LAWN MANOR ACADEMY

Computer Science GCSE

INDEPENDENCE

KNOWLEDGE / LITERACY / NUMERACY / ORACY / AGENCY

CONTENT / TOPICS / CONCEPTS

Year 10

- Introduction to computer science and computational thinking
- Understanding algorithms: flowcharts, pseudocode, and basic programming in Python
- Core programming concepts: variables, data types, input/output, sequencing, selection
- Computer systems: CPU architecture, fetch-execute cycle, factors affecting performance, embedded systems
- Data representation: binary, denary, hexadecimal, file handling
- Memory and storage: RAM, ROM, primary vs secondary storage, compression methods
- Algorithms: searching (linear, binary), sorting (bubble, merge), efficiency and complexity
- Networking basics: LAN/WAN, network topologies, hardware, protocols, security
- Robust programming: defensive design, input validation, error handling, testing
- System security: threats, prevention, encryption, authentication
- Boolean logic: AND, OR, NOT gates, truth tables, logic circuits
- Advanced programming: functions, file and string manipulation, arrays/lists
- Mini project: design, code, test, and present a simple game or application

Year 11

- Mastery of programming fundamentals: sequence, selection, iteration, subroutines, arrays, file handling
- Data representation: character sets (ASCII, Unicode), images, sound, compression
- Systems software: operating systems, utility software, simulation in code
- Impacts of digital technology: ethical, legal, cultural, environmental, legislation
- Robust programming: defensive design, input validation, error handling, testing, secure login sys tems
- System security: threats, prevention, encryption, authentication, scenario analysis
- Translators and programming languages: compilers, interpreters, assemblers, high vs low level
- IDE features and debugging strategies
- Project work: requirements, planning, development, testing, evaluation, presentation
- Revision and exam preparation: theory and programming practice, mock exams, targeted revision, exam skills workshops, final programming challenge and celebration

VOCABULARY

Abstraction, Decomposition, Algorithm, Flowchart, Pseudocode, IDE, Variable, Data Types, Input, Output, Sequence, Selection, If/Else, CPU, Von Neumann, Fetch-Execute Cycle, Clock Speed, Cache, Cores, Embedded systems, RAM, ROM, Primary Storage, Secondary Storage, Hexadecimal, ASCII, File handling, Array, List , Linear search, Binary search, Bubble sort, Merge sort, Trace table, Efficiency, Lan, Wan, Star Topology, Mesh topology, Protocol, TCP/IP, HTTP, FTP, NIC, Switch, Router, Peer-to-peer, Client server, Firewall, Malware, Phishing, Social engineering, Validation, Boundary, Erroneous, Boolean, Logic gate, Scope, String manipulation , Project, Planning, Game loop.

Casting, Procedure, Subroutine, Global variable, Local variable, Authentication, Encryption, User access levels, Sample rate, Bit depth, Metadata, Lossy compression, lossless compression, Operating system, Peripheral management, Utility software, Defragmentation, Backup, Environmental impact, Ethical, Legal, Cultural, Data Protection Act (DPA), Computer Misuse Act (CMA) Copyright, Open source, Proprietary, High-level language, Low level programming language, Compiler, Assembler, Interpreter, Translators, Debugger, IDE features, Breakpoints, Project evaluation, Test plan, Automation, Audit log, Reflection.

SKILLS

Direct technical and cognitive abilities developed throughout the course:
Computational thinking: abstraction, decomposition, pattern recognition, algorithm design. Programming (Python): sequence, selection, iteration, functions, variables, data types, arrays/lists.
Problem solving: designing, testing, debugging, optimizing algorithms and programs. Use of Integrated Development Environments (IDEs) and software tools. Data representation (binary, hexadecimal, ASCII/Unicode, images, sound). Understanding computer systems: hardware, memory, storage, networks, system architecture. Applying defensive design, validation, error handling, and software testing. Interpreting and using flowcharts, pseudocode, and trace tables. Networking fundamentals: LAN/WAN, protocols, hardware Awareness of system security: types of threats, encryption, authentication. Evaluating the impact of digital technology: ethical, legal, cultural, environmental

ASSESSMENT

End-of-unit/topic tests: At the end of each major topic or unit, students complete written or practical assessments to check understanding of key concepts (e.g., systems architecture, programming fundamentals, networking, algorithms).
Mock exams: Typically held at the end of Year 10 and during Year 11 to simulate the final exam experience and identify areas for improvement. Educake/online quizzes: Used at intervals (often at the end of terms or topics) for retrieval practice, instant feedback, and consolidation. Project assessments: Mini programming projects, especially in Year 10 (e.g., game or quiz development), are assessed on planning, coding, testing, and evaluation. Practical programming tasks: Throughout both years, students are assessed on their ability to design, write, test, and debug code in Python or pseudocode.
Feedback and Improvement: Peer and self-assessment: Students regularly review each other's work and reflect on their own progress, especially after projects and practical tasks.
Targeted improvement tasks: After formal assessments, students act on feedback to address gaps in knowledge or skills.

ATTITUDE

Understanding others, behaviour and attitudes, SMSC, PHSE

Students work in pairs and groups on programming tasks, projects, and peer reviews, promoting cooperation, respect, and listening to others' perspectives.
Peer and self-assessment activities encourage reflection, empathy, and valuing feedback from classmates.
Group discussions, debates (e.g., on ethical and legal impacts of technology), and project presentations foster communication skills and appreciation of diverse viewpoints.
The curriculum includes activities where students present, critique, and celebrate each other's work, helping to build trust and mutual respect.

RESILIENCE

Character, personal Development, wellbeing and CIAG

Builds resilience through problem-solving and learning from mistakes
Encourages teamwork, leadership, and independence
Fosters responsibility and ethical awareness
Supports digital wellbeing and online safety
Celebrates achievements and promotes self-improvement
Raises awareness of computing careers and transferable skills