Component 2 of the qualification, in particular Section B.	

	programming language: o High-level languages, Low-level languages - The purpose of translators - The characteristics of a compiler and an interpreter **<u>2.5.2 The Integrated Development Environment (IDE)</u>** - Common tools and facilities available in an integrated development environment (IDE): o Editors o Error diagnostics o Run-time environment o Translators		
Subject Impact	The GCSE Computer Science course will enable students to develop a real, in-depth understanding of how computer technology works, giving them an insight into what goes on 'under the lid' of a computer. You will need to think creatively, innovatively and logically to design and program solutions to real-world problems. Students will investigate the components that make up digital systems and how they communicate with one another and with other systems. They will also develop an understanding of the impacts of digital technology to the individual and to the wider society.		
Assessment	★ Formative Assessments at the end of each lesson topic, ***for e.g. Class work, Homework, Presentation, Short Recall Test, Practical Project, Quiz*** ★ 1 Summative assessment - PPE1 **(Summative assessment covers content taught in GCSE course, to date)**	★ Formative Assessments at the end of each lesson topic, ***for e.g. Class work, Homework, Presentation, Short Recall Test, Practical Project, Quiz*** ★ 1 Summative assessment - PPE2 **(Summative assessment covers entire GCSE specification)**	**GCSE EXAMS**