PONTEFRACT ACADEMIES TRUST

Computing Overview Key Stage 3 2025/26

Year 7

Term	1	4	3	3	5	6
Topic	**E-Safety**	**Computational Thinking**	**Micro:bit**	**Data Storage**	**JavaScript**	**Digital Solutions**
Enquiry Question	Is the internet a safe and inclusive environment?	How can we learn to solve computational problems efficiently?	How can we develop coding skills working with materials and electronics?	How do computers make decisions and perform instructions?	How can we create computer programmes by writing lines of code?	How can we use different digital tools together to solve a real problem?
Big Ideas/ Key Concepts	• The internet can be a place where young people are exploited • The internet is not always safe • The ways and means we can protect ourselves and others when online	• Abstraction as a way to remove unnecessary information • Decomposition as a tool to break problems down • Pattern recognition in problems • Flowcharts to plan algorithms • Pseudocode to plan algorithms	• Using variables to store data and information • Using loops to perform iteration • Using print statements to execute messages • Using coding skills to develop animated programs	• All data is represented as binary digits • Calculations can be performed in binary • Hexadecimal (hex) is also a commonly used number system • Programs use simple comparisons to help make decisions • Boolean (binary) logic is a form of algebra where all values are	• Using variables to store data • Creating arrays to store multiple values • Using methods to manipulate list contents • Using casting to convert user input into numbers	• Using technology not just to consume, but to design and make original products. • Selecting and combining multiple applications and devices to achieve a goal. • Collecting, organising, and interpreting data to draw conclusions or support decision-making. • Creating digital outcomes that

				either True or False		meet the needs of a specific audience or user group.
Key Knowledge and Skills	• To describe ways in which we can stay safe online • To describe the most effective ways to create and store passwords • To share and present with others	• To describe what the four branches of computational thinking are • To be able to solve computational thinking problems	• To describe iteration • To describe what a print statement does • To be able to run code on a micro:bit • To be able to create a 'for' loop • To be able to create a 'while' loop	• How to convert denary, binary and hexadecimal • How to perform binary number addition • To calculate an output from logic circuits and to state an output from logic gates	• To be able to write error free and robust code • To able to apply problem solving skills in a range of real-life scenarios • To identify and amend syntax errors • To be able identify variables and describe the purpose of a given line of code	• Knowing how to choose and use a range of software (e.g., spreadsheets, presentation tools, coding platforms) and combine them effectively. • Understanding how to collect, organise, and analyse data, then present it clearly using charts, graphs, or digital visualisations. • Working with others, often across devices, to share ideas and produce joint digital projects. • Applying user-focused design: identifying an audience, planning a solution, testing it, and

						refining based on feedback.
End Point	Students are aware of the dangers when browsing the web and how to use a mobile device safely Students can list and describe ways in which to stay safe online	Students can apply computational thinking to a range of problems when using a computer	Students can use coding skills to build their own programs using a micro:bit Students can analyse and annotate their code, finding errors and areas of improvement	Students understand how numbers are represented in binary and hex and can carry out simple operations on binary numbers Students understand simple Boolean logic [AND, OR and NOT] and some uses in circuits and programming	Students can confidently write robust code and display their work through a well kept and organised programming diary	By the end of this unit, pupils will be able to design and complete a creative digital project that combines multiple applications and devices. They will collect and analyse data, make purposeful choices about tools, and produce an outcome that effectively meets the needs of identified users.
Prior Knowledge	Real-life experience of the internet and web applications	Creating algorithms (micro:bit)	KS2 blockly coding (Scratch and code.org)	Mathematical operations Sums	Creating algorithms (micro:bit) Problem Solving (Computational Thinking)	Prior use of Google Docs and Slides, including limited formatting/research skills and programming skills developed in terms 2 and 5.
Common Misconceptions	• Passwords are short and don't include numbers and symbols • What we visit online is not recorded/tracked	• Pattern recognition is looking for the same thing in two different scenarios • Computational thinking is how computers think • Decomposition is similar to baking	• Interpreting errors messages • Ensuring blocks of code are embedded within the micro:bit and not standalone	• Assuming all hexadecimal numbers must include letters Logic gates are not made of anything else (no prior knowledge of	• Confusing assignments with equality • Believing Javascript is the same as standard English	• One app does everything. • Data is just numbers. • The project is for me, not the user.

				switches / transistors)	• Missing brackets or ; • The concept of variables storing data	
Key Words	• Safety • Digital footprint • Radicalisation • Relationships • Actions and effects • Threats	• Abstract • Decompose • Pattern • Flowchart • Algorithm • Pseudocode	• Loop • Print • Sound • Variable • For • While	• Binary • Denary • Hexadecimal • Boolean • Logic	• Variable • Array • Const • Let • Console.log	• Application • Device • Data • Analysis • Audience • Integration • Creativity • User

