Programme of study for Year 11 Computer Science

Autumn (1st term) Topic	Autumn (2nd term) Topic	Spring (1st term) Topic	Spring (2nd Term) Topic	Summer (1st term) Topic	Summer (2nd term) Topic
Component 2: 2.5.2 - The integrated Development Environment. Common tools and facilities available in an IDE. 2.5.1 – Languages Characteristics and purpose of different levels of programming language. 2.1.2 Identify common errors and trace tables. **Revisit** 2.2.1 - Programming fundamentals 2.2.2 - Data types	**Component 1:** 1.4.1- Threats to computer systems and networks. Forms of attack. 1.4.2- Identifying and preventing vulnerabilities. Common prevention methods. **Component 2:** 2.2.3 -Arrays and sub programs. **Revisit** 1.1 – System architecture 1.2 - Memory and storage	**Component 1:** 1.5.1 – Operating Systems. The purpose and functionality of operating systems. 1.5.2 – Utility software. The purpose and functionality of utility software. **Component 2:** 2.1.2 – Standard sorting Algorithms. 2.3.1 Producing robust programs. Defensive design considerations, validation and maintainability. **Revisit** 1.3- Computer networks, connections and protocols. 2.1 - Algorithms	**Component 1:** 1.6.1 Ethical, legal, cultural and environmental impact of digital technology. Impact of digital technology on wider society and legislation relevant to Computer Science. **Component 2:** 2.3.2 – Testing. The purpose of testing, types of testing, Selecting and using suitable test data and refining algorithms. **Revisit** 2.4 - Boolean logic	**Component 2:** 2.2.3 Additional programming techniques. File handling, records, SQL and random number generation.	
Skills: Programming and the use of the Integrated Development Environment (IDE)	**Skills:** Linking Threats to methods of preventing the Vulnerability. Programming using subroutines	**Skills:** Important computer maintenance routines. Computational Thinking Designing, Creating, and Refining Algorithms	**Skills:** Impact Analysis Testing methodology.	**Skills:** Advanced coding skills	**Skills:**

Key Learning Outcomes: Explore different programming languages and their applications. Learn to use IDEs effectively for software development and debugging	**Key Learning Outcomes:** Identify and analyse potential security threats. Implement security measures to prevent vulnerabilities	**Key Learning Outcomes:** Understand the role and functions of operating systems. Explore utility software and its applications. Develop algorithmic thinking skills and problem-solving ability. Learn to design, implement, and optimize algorithms	**Key Learning Outcomes:** Analyse the ethical, legal, cultural, and environmental implications of digital technology Develop testing skills to ensure software reliability and functionality	**Key Learning Outcomes:** Explore advanced programming techniques and practices	**Key Learning Outcomes:**
End of term 1 evidence to cover: Programming skills, understanding security threats		**End of term 2 evidence to cover:** Knowledge of good coding skills and theory		**End of year evidence to cover:** Exam ready	
Rationale for sequence: Interleaving coding skills and identifying errors. Discuss the role of compilers and interpreters in translating code. Cover the features and facilities of programming languages.	**Rationale for sequence:** Introduce computer networks, including LANs, WANs, and the internet. Network protocols, their role in data transmission and communication. Teach network security concepts, such as encryption, firewalls, and cybersecurity measures. Understanding security is vital in the digital age.	**Rationale for sequence:** Explore operating systems, utility software, and the role of system software in managing hardware resources and providing user interfaces. Teach students how to analyse algorithms. Teach error handling and debugging, to help students write reliable and robust code.	**Rationale for sequence:** End this component by discussing ethical and environmental considerations related to computer systems. This encourages students to think about the broader impact of technology. Teach testing strategies to help write reliable and robust code. Crucial for quality assurance and bug detection	**Rationale for sequence:** Introduce databases and SQL. Teach students how to create, query, and manipulate databases, which is a valuable skill in software development. Teach skills to manipulate data, store configurations, and interact with users. Preparing for more complex software development tasks	**Rationale for sequence:**
Home Learning: Study and experiment with python programming languages.	**Home Learning:** Study real-world security incidents, identify threats	**Home Learning:** Experiment with various utility software and tools Solve algorithmic problems and puzzles, practice problem-solving strategies	**Home Learning:** Research studies on technology impact	**Home Learning:** Learn about advanced programming concepts and methods	

Explore different IDEs, set up development environments					
Reading / High Quality Text: GCSE Computer Science programming languages (teach-ict.com)	**Reading / High Quality Text:** GCSE Computer Science Network computer threats (teach-ict.com) GCSE Computer Science Network vulnerability prevention methods (teach-ict.com)	**Reading / High Quality Text:** GCSE Computer Science - purpose and features of operating systems (teach-ict.com) GCSE Computer Science utility software (teach-ict.com)	**Reading / High Quality Text:** GCSE Ethics and its role in technology and society (teach-ict.com) GCSE Environment and technology (teach-ict.com) GCSE Computer Science Ethical legal privacy issues (teach-ict.com)	**Reading / High Quality Text:** GCSE Computer Science Basic string manipulation (teach-ict.com) GCSE Computer Science Basic file handling operations (teach-ict.com) GCSE Computer Science SQL Introduction (teach-ict.com)	
Numeracy: Analyse language features and syntax.	**Numeracy:** Units and storage data	**Numeracy** Count, fix errors. Ensure data is correct.	**Numeracy:** Numerical data, statistics	**Numeracy:** Array – position/index value.	

Enrichment / opportunities to develop cultural capital (including careers, WRL and SMSC):
currently investigating links with Industry that would see specialists from Game Design, Programming and System Architects provide meaningful projects for students to complete in order to gain experience of what is required to operate within this field.