Thomas Mills
High School & Sixth Form

CURRICULUM OVERVIEW: COMPUTER SCIENCE

Our Computer Science curriculum at Thomas Mills High School & Sixth Form aims to provide pupils with a broad understanding and depth of knowledge in computing, building directly upon the foundational skills and concepts introduced in Key Stage 3. We are committed to fostering computational thinking, problem-solving abilities and a deep understanding of how technology works, ensuring that pupils are not only informed users of technology but also creative and responsible digital citizens.

Through our curriculum, pupils will develop a strong understanding of various computing principles, from the representation of algorithms to the intricacies of computer networks and cybersecurity. We encourage analysis, evaluation and discussion throughout the schemes, providing opportunities for pupils to reflect on their own views and the impact of technology on society. There is a strong emphasis on respect, diversity of view and feeling empowered to share ideas and ask questions.

Our goal is to prepare pupils for their next steps, whether that be further academic study in Computer Science or related fields, apprenticeships, or future careers in the ever-evolving digital landscape. We equip them with the essential knowledge and practical skills to navigate and contribute positively to the digital world, empowering them to be confident, capable and ethical participants in a technology-driven society.

Qualification: AQA GCSE Computer Science 8525

Thomas Mills
High School & Sixth Form

CURRICULUM OVERVIEW: COMPUTER SCIENCE

Year 9

Term	Topic	Knowledge and Skills	Useful Links
1	**Unit 1A:** Representing Algorithms	In this topic, your child will begin their journey into the exciting world of algorithms, which are essentially step-by-step instructions that computers follow to solve problems. They will learn how to represent these instructions in different ways. **Knowledge:** • The difference between algorithms and computer programs. • The concept of computational thinking techniques including abstraction and decomposition. • How to define and identify algorithms represented as written descriptions, flowcharts and pseudocode. • The meaning and use of standard flowchart symbols (start, end, input, output, subroutine, decision). • The concepts of basic programming constructs: sequence (instructions in order), selection (making decisions with 'if/else') and iteration (repeating instructions with loops). • Different data types used in programming: integer (whole numbers), real/float (numbers with decimals), Boolean (true/false), character (single letter/symbol) and string (text).	**How you can help at home:** • Encourage your child to think about everyday tasks as a series of steps. For example, writing down the steps to make a cup of tea or getting ready for school. • You could also try creating simple flowcharts together for these activities. • Play board games that involve strategic thinking or following a set of rules, like chess or even simple card games, to reinforce algorithmic thinking. • Discuss how everyday apps or devices (e.g. - traffic lights, vending machines) follow a sequence of instructions. **AQA Specification:** • 3.1.1 • AQA | Computer Science | GCSE | GCSE Computer Science **Links:** • BBC Bitesize: Fundamentals of algorithms - AQA • YouTube: Flowcharts (A simple introduction to flowcharts)

		• How Boolean operators (AND, OR) are used with conditions. **Skills:** • Analysing and creating flowcharts. • Translating a flowchart into a program sequence. • Designing a flowchart for a program. • Interpreting, correcting, completing and refining algorithms using flowcharts. • Applying decomposition and abstraction to problem-solving. • Defining and using various data types. • Writing algorithms in pseudocode involving sequence, selection and iteration.	• Ada Computer Science: Computational thinking • Craig n Dave: SLR07 - Algorithms (Look for specific videos on flowcharts, pseudocode, decomposition, abstraction and programming constructs) • PG Online Textbook: Chapter 1: 1.1 to 1.3 (See pages related to algorithms, flowcharts, pseudocode) • Seneca Learning: Computer Science
Assessments	Your child will have a mid-point multiple-choice quiz and a summative assessment (made up of past paper examination questions) to evaluate their understanding of representing algorithms. Past paper questions will be a mix of 'shade the lozenge', short answer and longer answer coding-based questions.		