PONTEFRACT ACADEMIES TRUST

Year 8

Term	1	2	3	4	5	6
Topic	**E-Safety**	**Small Basic**	**Computer Systems**	**Python**	**Search & Sort Algorithms**	**Ethics & Law**
Enquiry Question	How can we behave in a way that is appropriate and respectful online?	How can we use a programming language to follow/execute instructions?	How does hardware and software determine our use and control of computer systems?	How can we use a high-level programming language to create simple programmes?	How do different algorithms solve the same problem in faster or slower ways, and why does that matter?	What role does AI play in our lives?
Big Ideas/ Key concepts	• Online identity – protecting ourselves/others • Correctly presenting ourselves online • Security – how to safeguard social media/our data • Risks (spam, hate, phishing, viruses)	• Using a programming language to perform instructions on a keyboard	• The role and importance of inputs and outputs • The role of hardware within a computer system • The role of primary and secondary storage	• Using commands, inputs and outputs to create Python animations • Understanding how functions and loops work in Python	• Compare the efficiency of searching and sorting algorithms • Understand the importance of algorithm choice in software development • Think critically about when and why to use one algorithm over another	• Copyright • GDPR • Data protection • AI (Artificial Intelligence) • Computers in the workplace • Deepfakes
Key Knowledge and skills	• Students can identify spam, phishing and inappropriate content • Know how to set up social media accounts so that they are private	• Students can use functions • Students develop problem solving skills • Students develop computational thinking skills	• Students can identify what inputs and outputs are • Students can identify the purpose of primary and secondary storage • Students can identify the roles of each component of computer hardware	• Students can use problem solving skills in code • Students can read and code using a high-level programming language	• Students can discuss and explain the difference between binary and linear search algorithms • students can practically explore how bubble,	• Students can describe features of key laws and acts associated with computing • Students can identify impacts of technology and how these are experienced,

	• Students can be respectful whilst browsing and interacting with others online			• Students can identify and rectify logic and syntax errors in code	insertion and merge sort algorithms handle data sets	negated or adapted to.

End Point	Students can identify risks when browsing the internet Students are able to differentiate between email and spam Students are aware of the help available if attacked online	Students can create effective code and programs that perform simple to intermediate tasks	Students can identify key software and hardware of a computer system Students can make informed decisions about what equipment is best suited for a specific task or scenario	Students can write error free and robust code using a high level programming language	students can both practically and theoretically demonstrate how common search and sort algorithms work	Students can understand the impact of technology on individuals, organisations and the planet through a range of real-world examples
Prior Knowledge	E-safety – is the internet safe?	Computational thinking	User experience of computer systems	Small Basic	Computational thinking Programming & Selection	E-safety and Misinformation
Common Misconceptions	• Spam is sent to elderly and vulnerable only • It is easy to block someone online • There are no repercussions to your behaviour and content share online	• Confusing TextWindow.ReadNumber() with TextWindow.Read() for numeric input • Forgetting that arrays in Small Basic start at index 1, not 0 • Using incorrect syntax for loops and conditionals (e.g. missing EndFor, EndIf)	• Software is tangible • Computers are just used in schools/offices • A phone or tablet is not a computer	• Using = instead of == in conditional statements • Misunderstanding indentation and code blocks • Treating strings and numbers as interchangeable without conversion	• Binary search always works faster than linear search • All sorting algorithms work the same way, just with different names.	• Concerns about radiation from mobile phones • Legitimate use of digital content and risks of copyright infringement
Key Words	• Spam • Phishing	• Input • Output	• Memory • Cache	• Input • Output	• Search • Array	• Privacy • Ethical

	• Hate • Cyberbullying • Identity • Passwords	• Line • Print • Loop • Draw • Penup • Pendown • Speed • Variable	• Registers • CPU • Hard drive • Primary / Secondary	• Selection • Casting • Variable	• Sort • Merge • Insert • Ordered	• Environmental • Cultural • Data legislation

