Computer Science Framework for Learning 2022-2023

OCR GCSE COMPUTER SCIENCE - J277

<table>
<tr><td>Year Group</td><td colspan="6">Year 10</td></tr>
<tr><td>Subject Intent</td><td colspan="6">Computer Science encourages students to:

★ Understand and apply the fundamental principles and concepts of Computer Science, including abstraction, decomposition, logic, algorithms, and data representation
★ Analyse problems in computational terms through practical experience of solving such problems, including designing, writing and debugging programs
★ Think creatively, innovatively, analytically, logically and critically
★ Understand the components that make up digital systems, and how they communicate with one another and with other systems
★ Understand the impacts of digital technology to the individual and to wider society
★ Apply mathematical skills relevant to Computer Science</td></tr>
<tr><td>Subject Implementation</td><td>Autumn 1</td><td>Autumn 2</td><td>Spring 1</td><td>Spring 2</td><td>Summer 1</td><td>Summer 2</td></tr>
<tr><td rowspan="2">Knowledge</td><td colspan="2">Student will develop an understanding of:

Component 1:
➔ Architecture of the CPU
➔ CPU Performance
➔ Embedded systems
➔ Primary storage (Memory)
➔ Secondary storage

Component 2:
➔ Computational thinking
➔ Designing, creating and refining algorithms
➔ Programming fundamentals
➔ Defensive design</td><td colspan="2">Student will develop an understanding of:

Component 1:
➔ Units - Data storage
◆ Numbers
◆ Characters
◆ Images
◆ Sound
➔ Compression

Component 2:
➔ Data types
➔ Additional programming techniques
➔ Boolean logic</td><td colspan="2">Student will develop an understanding of:

Component 1:
➔ Threats to computer systems and networks
➔ Identifying and preventing vulnerabilities
➔ Operating systems
➔ Utility software

Component 2:
➔ Testing</td></tr>
<tr><td colspan="6"><u>Practical Programming</u> - All students must be given the opportunity to undertake a programming task(s), either to a specification or to solve a problem (or problems), during their course of study. Students may draw on some of the content in <u>both components</u> when engaged in Practical Programming.</td></tr>
</table>

Skills	**COMPONENT 1** **Students will study:** **1.1.1 Architecture of the CPU** - the purpose of the CPU o the fetch-execute cycle - common CPU components and their function: o ALU (Arithmetic Logic Unit), CU (Control Unit), Cache, Registers - Von Neumann architecture: o MAR (Memory Address Register), MDR (Memory Data Register), Program Counter, Accumulator **1.1.2 CPU Performance** - how common characteristics of CPUs affect their performance: o Clock speed, Cache size, Number of Cores **1.1.3 Embedded systems** - the purpose and characteristics of embedded systems - Examples of embedded systems **1.2.1 Primary storage (Memory)** - The need for primary storage - The difference between RAM and ROM - The purpose of ROM in a computer system - The purpose of RAM in a computer system - Virtual memory **1.2.2 Secondary storage** - The need for secondary storage - Common types of storage: o Optical, Magnetic, Solid state - Suitable storage devices and storage media for a	**COMPONENT 1** **Students will study:** **1.2.3 Units** - The units of data storage: o Bit, Nibble (4 bits), Byte (8 bits), Kilobyte (1000, (bytes or 1 KB), Megabyte (1,000 KB), Gigabyte (1,000 MB), Terabyte (1,000 GB), Petabyte (1,000 TB) - How data needs to be converted into a binary format to be processed by a computer. - Data capacity and calculation of data capacity requirements **1.2.4 Data storage** **Numbers** - How to convert positive denary whole numbers to binary numbers (up to and including 8 bits) and vice versa - How to add two binary integers together (up to and including 8 bits) and explain overflow errors which may occur - How to convert positive denary whole numbers into 2-digit hexadecimal numbers and vice versa - How to convert from binary to hexadecimal equivalents and vice versa - Binary shifts **Characters** - The use of binary codes to represent characters - The term 'character-set' - The relationship between the number of bits per character in a character set, and the number of characters which can be represented, e.g.: o ASCII, Unicode	**COMPONENT 1** **Students will study:** **1.4.1 Threats to computer systems and networks** - Forms of attack o Malware, Social engineering, e.g. phishing, people as the 'weak point', Brute-force attacks, Denial of service attacks, Data interception and theft, The concept of SQL injection **1.4.2 Identifying and preventing vulnerabilities** - Common prevention methods: o Penetration Testing, Anti-malware software, Firewalls, User access levels, Passwords, Encryption, Physical Security **1.5.1 Operating systems** - The purpose and functionality of operating systems: o User interface, Memory management and multitasking, Peripheral management and drivers, User management, File management **1.5.2 Utility software** - The purpose and functionality of utility software - Utility system software: o Encryption software, Defragmentation, Data Compression **COMPONENT 2** **Students will study:** **2.3.2 Testing** - The purpose of testing - Types of testing: o Iterative, Final/terminal

