Year 10		Curriculum Checkpoints: What do students know and what can they do?			
Computer Science		Developing	Securing	Flourishing	Excelling
Exam 1					
1.1 Systems Architecture	1.1.1 Architecture of the CPU	I can identify the main parts of the CPU and know it carries out instructions.	I can describe what happens in the fetch–execute cycle and name the key components such as the ALU, CU, cache, and registers.	I can explain what each CPU component and register does and how they work together in the fetch–execute cycle.	I can confidently explain how the CPU processes instructions step-by-step, including the role of registers and the Von Neumann architecture.
	1.1.2 CPU performance	I know that factors like clock speed, cache size and number of cores affect how fast the CPU works.	I can describe how changing clock speed, cache size or cores can make a CPU faster or slower.	I can explain clearly how these factors impact CPU performance and give examples of when each is most important.	I can evaluate how combinations of CPU characteristics affect performance in different situations and justify which would be most effective.
	1.1.3 Embedded systems	I can recognise that some devices have built-in computers called embedded systems.	I can describe what an embedded system is and give simple examples (e.g. washing machine, sat-nav).	I can explain the main characteristics of embedded systems and how they are used for specific tasks.	I can compare embedded and general-purpose systems and explain the advantages of using embedded systems in real products.
R094 NEA Visual Identity and Digital Graphics	1.2.1 Primary storage (Memory)	I can recognise that computers use memory to store data and instructions.	I can describe the difference between RAM and ROM and explain their basic purposes.	I can explain how RAM, ROM, and cache work together and why virtual memory is needed.	I can explain how different types of memory interact to improve performance, giving clear examples.
	1.2.2 Secondary storage	I can identify that computers use storage devices to save data.	I can describe common types of storage such as optical, magnetic, and solid-state.	I can explain the advantages and disadvantages of each storage type and suggest suitable devices for different uses.	I can evaluate which storage device is most suitable for a scenario, using characteristics such as capacity, speed, and reliability.
	1.2.3 Units	I can recognise the basic units of data (bit, byte, kilobyte, etc.).	I can convert between units like KB, MB, and GB, and understand that data must be stored in binary.	I can calculate file sizes for text, images, and sound using formulas.	I can confidently calculate and compare storage requirements across file types, explaining how characteristics like bit depth and resolution affect size.
	1.2.4 Data storage	I can convert small num I can recognise that sound can be stored digitally.bers between denary and binary. I can recognise that computers use codes to represent letters and symbols. I can recognise that images are made up of pixels.	I can convert between binary, denary, and hexadecimal numbers and carry out simple binary addition. I can describe how ASCII and Unicode are used to represent text on computers. I can describe how colour depth and resolution affect image quality. I can describe how sound is sampled and the meaning of sample rate and bit depth.	I can explain overflow errors, use binary shifts, and convert numbers confidently between systems. I can explain how the number of bits used affects the range of characters that can be represented. I can explain how colour depth and resolution affect both image quality and file size. I can explain how sample rate, duration, and bit depth affect sound quality and file size.	I can apply binary, hexadecimal, and shift knowledge to real exam-style problems and explain their impact on data representation. I can compare ASCII and Unicode, explaining why Unicode is needed for global communication. I can analyse how changing colour depth or resolution impacts storage and quality in different contexts. I can evaluate trade-offs between sound quality and file size for different digital audio formats.
	1.2.5 Compression	I can understand that compression reduces file size.	I can describe the difference between lossy and lossless compression.	I can explain how lossy and lossless compression work and when each would be used.	I can evaluate the impact of using lossy or lossless compression in different scenarios (e.g., music vs text files).