PONTEFRACT ACADEMIES TRUST

Year 9

Term	1	2	3	5	5	6
Topic	**E-Safety**	**Python Programming**	**Networks**	**Augmented Reality**	**Processors and Data**	**Boolean Logic**
Enquiry Question	What are some of the dangers and risks to young adults and children when using the internet?	How can I create programs using a high-level programming language?	What are the 'internet' and the 'World Wide Web', what are the benefits of networks?	How can I explore augmented reality to create unique images and graphics?	How does a computer process different data types?	In what ways can different combinations of logic gates be used to build and simplify Boolean expressions?
Big Ideas/ Key concepts	• Types of grooming • Types of online abuse • Reporting abuse • Spotting abuse • The internet as a tool for abuse/spreading hate	• Identifying syntax and logic errors • Creating variables • Creating inputs • Lists and functions • Loops	• Data transmission • Topologies • Wired/Wireless connections • Network hardware • The internet	• Augmented reality • Enhance real artefacts with AR generated information • Designers use Adobe Aero to create AR content	• The purpose of the CPU, common components and their function • Data capacity and file types • Common scenarios when compression may be needed	• Logical decisions can be represented in multiple ways • Boolean expressions and logic circuits can often be simplified • Logic gates are the fundamental building blocks of digital systems • When combined, they form the basis for complex circuits
Key Knowledge and skills	• Students can identify abuse and explain types • Students can discuss how extremists use the internet to groom • Students create effective websites	• Students can create robust code • Students can document their process • Students can reflect on programming	• Students describe components of networks and how they work together • Students know the difference between the internet, its services, and the World Wide Web	• Students can overlay digital content onto real-life environments and objects. • Students are able to create graphics which are both	• Students can identify the main CPU components • Students familiar with data units • Students know storage devices have different capacities • Students know the advantages and	• Recognising and understanding the standard logic gates (AND,OR,NOT) • Being able to create truth tables for given logic gates or expressions, and

	/multimedia content			interactive and immersive	disadvantages of compression	interpret how inputs determine outputs. ● Writing Boolean expressions to represent logic circuits

End Point	Students can identify the various types of abuse and grooming and explain these through their own website	Students can create robust programmes that imitate real-life scenarios	Students can identify the hardware used in computer networks and explain the benefits of different network types	Students can create digital artefacts for a given audience, with attention to design and usability	Students can identify how a computer processor works Students can make informed decisions about data units, types and file compression	By the end of this unit, students will be able to confidently represent, construct, and simplify Boolean expressions and logic circuits, using truth tables and standard gates to design efficient solutions that model real-world computational problems.
Prior Knowledge	E-safety How we behave online	Python, Small Basic, micro:bit	Computer systems, hardware	Prior use of digital graphic applications	Computer systems, hardware	Data Storage and Programming (Small Basic, JS and Python)
Common Misconceptions	• Radicalisation is only connected to religion • Grooming just involves elderly males and young girls • Abuse doesn't hurt if it is only online • Sexual abuse is physical and can't take place online • Live Streaming is harmless	• Python understands plain English • X = X and not * • Brackets don't have to be closed • Spellings are in UK English (Python uses US English)	• Star topology – can assume this is how it is actually configured (server in the middle and nodes in a circle) • Wi-Fi connects you directly to internet • Servers are big	• Differences between AR and AI • Lack of familiar examples of the potential of AR	• Clock speed (More cycles per second doesn't automatically mean faster.) • Some CPUs do more in each clock cycle, or memory may not be able to keep up	• Mixing up gate symbols • NOT flips all values in a circuit • The longest or most literal Boolean expression is always correct • Some assume Boolean values are just numbers (e.g., 1 always means "true" in every context), without recognising they represent states of logic, not quantities.
Key Words	• Extremism • Abuse • Groom • Radicalisation • Incitement • County lines • Exploitation	• Loop • Error • Syntax • Logic • Iteration • Function • Variable • Boolean	• Connectivity • Protocol • Server • Router • Switch • Hub	• Asset • Scene • Tap • Orbit • Audio • Insert • Show • Hide	• CPU • ALU • Control Unit • Registers • Data • Compression	• AND • OR • NOT • Boolean Logic • Expression • Truth Table

Term	1	2	3	4	5	6
Topic	1.1.1 Architecture of the CPU 1.1.2 CPU performance 1.1.3 Embedded systems 1.2.1 Primary storage (Memory) 1.2.2 Secondary storage (Memory) 2.2.1 Programming fundamentals	1.2.3 Units 1.2.4 Data Storage 1.2.5 Compression 1.3.1 Networks and topologies 2.2.1 Programming fundamentals 2.2.2 Data types	1.3.1 Networks and topologies 1.3.2 Wired and wireless networks, protocols and layers 1.4.1 Threats to computer systems and networks 1.4.2 Identifying and preventing vulnerabilities 2.2.1 Programming fundamentals	1.5 Operating systems 1.5.2 Utilities 1.6 Ethical, Environmental, Cultural and Legal Issues 2.2.3 Additional programming techniques	1.6 Ethical, Environmental, Cultural and Legal Issues 2.1.2 Designing, creating and refining algorithms 2.1.3 Searching and sorting algorithms	2.1.1 Computational thinking 2.2.3 Additional programming techniques 2.4.1 Boolean logic
Enquiry Question	"How does the design and performance of computer systems impact our everyday use of technology?"	"How is data stored, shared, and made more efficient in the digital world?"	"How do computer systems connect and stay secure in a world built on code?"	"Who is responsible when technology causes harm?"	"Do digital laws do enough to protect people and their data?"	"How do computers use logic to make decisions?"