Thomas Mills
High School & Sixth Form

CURRICULUM OVERVIEW: COMPUTER SCIENCE

1	**Unit 5:** Networks	This unit introduces the fundamental concepts of computer networks, explaining how devices connect and communicate with each other. **Knowledge:** • The definition of a computer network. • The advantages and disadvantages of computer networks. • The purpose and characteristics of Personal Area Networks (PAN), Local Area Networks (LAN) and Wide Area Networks (WAN). • The attributes of fibre optic and copper cables used in wired networks. • Bluetooth as a mode of connection. • The advantages and disadvantages of wireless networks compared to wired networks. • The need for standards and protocols in network communications. • The purpose and common use of network protocols: Ethernet, WiFi, HTTP, HTTPS, FTP, POP, SMTP and IMAP. • The four layers of the TCP/IP model (Link, Internet, Transport, Application). • Which protocols operate at each layer of the TCP/IP model (e.g., HTTP at application layer, TCP at transport layer, IP at internet layer). • Different types of network security measures: encryption, authentication, firewall and MAC address filtering.	**How you can help at home:** • Discuss your home network with your child. Explain what your router does, how your devices connect and the importance of strong Wi-Fi passwords. • Talk about the difference between wired and wireless connections in your home and why you might choose one over the other for certain devices. • Discuss the concept of "the cloud" and how data is stored and accessed over networks. **AQA Specification:** • 3.5 • AQA | Computer Science | GCSE | GCSE Computer Science **Links:** • BBC Bitesize: Fundamentals of computer networks - AQA • BBC Bitesize: Network topologies, protocols and layers - AQA • Ada Computer Science: Network fundamentals • Ada Computer Science: Network security • Ada Computer Science: Encryption • Craig n Dave: SLR03 - Computer networks, protocols and layers • PG Online Textbook: Chapter 5: 5.1 to 5.4 • Seneca Learning: Computer Science

		Skills: • Discussing the advantages and disadvantages of computer networks. • Drawing and describing star and bus network topologies. • Determining the need for standards in network communications. • Defining and explaining the purpose of various network protocols. • Describing the layers of the TCP/IP model. • Describing and applying different types of security measures.	
Assessments		Your child will have a mid-point multiple-choice quiz and a summative assessment (made up of past paper examination questions) to evaluate their understanding of networks. Past paper questions will be a mix of 'shade the lozenge', short answer and longer answer written questions.	

Thomas Mills
High School & Sixth Form

CURRICULUM OVERVIEW: COMPUTER SCIENCE

1	**Unit 2A:** Learn how to write structured programs	This unit introduces your child to the basics of writing computer programs, primarily using Python. They will learn about the building blocks of programs and how to make them organised and readable. **Knowledge:** • The concept of a program and how it is structured into smaller parts. • What subroutines (functions and procedures) are and their role in structuring code. • The difference between procedures (no return value) and functions (return values). • The purpose of comments in code for explanation and readability. • The importance of indentation for code structure and readability. • The role of keywords in programming languages (e.g., print, if, else) and why they are reserved words. • How text is represented as strings in programming languages, enclosed in quotes. • The difference between constants (fixed values) and variables (data storage that can change). • How identifiers are used to name variables and constants. • The role of parameters in functions and how parameter passing works. • The distinction between arguments (actual data passed) and parameters (placeholders in function definitions).	**How you can help at home:** • Encourage your child to try out simple Python programs using online interpreters such as Edublocks. • Ask them to explain what different parts of a simple program do. • Look for beginner-friendly coding challenges online that focus on basic programming concepts. • Discuss how breaking down complex tasks into smaller functions makes them easier to manage. **AQA Specification:** • 3.2.2 and 3.2.10 • AQA | Computer Science | GCSE | GCSE Computer Science **Links:** • BBC Bitesize: Programming concepts - AQA • BBC Bitesize: Programming languages - AQA • Ada Computer Science: Programming concepts • Ada Computer Science: Subroutines • W3Schools Python Tutorial (A good beginner-friendly Python resource) • Craig n Dave: SLR08 - Basic programming concepts • PG Online Textbook: Chapter 2: 2A.1 to 2A.2 and 2B.1

		• The concept of sequence in programming, where instructions execute one after another. • The term "debugging" and common strategies for finding and fixing bugs, particularly syntax errors. **Skills:** • Understanding and using subroutines (functions and procedures). • Adding comments to code for clarity. • Using keywords and understanding case sensitivity in Python. • Working with basic strings in programs. • Assigning values to and using variables and constants. • Creating functions that take parameters and return values. • Using variables as arguments when calling functions. • Identifying and fixing syntax errors. • Applying basic debugging strategies.	• Seneca Learning: Computer Science
Assessment		Your child will have a mid-point multiple-choice quiz to evaluate their understanding of structured programs.	

