| **YEAR 11** | **Autumn Term**<br><br>**(Cycle 1)** | **Spring Term**<br><br>**(Cycle 2)** | **Summer Term**<br><br>**(Cycle 3)** |
|---|---|---|---|
| **Students will know and remember …** | **Defensive Design:**<br><br>Input sanitisation/validation; Planning for contingencies; Anticipating misuse; Authentication<br><br>**Testing**: Comments; Indentation; Sub-Programs<br><br>The purpose of testing (iterative and final/terminal)<br><br>How to identify syntax and logic errors<br><br>How to select and use suitable test data (normal, erroneous, extreme) and understand the difference between valid and invalid data<br><br>**Threats:**<br><br>Malware (Virus, Worm, Trojan); Phishing; People as the weak point; Brute force; Denial of service; Data interception and theft; SQL injection<br><br>Identifying and preventing vulnerabilities, including: Penetration Testing; Network Forensics; Network Policies; Anti-Malware; Firewalls; User Access Levels; Passwords; Encryption<br><br>**Operating Systems – Purpose & Functions:**<br><br>Operating Systems: User interface; Memory management/multitasking; Peripheral management and drivers; User management; File management<br><br>Utility systems software: Encryption software; Defragmentation; Data compression; The role and methods and backup (Full and incremental) | **Languages – High vs Low Level:**<br><br>Characteristics and purpose of different levels of programming language: High level languages and low level languages<br><br>The differences between high and low level programming languages<br><br>The purposes of translators<br><br>The characteristics of a compiler and an interpreter and the differences, benefits and drawbacks of using a compiler or an interpreter<br><br>**Computational thinking**<br><br>Principles of computational thinking: Abstraction; decomposition and algorithmic thinking.<br><br>Inputs, processes, and outputs for a problem.<br><br>**Practical Programming Skills** | **Theory Revision & Practical Programming**<br><br>Students will be able to apply all knowledge learned across the two years of study and apply it to the skills required for practical programming<br><br>**Exam skills practice** |

| | | | |
|---|---|---|---|
| | **Ethical & Legal**<br><br>How to investigate and discuss Computer Science technologies while considering: Ethical issues; Legal issues; Cultural issues; Environmental issues; Privacy issues<br><br>How key stakeholders are affected by technologies<br><br>The environmental impact of Computer Science<br><br>The cultural implications of Computer Science<br><br>Open source and proprietary software<br><br>Legislation relevant to Computer Science: The Data Protection Act 1998; Computer Misuse Act 1990; Copyrights Designs and patents Act; Creative Commons Licensing; Freedom of Information Act<br><br>**Practical Programming Skills** | | |

| | | | |
|---|---|---|---|
| **So that they can…** | **Defensive Design & Testing:**<br><br>Describe how the defensive design considerations can improve results in a more robust program<br><br>Describe how comments and indentation can improve the maintainability of programs<br><br>Correct syntax and logic errors found in given examples of code<br><br>Explain why it is important to consider defensive design<br><br>Explain the importance of commenting, indentation and sub-programs<br><br>Explain why different types of test data are suitable to given situations<br><br>**Threats:**<br><br>Identify the two main TYPES of attack (Passive, Active) that can take place on a network<br><br>Identify measures that can be implemented to reduce the threats faced by networks (Good Network Policy, Penetration Testing, Network Forensics, Passwords, User Access Levels, Anti-Malware, Encryption)<br><br>Describe how the two TYPES of attack might take place<br><br>Describe how each of the forms of threat work<br><br>Describe how the implementation of different measures could improve the security of data on a network<br><br>Give examples of what the intention/outcomes of the two TYPES of attack might be<br><br>Explain the purpose of the different forms of threat<br><br>**Operating Systems – Purpose & Functions:** | **Languages – High vs Low Level:**<br><br>Understand why an interpreter may potentially be better when designing a program and a compiler better for distributing a program<br><br>Understand the differences between high level language and low level language<br><br>Be able to translate high level language into machine code in order to run<br><br>Describe the advantages of writing a program in a high level language instead of an assembly language<br><br>**Computational thinking:**<br><br>Understand the principles of computational thinking and how they are used to define and refine problems<br><br>Understand the computational thinking means creating a logical solution to a problem, not thinking like a computer<br><br>Understand that abstraction means focusing on the important details and ignoring the rest<br><br>Understand that decomposition means breaking down a problem into smaller, easier to solve tasks<br><br>Understand that algorithmic thinking means creating a step by step set process of reaching a solution | |

| | | | |
|---|---|---|---|
| | Identify the each of the purposes of operating systems<br><br>Identify the different types of utility software<br><br>Describe the function of each aspect of the operating system<br><br>Explain why different types of OS are suitable for different purposes<br><br>Explain the benefits of using each of the types of utility software<br><br>**Ethical & Legal:**<br><br>Describe some scenarios where the following issues might exist:<br>Ethical issues<br>Legal Issues<br>Cultural Issues<br>Environmental Issues<br>Privacy Issues:<br>Including how it is now difficult to keep information private<br><br>Describe the laws surrounding Computer Science and data<br><br>Explain how key stakeholders are affected by technologies<br><br>Explain the impacts of the digital divide | | |