PONTEFRACT ACADEMIES TRUST

Big Ideas/ Key Concepts						
	• The CPU is the "brain" of the computer, processing instructions via a structured cycle. • Data and instructions are stored in memory and moved via buses (address, data, control). • CPU performance is about speed and efficiency in executing instructions. • Without sufficient RAM, systems become slower or unstable when multitasking. • Secondary storage is essential for long-term data retention. • The choice of storage affects speed, durability, and system cost. • Programming uses structured logic to solve problems. • Code is written using predictable constructs like loops, conditions, and variables.	• Understanding how data is measured and scaled • Everything we see, hear, and use on a computer is stored as binary data. • Compression enables faster transmission and storage, especially for multimedia. • The structure of a network (topology) affects its performance, reliability, and scalability. • Fundamental skills build the foundation for developing more complex algorithms and programs.	• Everything is Connected • Communication Needs Rules • Digital Systems Must Be Defended • Secure Design is Smart Design • Code Controls the Machine	• The Operating System is the Boss • Utility Software Keeps Systems Healthy • Tech Has Consequences • What is the responsibility of developers and users? • Powerful Code = Flexible Software (The use of sub-programs) • Systems Are Built in Layers (The OS, utilities, hardware, and applications all work together)	• Technology Shapes Society • digital divide • Legal frameworks such as Data Protection Act, Computer Misuse Act, Copyright, Designs and Patents Act and Creative Commons Licensing • Algorithms Make the Digital World Work • Efficient algorithms are essential for fast, reliable computing. • Design basic flowcharts to solve problems	• Think Like a Computer Scientist and use key concepts such as Decomposition, Abstraction ,Algorithmic Thinking and Pattern recognition • Powerful Code Solves Real Problems • Logic Powers Everything • Computers make decisions using simple logic—just 1s and 0s.

PONTEFRACT ACADEMIES TRUST

Key Knowledge and Skills	• Describe the purpose and function of key CPU components • Compare the impact of different factors on CPU performance • Identify and explain the role of embedded systems in real-world devices • Differentiate between primary and secondary storage types • Evaluate the suitability of storage devices for different uses • Write code that features inputs, outputs, casting and loops.	• Convert between different units of data storage • Explain how characters, images, and sound are represented in binary • Describe and compare different methods of compression • Identify and describe different network types and topologies • Use programming constructs and data types to write and understand code	• Describe different types of networks and network topologies • Compare the characteristics of wired and wireless networks • Explain the purpose of common network protocols and the layers of the TCP/IP model • Identify common threats to computer systems and how they affect security • Suggest and explain methods for preventing and detecting cyber security vulnerabilities	• Explain the purpose and functions of an operating system • Describe different types of utility software and their uses • Compare backup types and data management tools • Discuss the ethical, environmental, and cultural impacts of digital technology • Identify and explain key legal issues relating to computer use and data protection	• Discuss the ethical, environmental, and cultural implications of computing technologies • Identify and explain key legal frameworks related to digital technology and data use • Describe how different searching algorithms work and when to use them • Explain how different sorting algorithms function and compare their efficiency • Select and justify appropriate algorithms for given scenarios or data sets	• Apply decomposition, abstraction, and algorithmic thinking to solve problems • Design and use subprograms to structure and simplify code • Read from and write to external files using programming techniques • Use Boolean logic and truth tables to construct and evaluate logical expressions • Implement validation, authentication, and randomisation within programs
End Point	By the end of studying these topics, students should understand how	By the end of studying these topics, students should be able to:	By the end of studying these topics, students should understand how	By the end of studying these topics, students should understand the	By the end of studying these topics, students should be able to	By the end of studying these topics, students should be able to apply

	computers process, store, and retrieve data, including the role of the CPU, different types of memory and storage, and the basics of writing and understanding programs. They should be able to explain how CPU performance is influenced, describe the function of embedded systems, and apply programming fundamentals like variables, control structures, and data types—preparing them for exam questions and practical coding tasks.	Understand how digital data is measured, stored, and represented, including binary units, file sizes, and different data types. Explain how compression reduces file size, and identify when different compression methods are suitable. Describe how computer networks are structured and connected, including common topologies and key components. Apply core programming principles such as variables, selection, iteration, and correct use of data types to solve problems in code.	data travels through networks, the differences between wired and wireless communication, and how protocols and layers work together to ensure effective data transmission. They will also be able to identify common cybersecurity threats, such as malware and phishing, and explain how vulnerabilities can be reduced using technical and human measures. In programming, students will have a solid grasp of fundamental concepts like variables, input/output, selection, and iteration, enabling them to write clear and logical code. This prepares them to approach both theory and practical questions with confidence, particularly in Paper 1 (Computer Systems) and Paper 2 (Computational Thinking, Algorithms and Programming).	role of operating systems in managing hardware and software resources, as well as the purpose of utility software in maintaining system health and performance. They will be aware of the ethical, environmental, cultural, and legal considerations related to computing, including data privacy, sustainability, and intellectual property. Additionally, students will be able to apply advanced programming techniques such as subprograms, file handling, and data validation to create more efficient and reliable code. This knowledge equips students to handle both theoretical questions and practical programming challenges in their exams confidently.	design, create, and refine algorithms that solve problems efficiently and effectively. They will understand how different searching and sorting algorithms work, their advantages and limitations, and how to choose the best approach for a given situation. This knowledge enables students to apply algorithmic thinking in programming tasks and to explain their solutions clearly in exams.	computational thinking skills such as decomposition, abstraction, and pattern recognition to break down complex problems. They will be proficient in using additional programming techniques like subprograms, file handling, and data validation to write efficient and organized code. Additionally, students will understand and use Boolean logic to control program flow and make decisions within algorithms. This foundation prepares them to solve programming challenges confidently and explain their reasoning clearly in exams.