2	**Unit 6:** Cyber Security	This unit focuses on understanding the threats to computer systems and networks and how to protect against them. **Knowledge:** • The impact of cybercrime on businesses and individuals. • What motivates hackers in cyberattacks. • The definitions of cybersecurity and network security, their importance and the distinction between them. • Non-automated forms of cyberattacks (social engineering) and how humans can be weak links in the security chain. • Various cyber security threats: weak and default passwords, misconfigured access rights, removable media risks and unpatched/outdated software. • Different forms of malware: computer virus, Trojan and spyware. • How malware can be protected against. • Different security measures: biometric measures, password systems, CAPTCHA, email confirmation for user identity and automatic software updates. • What penetration testing is and what it is used for. **Skills:**	**How you can help at home:** • Discuss online safety with your child. Emphasize the importance of strong, unique passwords, being cautious about suspicious emails or links and keeping software updated. • Talk about recent cybersecurity news stories or data breaches and their impact. • Encourage them to think critically about information they encounter online and the potential risks of sharing personal data. **AQA Specification:** • 3.6.1 – 3.6.3 • AQA | Computer Science | GCSE | GCSE Computer Science **Links:** • BBC Bitesize: Fundamentals of cyber security - AQA • Ada Computer Science: Social engineering • Ada Computer Science: Malicious software • Ada Computer Science: Network security • Craig n Dave: SLR04 - Cyber security • PG Online Textbook: Chapter 6: 6.1 to 6.4 • Seneca Learning: Computer Science

		• Analysing an attack on a company and identifying hacker motivations. • Demonstrating knowledge of social engineering through role-playing and case studies. • Understanding and explaining various cyber security threats. • Defining and describing different forms of malware. • Describing how to protect against malware. • Understanding and explaining various security measures. • Explaining the purpose of penetration testing.	
Assessments		Your child will have a mid-point multiple-choice quiz and a summative assessment (made up of past paper examination questions) to evaluate their understanding of cyber security. Past paper questions will be a mix of 'shade the lozenge', short answer and longer answer written questions.	

2	**Unit 2B:** Learn how to use selection	This unit delves deeper into how programs make decisions, building on the selection concepts introduced with pseudocode (Unit 1A). Your child will learn to use selection statements effectively in Python. **Knowledge:** • How to evaluate arithmetic expressions using rules of operator precedence (BIDMAS). • Different arithmetic operators (addition, subtraction, multiplication, real division, integer division, modulo, to the power). • The concept of a condition as an expression that evaluates to either True or False. • How selection uses conditions to control the flow of execution in a program. • The structure of selection statements (if, elif, else) in Python. • Various comparison operators (equal to, not equal to, less than, greater than, less than or equal to, greater than or equal to). • How Boolean/logical operators (AND, OR) can be used to combine conditions. • The concept of nested selection (selection statements within other selection statements). **Skills:** • Writing and using expressions that use arithmetic operators. • Assigning expressions to variables. • Identifying flowchart symbols for decisions.	**How you can help at home:** • Encourage your child to try out simple selection Python programs using online interpreters such as Edublocks. • Discuss scenarios where decisions need to be made based on certain conditions (e.g., "If it's raining, take an umbrella, OR if it's cold, wear a coat"). This helps reinforce the idea of conditional logic. • Play simple online coding games that involve choosing different paths based on conditions. • Encourage them to think about how selection is used in everyday technology, such as choosing options in a menu or responding to user input. **AQA Specification:** • 3.2.2 – 3.2.5 • AQA | Computer Science | GCSE | GCSE Computer Science **Links:** • BBC Bitesize: Programming concepts - AQA • BBC Bitesize: Programming languages - AQA • Ada Computer Science: Programming concepts • W3Schools Python Tutorial (A good beginner-friendly Python resource)

		• Walking through and interpreting code that includes selection statements. • Using selection statements in programs. • Identifying when selection statements should be used. • Writing and using expressions that use comparison operators. • Writing and using expressions that use Boolean/logical operators (AND, OR). • Walking through code that uses nested selection. • Modifying programs that use nested selection. • Writing algorithms in Python involving selection.	• Craig n Dave: SLR08 - Basic programming concepts • PG Online Textbook: Chapter 2: 2A.1 to 2A.2 • Seneca Learning: Computer Science
Assessment		Your child will have a mid-point multiple-choice quiz and a summative assessment (made up of past paper examination questions) to evaluate their understanding of structured programs (Unit 2A) and selection (Unit 2B). Past paper questions will be a mix of 'shade the lozenge', short answer and longer answer coding-based questions.	