R094 NEA Visual Identity and Digital Graphics	1.3.1 Networks and topologies	I can recognise that computers can be connected together to share information. I can recognise that the Internet connects networks together.	I can describe what a LAN and WAN are and name basic network hardware such as routers and switches. I can describe what the Internet, DNS, and the Cloud are used for.	I can explain how hardware, bandwidth and number of devices affect network performance, and describe client-server and peer-to-peer setups. I can explain how DNS converts web addresses to IP addresses and how the Cloud provides online services.	I can compare different network types, hardware and topologies, evaluating which setup would be most effective for different scenarios. I can evaluate advantages and disadvantages of DNS, the Cloud, and hosting options when designing or maintaining a network.
	1.3.2 Wired and wireless networks, protocols and layers	I can identify examples of wired and wireless connections. I can recognise that data needs to be kept safe when sent across a network. I can recognise that computers use rules to communicate.	I can describe the differences between Ethernet, Wi-Fi, and Bluetooth connections and their basic uses. I can describe what encryption is and why IP and MAC addresses are important. I can describe what a protocol and a standard are, and name a few examples. I can explain how common protocols (such as HTTP, HTTPS, FTP, and SMTP) are used to transfer data across networks.	I can explain advantages and disadvantages of wired and wireless connections and recommend suitable options for given needs. I can explain how encryption secures data and how IP and MAC addressing work within networks. I can explain how encryption secures data and how IP and MAC addressing work within networks. I can explain how common protocols (such as HTTP, HTTPS, FTP, and SMTP) are used to transfer data across networks.	I can evaluate and justify connection choices for real-world situations, considering speed, reliability, cost, and security. I can compare IPv4 and IPv6 formats and evaluate how encryption and addressing maintain privacy and security. I can explain how the TCP/IP model organises these protocols into layers and analyse the benefits of using a layered approach.
1.4 Network Security	1.4.1 Threats to computer systems and networks	I can recognise that computers and networks can be attacked in different ways.	I can describe common types of attacks such as malware, phishing, and brute-force attacks.	I can explain how different attacks work and what the attacker is trying to achieve, such as stealing data or disrupting services.	I can analyse and compare different types of cyber attacks, explaining their methods, impacts, and how they exploit vulnerabilities.
	1.4.2 Identifying and preventing vulnerabilities	I can recognise that there are ways to protect computer systems from attacks.	I can describe basic prevention methods such as firewalls, passwords, and anti-malware software.	I can explain how a range of prevention methods (like penetration testing, encryption, and physical security) help reduce risks.	I can evaluate which prevention methods are most effective for different situations and explain how layers of protection work together to keep systems secure.
1.5 System Software	1.5.1 Operating systems	I can recognise that a computer needs an operating system to work.	I can describe the main functions of an operating system, such as providing a user interface and managing files and memory.	I can explain how the operating system manages memory, hardware, users, and files to allow multitasking and secure access.	I can explain in detail how an operating system coordinates resources between hardware and software, and evaluate how different features improve performance and usability.
	1.5.2 Utility software	I can recognise that computers use utility software to help with maintenance tasks.	I can describe examples of utility software such as encryption, defragmentation, and compression.	I can explain how each type of utility software works and why it is important for system performance and security.	I can evaluate how utility software improves efficiency, security, and storage management, comparing which tools are most useful in different contexts.
1.6 Ethical, Legal, Cultural and Environmental Impacts of Digital Technolog	1.6.1 - Ethial, Legal, Cultural and Environmental Impact	I can recognise that technology can have positive and negative effects on people and society.	I can describe different types of issues such as ethical, legal, cultural, environmental, and privacy issues, and give simple examples.	I can explain how different technologies create these issues and how laws like the Data Protection Act, Computer Misuse Act, and Copyright Act help protect people.	I can evaluate the impact of digital technology on society, explain how legislation addresses these issues, and justify the best type of software licence (open source or proprietary) for a given scenario.

Exam 2

2.1 - Algorithms	2.1.1 - Computational Thinking	I can recognise that problems can be broken down into smaller parts.	I can describe the ideas of abstraction, decomposition, and algorithmic thinking.	I can explain how abstraction removes unnecessary detail, decomposition breaks down problems, and algorithmic thinking creates step-by-step solutions.	I can confidently apply abstraction, decomposition, and algorithmic thinking to design efficient solutions to complex problems.