PONTEFRACT ACADEMIES TRUST

Prior Knowledge	Year 8 - Computing Hardware and Software Year 7, 8 and 9 - Computer Programming	Year 7 - Numbers Year 9 - Processing & Storage Year 9 - Networks	Year 9 - Networks	Year 9 - Ethics & Law	Year 8 - Search & Sort Algorithms Year 9 - Ethics & Law	Year 7 - Computational Thinking
Assessment Objectives	AO1 - Demonstrate knowledge and understanding of the key concepts and principles of Computer Science. 30% AO2 - Apply knowledge and understanding of the key concepts and principles of Computer Science. 40% AO3 - Analyse problems in computational terms to make reasoned judgments and to design, program, evaluate and refine solutions. 30%					
Assessment Objectives Covered	AO1 AO2 AO3	AO1 AO2 AO3	AO1 AO2 AO3	AO1 AO2 AO3	AO2 AO3	AO1 AO2 AO3
Assessment Objectives Covered with Examples	AO1 - 'Identify two events that take place during the fetch-execute cycle' AO2 - 'Write the missing arithmetic operator for each algorithm.' AO3 - 'Complete the following pseudocode for an algorithm to count up how many matches with 0 goals are stored in the array and then print out this value.'	AO1 - 'Describe what happens when the computer converts the music into a file.' AO2 - 'Complete the table by writing the missing denary, 8-bit binary or hexadecimal values.' AO3 - 'A programmer creates an algorithm using a flow chart. Write this algorithm using pseudocode'	AO1 - 'Define what is meant by a 'network protocol'. AO2 - 'Give two reasons why the bakery may use a star network topology for their LAN.' AO3 -'A second program needs to perform the following tasks: • Input a number from the user • Double the number input and print the result • Repeat bullets 1 and 2 until the	AO1 - 'Identify three ways Xander can make use of the file management facility' AO2 - 'Describe the environmental impacts of Fiona's decision'' AO3 - 'Write an SQL statement to display the sensor IDs of the door sensors that have been triggered for more than 20 seconds.'	AO2 - 'The flowchart statements have been written for the algorithm, but the flowchart is incomplete. Complete the flowchart.' AO3 - 'Discuss the features, benefits and drawbacks of each type of licence'	AO1 - 'Describe the purpose of a truth table.' AO2 - 'Show how a binary search will be used…' AO3 - 'Draw a logic circuit for P = NOT A AND (B OR C)'

			user enters a number less than 0. Write an algorithm for this program'			
Command Words/Exam Technique Covered	Identify List Define State Tick Show Complete	Convert Compare Describe Explain	Draw Show Describe State Identify Complete	Complete Discuss Describe Explain Give Identify	Complete Discuss Describe Show Tick Explain	Draw Show Order Write Design Give Write
Example Exam Question Explored In This Term	*"Complete the sentences by filling in the missing words"*	*"Convert the denary number 221 into 8 bit binary"*	*"Describe the benefits of the student changing their home LAN to include wireless connectivity"*	*"Identify three ways Xander can make use of the file management facility"*	*"Discuss the ethical, legal and cultural impacts of this decision"*	*"Draw the logic diagram represented by Q=A OR NOT B"*
Key Words	• CPU • Fetch–Decode–Execute Cycle • Register • Clock Speed • Core • Embedded System • RAM • ROM • Solid State Drive (SSD) • Iteration	• Bit • Byte • Metadata • ASCII • Sampling • Lossless • Lossy • Topology • Switch • Router • Bandwidth	• LAN • WAN • Topology • Bandwith • Protocol • POP • MAC • Firewall • Transmission • interception	• Multitasking • Encryption • Defragmentation • Backup • Copyright • Data Protection • Sustainability • Subprogram • File handling • Authentication	• Privacy • E-waste • Digital divide • Copyright • Surveillance • Linear search • Binary search • Bubble sort • Merge sort • Efficiency	• Decomposition • Abstraction • Algorithm • Subprogram • Procedure • Validation • Authentication • Array • Boolean • Truth table