	given application - The advantages and disadvantages of different storage devices and storage media relating to these characteristics: o Capacity, Speed, Portability, Durability, Reliability, Cost **COMPONENT 2** **Students will study:** **2.1.1 Computational thinking** - Principles of computational thinking o Abstraction, Decomposition, Algorithmic Thinking. **2.1.2 Designing, creating and refining algorithms** - Identify the inputs, processes, and outputs for a problem - Structure diagrams - Create, interpret, correct, complete, and refine algorithms using: o Pseudocode, Flowcharts, Reference language/high-level programming language - Identify common errors - Trace tables - Standard searching algorithms: o Binary search, Linear search - Standard sorting algorithms: o Bubble sort, Merge sort, Insertion sort **2.2.1 Programming fundamentals** - The use of variables, constants, operators, inputs, outputs and assignments - The use of the three basic programming constructs used to control the flow of a program: o Sequence, Selection, Iteration (count- and condition- controlled loops)	**Images** - How an image is represented as a series of pixels, represented in binary - Metadata - The effect of colour depth and resolution on: o The quality of the image, The size of an image file **Sound** - How sound can be sampled and stored in digital form - The effect of sample rate, duration and bit depth on: o The playback quality, The size of a sound file **1.2.5 Compression** - The need for compression - Types of compression: o Lossy, Lossless **COMPONENT 2** **Students will study:** **2.2.2 Data types** - The use of data types: o Integer, Real, Boolean, Character and string, Casting **2.2.3 Additional programming techniques** - The use of basic string manipulation - The use of basic file handling operations: o Open, Read, Write, Close - The use of records to store data - The use of SQL to search for data - The use of arrays (or equivalent) when solving problems, including both one-dimensional (1D) and two-dimensional (2D) arrays	- Identify syntax and logic errors, Selecting and using suitable test data o Normal, Boundary, Invalid, Erroneous - Refining algorithms

	- The common arithmetic operators - The common Boolean operators AND, OR, NOT **2.3.1 Defensive design** - Defensive design considerations: o Anticipating misuse, Authentication - Input validation - Maintainability: o Use of sub programs, Naming conventions, Indentation, Commenting	- How to use subprograms (functions and procedures) to produce structured code - Random number generation - Defensive design considerations: o Anticipating misuse, Authentication - Input validation - Maintainability: o Use of sub programs, Naming conventions, Indentation, Commenting **2.4.1 Boolean logic** - Simple logic diagrams using the operations AND, OR and NOT - Truth tables - Combining Boolean operators using AND, OR and NOT - Applying logical operators in truth tables to solve problems	
Subject Impact	The GCSE Computer Science course will enable students to develop a real, in-depth understanding of how computer technology works, giving them an insight into what goes on 'under the lid' of a computer. You will need to think creatively, innovatively and logically to design and program solutions to real-world problems. Students will investigate the components that make up digital systems and how they communicate with one another and with other systems. They will also develop an understanding of the impacts of digital technology to the individual and to the wider society.		
Assessment	★ Formative Assessments at the end of each lesson topic, ***for e.g. Class work, Homework, Presentation, Short Recall Test, Practical Project, Quiz*** ★ 1 Summative assessment, ***for e.g. Internal School Examinations, End of Topic/Unit Tests*** **(Summative assessment covers content taught in Autumn Term)**	★ Formative Assessments at the end of each lesson topic, ***for e.g. Class work, Homework, Presentation, Short Recall Test, Practical Project, Quiz*** ★ 1 Summative assessment, ***for e.g. Internal School Examinations, End of Topic/Unit Tests*** **(Summative assessment covers content taught in Autumn and Spring Term)**	★ Formative Assessments at the end of each lesson topic, ***for e.g. Class work, Homework, Presentation, Short Recall Test, Practical Project, Quiz*** ★ 1 Summative assessment, ***for e.g. Internal School Examinations, End of Topic/Unit Tests*** **(Summative assessment covers content taught in the curriculum over the whole academic year)**