	2.1.2 - Designing. Creating and Refining Algorithms	I can identify the inputs, processes, and outputs of a simple problem.	I can use flowcharts or pseudocode to design an algorithm that solves a problem.	I can refine algorithms using decomposition and selection/iteration, and identify and correct logic or syntax errors.	I can design clear, efficient, and well-structured algorithms using trace tables to test and refine solutions, explaining how each improvement makes the algorithm more effective.
	2.1.3 - Searching and Sorting Algorithms	I can recognise that computers use algorithms to search and sort data.	I can describe how basic searches (linear and binary) and sorts (bubble, insertion, and merge) work in simple terms.	I can explain the main steps of each searching and sorting algorithm and apply them to small data sets.	I can compare the efficiency and use of different searching and sorting algorithms, explaining which is most suitable in different situations.
2.2 - Programming Fundamentals	2.2.1 - Programming Fundamentals	I can recognise what variables, inputs, and outputs are in a program.	I can create simple programs that use variables, inputs, outputs, and basic arithmetic and Boolean operators.	I can use sequence, selection, and iteration effectively to control how a program runs.	I can design and write efficient programs that combine sequence, selection, and iteration to solve more complex problems.
	2.2.2 - Data Types	I can recognise that different types of data are used in programs.	I can use basic data types such as integers, real numbers, strings, and booleans in my code.	I can choose suitable data types for a task and use type casting when needed.	I can justify my choice of data types and apply them accurately in different programming contexts.
	2.2.3 - Additional Programming Techniques	I can recognise that programs can use extra features such as lists, strings, and files.	I can use basic string manipulation and simple arrays or lists in my programs.	I can use a range of programming techniques, including file handling, arrays, functions, and random numbers.	I can confidently combine advanced techniques (like 2D arrays, SQL, and subprograms) to create structured, reusable, and efficient code.
2.3 - Producing Robust Programs	2.3.1 - Defensive Design	I can recognise that programs need to deal with errors and prevent misuse.	I can add simple validation checks and use authentication like usernames and passwords.	I can design programs that include input validation, authentication, and clear commenting to make code easier to understand and maintain.	I can create well-structured, secure programs using subprograms, consistent naming, indentation, and commenting that make code easy to debug, maintain, and expand.
	2.3.2 - Testing	I can recognise that programs need testing to check they work correctly.	I can describe different types of testing (such as iterative and final) and identify syntax and logic errors.	I can use suitable test data (normal, boundary, and erroneous) to check if my program works correctly.	I can plan, carry out, and refine tests effectively, explaining how my chosen test data helps find and fix errors in complex programs.
2.4 - Boolean Logic	2.4.1 - Boolean Logic	I can recognise the AND, OR, and NOT logic gates.	I can draw and interpret simple logic gate diagrams and complete truth tables for one gate.	I can combine logic gates using AND, OR, and NOT to create and complete truth tables for more complex circuits.	I can confidently create and simplify multi-gate logic diagrams and truth tables, explaining how logical operations can be used to solve real-world problems.

2.5 - Programming Languages and Integrated Development Environments (IDEs)	2.5.1 - Languages	I can recognise that there are different types of programming languages.	I can describe the difference between high-level and low-level languages and explain why translators are needed.	I can explain the purpose of compilers and interpreters and describe their advantages and disadvantages.	I can evaluate when to use high-level or low-level languages and justify whether a compiler or interpreter would be more effective in a given situation.
	2.5.2 - The Integrated Development Environment (IDE)	I can recognise that programmers use software to help them write and test programs.	I can describe the main features of an IDE, such as editors, translators, and error checking.	I can explain how IDE tools such as error diagnostics, debugging, and run-time environments help programmers develop programs.	I can confidently use and evaluate IDE tools to write, test, and improve code efficiently, explaining how each tool supports the development process.