Term	1	2	3	4
Topic	2.1.1 Computational thinking 2.1.2 Designing, creating and refining algorithms 2.1.3 Searching and sorting algorithms	2.3.1 Defensive design 2.3.2 Testing 2.4.1 Boolean logic 2.5.1 Languages 2.5.2 The Integrated Development Environment (IDE)	2.2.3 Additional programming techniques	2.2.3 Additional programming techniques Exam Preparation and Recap. 1.1.1 Architecture of the CPU 1.2.4 Data Storage 1.3.1 Networks and topologies 1.6 Ethical, Environmental, Cultural and Legal Issues
Enquiry Question	"How can we use computational thinking to design efficient algorithms for solving real-world problems?"	"How do defensive design principles, testing methods, and programming tools like Boolean logic and IDEs improve the development of reliable and efficient software?"	"How do additional programming techniques like subprograms, file handling, and data validation enhance the functionality and reliability of software?"	"How do the design of computer systems and the way data is stored, shared, and used impact society and technology today?"
Big Ideas/ Key Concepts	• Clear Thinking Solves Complex Problems • Good Algorithms Save Time and Resources • Algorithms Must Be Designed, Not Just Written • Not All Algorithms Are Equal • Thinking Like a Computer Scientist Builds Better Solutions	• Reliability and Security in Software • Ensuring Software Works as Intended • Decision-Making Through Logic • Communication Between Humans and Machines • Tools That Support Effective Software Development	• Subprograms – Understanding and using functions and procedures to structure code for clarity and reuse. • File Handling – Reading from and writing to external files • Data Validation and Authentication – Implementing input validation techniques to ensure data integrity.	• The CPU as The Brain of the Computer • Representing and Storing Information Digitally • Connecting Computers for Communication and Sharing • Responsible Computing in a Connected World

			• Random Number Generation – Using random functions in programs • Using Arrays (or Lists) – Creating, accessing, updating, and iterating through arrays or lists to store collections of data efficiently.	
Key Knowledge and Skills	• Apply computational thinking techniques • Design and represent algorithms clearly • Analyse and refine algorithms for accuracy and efficiency • Understand and compare common searching and sorting algorithms • Select appropriate algorithms based on context	• Apply defensive programming techniques • Understand and carry out different types of testing • Use Boolean logic to control program flow • Compare and evaluate programming languages and translators • Use features of an IDE to develop and debug code	• Design and use subprograms to structure and simplify code • Read from and write to external files using programming techniques	• Identify and describe key components of the CPU • Explain how different types of data (text, images, sound) are stored in binarY • Compare different network topologies and their advantages/disadvantages • Evaluate the ethical and legal implications of using computer systems • Apply knowledge to real-world scenarios
End Point	Students will be able to think like computer scientists—breaking down problems, designing and refining efficient algorithms, and applying suitable searching and sorting techniques to solve real-world computational problems effectively.	These topics help students succeed in both practical coding questions and theory-based exam papers. They build essential problem-solving skills and confidence in understanding how software is developed, tested, and run in real-world environments.	This topic prepares students for high-mark coding tasks in assessments and builds strong foundations for further programming study at A Level or in real-world applications.	Studying these topics equips students with a solid understanding of how computer systems work, how data is stored and shared, and the wider impact of technology on society. It prepares them for GCSE exams by building core knowledge, critical thinking, and the ability to apply concepts to real-world and exam scenarios.
Prior Knowledge	Year 7 - Computational Thinking Year 8 - Search and Sort Algorithms Year 7,8 and 9 - Programming Fundamentals	Year 7 - Boolean Logic Year 9 - AR and Software Testing	Year 9 - The use of 2D Arrays and Functions	Year 10
Assessment Objectives	AO1 - Demonstrate knowledge and understanding of the key concepts and principles of Computer Science. 30%			

