GCSE Computer Science – Curriculum Journey

ACET EMPOWERING ACHIEVEMENT SHIREBROOK ACADEMY STRIVING FOR EXCELLENCE

1.1 System Architecture
Understand the purpose of the Central Processing Unit (CPU) and the fetch-execute cycle.
Understand common characteristics of CPUs and how these could affect their performance.
Know the purpose and characteristics of embedded systems.
Recall examples of embedded systems.

1.2 Memory and Storage
Know why computers have primary storage and how this usually consists of RAM and ROM.
State the key characteristics of RAM and ROM.
Understand how virtual memory works and why it may be needed in a system.
Know why computers have secondary storage and recognise a range of secondary storage devices/media.
Compare advantages/disadvantages for different storage devices.
Learn about units and understand why data must be stored in binary format
Calculate the capacity of devices, the required capacity for given files, and file sizes of sound, images and text files.
Convert with denary numbers, binary numbers and hexadecimal equivalents. Perform binary shifts.
Understand how characters are represented in binary and use the term 'character set'
Understand how an image is represented as a series of pixels, represented in binary.
Understand the need for file compression and the effect of each type.

1.3 Computer Networks, Connections & Protocols
Learn about the types of network: LAN (local area network) & WAN (wider area network).
Know the characteristics of LANs & WANs including common examples of each
Understand what different factors can affect the performance of a network
Know what hardware is needed to connect stand-alone computers into a LAN
Recognise that the Internet is a worldwide collection of computer networks, e.g. The Cloud
Acknowledge the advantages and disadvantaged of The Cloud
Compare the benefits and drawbacks of wired versus wireless connection
Know and recall common protocols and their purposes
Identify advantages and disadvantages of star and mesh topologies
Understand the concept of layers and the TCP/IP model

1.4 Network Security
Recognise threats posed to computer systems and networks.
E.g. of forms of attack studied: Malware, Social engineering, Brute-force attacks, Denial of service attacks, Data interception and theft and the concept of SQL injection.
Know the principles of each form of attack, including how the attack is used and the purpose of the attack
Understand how to limit threats and remove vulnerabilities through the use of common prevention methods: Penetration testing, Anti-malware software, Firewalls, User access levels, Passwords, Encryption, Physical security

1.5 Systems Software
Know what each function of an operating system does
Recall features of a user interface
Understand that data is transferred between devices and the processor, and that this process needs to be managed.
Understand how User management functions and file management is used to organise and save files.
Know the purpose and functionality of utility software
Understand that computers often come with utility software, and how this performs housekeeping tasks

1.6 Ethical, Legal, Cultural & Environmental Impacts of Digital Technology
Recall, acknowledge and discuss the impacts of digital technology on wider society, including: Ethical issues, Legal issues, Cultural issues, Environmental issues, Privacy issues
Know the legislation relevant to Computer Science and the purpose of each piece of legislation and the specific actions it allows or prohibits
Understand the features of open source and the features of proprietary
Recommend a type of licence for a given scenario including benefits and drawbacks

2.1 Algorithms
Study the principles of computational thinking and understand how they are used to define and refine problems: Abstraction, Decomposition, Algorithmic thinking
Produce simple diagrams to show: the structure of a problem, subsections and their links to other subsections
Complete, write or refine an algorithm using: Pseudocode, Flowcharts, Reference language/high-level programming language
Identify common errors and use trace tables
Know, understand and use standard searching and sorting algorithms such as Binary search or Bubble sort.
Apply algorithms to data sets and identify the algorithm from given code or pseudocode

2.2 Programming Fundamentals
Recall the use of variables, constants, operators, inputs, outputs and assignments.
Use the three basic programming constructs used to control the flow of a program: Sequence, Selection, Iteration
Recognise and use comparison operators and arithmetic operators
Understand the use of data types and choose suitable data types for data in a given scenario and understand how data types can change
Use basic string manipulation and basic file handling operations such as: open, read, write, close
Understand the use of records and SQL queries to search for data
Understand the use of 1D and 2D arrays to store and retrieve data
How to use sub programs

2.3 Producing Robust Programs
Understand the issues a programmer should consider to ensure that a program caters for all likely input values
Understand how to deal with invalid data in a program
Know the difference between testing modules of a program during development and testing the program at the end of production
Understand syntax errors as errors which break the grammatical rules of the programming language and stop it from being run
Know logic errors as errors which produce unexpected outputs
Select and use suitable test data such as: normal, boundary, invalid, erroneous
Ability to create/complete a test plan

2.4 Boolean Logic
Use simple logic diagrams using the operators AND, OR and NOT
Combine Boolean operators using AND, OR and NOT
Understand truth tables for each logic gate
Recognise each gate symbol
Understand how to create, complete or edit logic diagrams and truth tables from given scenarios
Develop the ability to work with more than one gate in a logic diagram

2.5 Programming Language and Interated Development Environments
Know the characteristic and purpose of high-level & low-level programming language
Understand the purpose and need for translators
Know the difference, benefits and drawbacks of using a compiler or an interpreter
Have knowledge of the tools that an Integrates Development Environment (IDE) provides and understand how each tool can be used to help a programmer develop a program
Gain practical experience by using a range of tools within at least one IDE

During the course of study students will undertake a programming task/tasks
The programming task(s) will allow students to develop skills within the following areas of programming: Design, Write, Test, Refine
The task(s) will use a high-level text based programming language such as, Python or Small BASIC

J277/01: Computer Systems

J277/02: Computational thinking, algorithms and programming

Practical Programming Skills