3	**Unit 2C:** Learn how to use number data types	This unit focuses on working with different types of numbers in Python and introduces the concept of randomness. **Knowledge:** • How to locate information using language documentation (e.g., Python's official documentation). • The concept of importing modules into code to extend functionality. • How to generate random numbers in a programming language. • The definition and differentiation between int (integers) and float (real numbers) data types in Python. • The concept of "casting" as data type conversion. • How float() converts data to real numbers and int() converts data to integers (understanding that int() truncates, it does not round). • Potential errors that can occur when casting strings to integers. **Skills:** • Locating and using information from programming language documentation. • Importing modules into their code. • Demonstrating how to generate random numbers.	**How you can help at home:** • Encourage your child to try out simple number-based Python programs using online interpreters such as Edublocks. • Explore simple online games that use random numbers (e.g. - dice rolls) and discuss how the computer might be generating those numbers. • Discuss real-world examples of how whole numbers (integers) and decimal numbers (floats) are used differently (e.g., counting discrete items vs. measurements). • Try a simple project together, like a "guess the number" game, where random numbers are used. **AQA Specification:** • 3.2.1 and 3.2.9 • AQA | Computer Science | GCSE | GCSE Computer Science **Links:** • BBC Bitesize: Further programming language operations - AQA • Ada Computer Science: Random number generation • W3Schools Python Tutorial (A good beginner-friendly Python resource) • Craig n Dave: SLR09 - Advanced programming concepts

		• Comparing and contrasting randint() and randrange() for random number generation. • Converting data to float and int using float() and int(). • Writing algorithms in Python involving number data types.	• PG Online Textbook: Chapter 2: 2A.3 • Seneca Learning: Computer Science
Assessment		Your child will have a mid-point multiple-choice quiz to evaluate their understanding of using number data types. Additionally, your child will have a summative assessment assessing all content taught to date, focusing on specification points 3.1, 3.2, 3.5 and 3.6. Past paper questions will be a mix of 'shade the lozenge', short answer and longer answer coding/written questions. This should be read in conjunction with the AQA subject content within the specification.	

3	**Unit 2D:** Learn how to use strings	This unit introduces your child to working with text in computer programs, known as "strings." They will learn how to manipulate text, combine it, and extract information from it. **Knowledge:** • What strings are in programming (sequences of characters, or text). • How to use basic operations like combining text (concatenation), finding the length of text, and extracting parts of text (substrings). • How characters are represented numerically (ASCII) and how to convert between characters and their ASCII values. • How to check if a piece of text contains another specific piece of text. • How to use loops to process each character or section of a string. • The specific commands and rules for performing string operations in Python. **Skills:** • Using various string handling techniques (concatenation, substrings, chr(), ord(), in operator) in practical programming scenarios. • Designing step-by-step instructions (algorithms) to solve problems that involve manipulating text. • Writing functional Python code that effectively uses string handling techniques to achieve desired outcomes.	**How you can help at home:** • Encourage your child to try out simple string-based Python programs using online interpreters such as Edublocks. • Point out how text manipulation is used in daily life (e.g., search engines, social media, online forms, games, word processors) to make learning more relevant. • Encourage hands-on coding practice by trying out lesson examples or simple text-based programming challenges (e.g. - reversing words, counting characters). **AQA Specification:** • 3.2.1 and 3.2.8 • AQA | Computer Science | GCSE | GCSE Computer Science **Links:** • BBC Bitesize: Further programming language operations - AQA • Ada Computer Science: String handling • W3Schools Python Tutorial (A good beginner-friendly Python resource) • Craig n Dave: SLR09 - Advanced programming concepts • PG Online Textbook: Chapter 2: 2A.1 to 2A.2 • Seneca Learning: Computer Science

		• Breaking down text-based problems into smaller, manageable steps that can be solved with programming. • Evaluating and refining algorithms for efficiency and correctness when dealing with text data.	
Assessment		Your child will have a mid-point multiple-choice quiz and a summative assessment (made up of past paper examination questions) to evaluate their understanding of number data types (Unit 2C) and the string data type (Unit 2D). Past paper questions will be a mix of 'shade the lozenge', short answer and longer answer coding-based questions.	