PONTEFRACT ACADEMIES TRUST

	AO2 - Apply knowledge and understanding of the key concepts and principles of Computer Science. 40% AO3 - Analyse problems in computational terms to make reasoned judgments and to design, program, evaluate and refine solutions. 30%			
Assessment Objectives Covered	AO1 AO2 AO3	AO1 AO2 AO3	AO2 AO3	AO1 AO2 AO3
Assessment Objectives Covered with Examples	AO1 - "State the name of each of the following computational thinking techniques" AO2 - "Complete the merge sort of the data by showing each step of the process" AO3 - "...Design the algorithm using a flowchart."	AO1 - "Complete the description of programming languages and translators by writing the correct term from the box in each space" AO2 - "Complete the following logic diagram for P = (A OR B) AND NOT C by drawing one logic gate in each box" AO3 - "Complete the following test plan to check whether the number of nights is validated correctly"	AO2 - "Give two ways that the maintainability of this program could be improved." AO3 - "A Basic room costs £60 each night. A Premium room costs £80 each night. Create a function, newPrice(), that takes the number of nights and the type of room as parameters, calculates and returns the price to pay. You do not have to validate these parameters."	AO1 - 'Identify two events that take place during the fetch-execute cycle' AO2 - 'Complete the table by writing the missing denary, 8-bit binary or hexadecimal values.' AO3 - 'Discuss the features, benefits and drawbacks of each type of licence'
Command Words/Exam Technique Covered	Describe Explain Compare Design Evaluate	Explain Describe Identify Compare Evaluate	Explain Describe Design Implement Evaluate Identify Compare	List Describe Discuss Explain
Example Exam Question Explored In This Term	*"State, using the process in Fig. 2, the username for Rebecca Ellis."*	*"Tick one box to identify the correct logic diagram for P = NOT(A AND B)."*	*"Write an algorithm for the updated program design shown in question"*	*"Discuss the ethical, legal and environmental impacts of this decision"*
Key Words	• Decomposition • Abstraction	• Validation • Authentication	• Subprogram • Function	• CPU • Fetch-Decode-Execute

	• Algorithm • Pseudocode • Flowchart • Linear search • Binary search • Bubble sort • Merge sort • Efficiency	• Maintainability • Syntax error • Logic error • Boolean • AND • Compiler • Interpreter • Debugger	• Call • Procedure • File handling • Data validation • Authentication • Random number generation • Array • List	• Cache • Binary • Bandwidth • Topology • Router • Encryption • Sustainability • Intellectual property

Year 11 2026-2027

Term	**1**	**2**	**3**	**4**
Topic	2.1.1 Computational thinking	2.3.1 Defensive design 2.3.2 Testing	2.2.3 Additional programming techniques	2.2.3 Additional programming techniques Exam Preparation and Recap.

	2.1.2 Designing, creating and refining algorithms 2.1.3 Searching and sorting algorithms	2.4.1 Boolean logic 2.5.1 Languages 2.5.2 The Integrated Development Environment (IDE)		1.1.1 Architecture of the CPU 1.2.4 Data Storage 1.3.1 Networks and topologies 1.6 Ethical, Environmental, Cultural and Legal Issues
Enquiry Question	"How can we use computational thinking to design efficient algorithms for solving real-world problems?"	"How do defensive design principles, testing methods, and programming tools like Boolean logic and IDEs improve the development of reliable and efficient software?"	"How do additional programming techniques like subprograms, file handling, and data validation enhance the functionality and reliability of software?"	"How do the design of computer systems and the way data is stored, shared, and used impact society and technology today?"
Big Ideas/ Key Concepts	• Clear Thinking Solves Complex Problems • Good Algorithms Save Time and Resources • Algorithms Must Be Designed, Not Just Written • Not All Algorithms Are Equal • Thinking Like a Computer Scientist Builds Better Solutions	• Reliability and Security in Software • Ensuring Software Works as Intended • Decision-Making Through Logic • Communication Between Humans and Machines • Tools That Support Effective Software Development	• Subprograms – Understanding and using functions and procedures to structure code for clarity and reuse. • File Handling – Reading from and writing to external files • Data Validation and Authentication – Implementing input validation techniques to ensure data integrity. • Random Number Generation – Using random functions in programs	• The CPU as The Brain of the Computer • Representing and Storing Information Digitally • Connecting Computers for Communication and Sharing • Responsible Computing in a Connected World

			• Using Arrays (or Lists) – Creating, accessing, updating, and iterating through arrays or lists to store collections of data efficiently.	
Key Knowledge and Skills	• Apply computational thinking techniques • Design and represent algorithms clearly • Analyse and refine algorithms for accuracy and efficiency • Understand and compare common searching and sorting algorithms • Select appropriate algorithms based on context	• Apply defensive programming techniques • Understand and carry out different types of testing • Use Boolean logic to control program flow • Compare and evaluate programming languages and translators • Use features of an IDE to develop and debug code	• Design and use subprograms to structure and simplify code • Read from and write to external files using programming techniques	• Identify and describe key components of the CPU • Explain how different types of data (text, images, sound) are stored in binarY • Compare different network topologies and their advantages/disadvantages • Evaluate the ethical and legal implications of using computer systems • Apply knowledge to real-world scenarios

Focus during the additional hour of learning and rationale.	1.1.1 Architecture of the CPU 1.2.4 Data Storage: Numbers These two topics (along with 1.3.1 and 1.3.2 [see Term 3 additional hours]) can take up (on average) 40 marks of an 80 mark paper. The additional hour will be spent recapping these major topics and also exploring them through the lenses of past paper exam material.	2.2 Programming Fundamentals Why this topic? Students find this section of paper 2 extremely challenging and difficult. Why it's challenging: This section requires actual code understanding and problem-solving ability — not just theory. Specific areas to be practically explored include: Variables, data types, inputs/outputs Arithmetic & Boolean operations Selection (IF statements), iteration (loops) String manipulation and file handling Subroutines (functions/procedures)	1.3.1 Networks and topologies 1.3.2 Wired and wireless networks, protocols and layers	1.5.1 OS 1.5.2 Utilities These topics (although quite rich and vast in content) have rarely made an appearance in past papers. When they have, they have been very low-level, AO1 style 'fill in the blanks' style questions. Even then, students still struggle with these two topics. The additional hours of this term would be spent exploring the tier 3 vocabulary associated with the topics and applying them to exam style questions.
End Point	Students will be able to think like computer scientists—breaking down problems, designing and refining efficient algorithms, and applying suitable searching and	These topics help students succeed in both practical coding questions and theory-based exam papers. They build essential problem-solving skills	This topic prepares students for high-mark coding tasks in assessments and builds strong foundations for further	Studying these topics equips students with a solid understanding of how computer systems work, how data is stored and shared, and the wider impact of technology on society. It prepares them for GCSE exams by building

	sorting techniques to solve real-world computational problems effectively.	and confidence in understanding how software is developed, tested, and run in real-world environments.	programming study at A Level or in real-world applications.	core knowledge, critical thinking, and the ability to apply concepts to real-world and exam scenarios.
Prior Knowledge	Year 7 - Computational Thinking Year 8 - Search and Sort Algorithms Year 7, 8, 9 Computer Programming	Year 7 - Boolean Logic Year 9 - AR and Software Testing Years 8 and 9 - Python Programming	Year 7, 8 and 9 - Programming	Year 7, 8 and 9 - Programming Year 8 - Ethics Year 7 - Data Storage Year 8 - The role of the CPU Year 9 - Networks
Assessment Objectives	AO1 - Demonstrate knowledge and understanding of the key concepts and principles of Computer Science. 30% AO2 - Apply knowledge and understanding of the key concepts and principles of Computer Science. 40% AO3 - Analyse problems in computational terms to make reasoned judgments and to design, program, evaluate and refine solutions. 30%			
Assessment Objectives Covered	AO1 AO2 AO3	AO1 AO2 AO3	AO2 AO3	AO1 AO2 AO3
Assessment Objectives Covered with Examples	AO1 - “State the name of each of the following computational thinking techniques” AO2 - “Complete the merge sort of the data by showing each step of the process” AO3 - “...Design the algorithm using a flowchart.”	AO1 - “Complete the description of programming languages and translators by writing the correct term from the box in each space” AO2 - “Complete the following logic diagram for P = (A OR B) AND NOT C by drawing one logic gate in each box” AO3 - “Complete the following test plan to check	AO2 - “Give two ways that the maintainability of this program could be improved.” AO3 - “A Basic room costs £60 each night. A Premium room costs £80 each night. Create a function, newPrice(), that takes the number of nights and the type of room as parameters, calculates and returns the price to pay.	AO1 - “Identify two events that take place during the fetch-execute cycle.” AO2 - “Convert the binary number 11001011 into denary.” AO3 - “A second program needs to perform the following tasks: • Input a number from the user • Double the number input and print the result • Repeat bullets 1 and 2 until the user enters a number less than 0. Write an algorithm for this program.”

		whether the number of nights is validated correctly"	You do not have to validate these parameters."	
Command Words/Exam Technique Covered	Describe Explain Compare Design Evaluate	Explain Describe Identify Compare Evaluate	Explain Describe Design Implement Evaluate Identify Compare	State List Draw Describe Explain Discuss Write
Example Exam Question Explored In This Term	*"Give two computational thinking techniques that Taylor has used, describing how they have been used."*	*"Draw the logic diagram for the logic system P=A OR (B AND C)*	*"Write an algorithm that will identify whether the data in the studentdata array shows that a letter has been sent home or not for the student. The algorithm should then output either "sent" (if a letter has been sent) or "not sent" (if a letter has not been sent)."*	*"Describe the difference between a LAN and a WAN"*
Key Words	• Decomposition • Abstraction • Algorithm • Pseudocode • Flowchart • Linear search • Binary search • Bubble sort • Merge sort • Efficiency	• Validation • Authentication • Maintainability • Syntax error • Logic error • Boolean • AND • Compiler • Interpreter • Debugger	• Subprogram • Function • Call • Procedure • File handling • Data validation • Authentication • Random number generation • Array • List	• CPU • Fetch-Decode-Execute • Cache • Binary • Bandwidth • Topology • Router • Encryption • Sustainability • Intellectual property