GCSE Computer Science Checklist

Unit 2 - Computational Thinking and Programming

(On-Screen Exam: 2 hour - 60 Marks - 30% of Qualification)

Topic	Sub-Topic	Explanation	I can statement	Studied	R	A	G
Problem Solving	Problem Solving	Use a systematic approach to problem solving including the use of decomposition and abstraction.	I can break a complex problem into smaller component parts.				
			I can remove unnecessary detail from a given scenario.				
			I can simplify a given scenario.				
		Use abstraction effectively to model selected aspects of the external world in an algorithm or program.	I can outline the inputs required for a real world situation.				
			I can outline the outputs required for a real world situation.				
			I can outline the processes involved in a real world situation.				
			I can explain the purpose of a given algorithm or program.				
		Use abstraction effectively to appropriately structure programs into modular parts with clear, well-documented interfaces.	I can produce modular programs which contain self-contained subroutines.				
			I can produce interfaces which are clear and unambiguous.				
			I can produce unambiguous documentation for a program.				
Algorithms and Programming Constructs	Algorithms	Use common methods of defining algorithms, including pseudo-code and flowcharts.	I can explain what an algorithm is.				
			I can explain the difference between pseudo-code and a flowchart.				
			I can write a piece of pseudo-code using the correct conventions.				
			I can produce a flowchart using the correct conventions.				
	Programming Constructs	Identify, explain and use sub routines in algorithms and programs.	I can identify subroutines in algorithms and explain their function.				
			I can explain what a subroutine is.				

			I can use subroutines to solve given problems.				
		Identify, explain and use sequence, selection and iteration in algorithms and programs.	I can explain the difference between sequence, selection and iteration in algorithms.				
			I can identify sequence, selection and iteration in algorithms.				
			I can explain the function of sequence, selection and iteration in algorithms.				
		Identify, explain and use counts and rogue values in algorithms and programs.	I can explain why iterations within a loop must eventually be terminated.				
			I can explain the different methods which can be used to terminate a loop.				
			I can explain how a count works.				
			I can describe what a rogue value is.				
		Identify and explain constructs in object orientated programs.	I can identify a superclass, class, objects, properties, methods and comments in an object orientated program.				
			I can explain the purpose of classes, objects, properties, methods and comments which have been used in an object-orientated program.				
	Variables	Identify, explain and use local and global variables in algorithms and programs.	I can explain the difference between a local and global variable.				
			I can identify local and global variables in algorithms.				
			I can construct algorithms which contain local and global variables.				
			I can construct a program which contains local and global variables.				
	Identifiers	Explain why the use of self-documenting identifiers and annotation are important in programs.	I can explain why self-documenting identifiers in a program are important.				
			I can explain why annotation is important in programs.				
		Give examples of self documenting identifiers and annotation.	I can produce a piece of code which contains a range of self-documenting identifiers.				
			I can produce a piece of code which contains annotation.				

	String Handling	Identify, explain and use routines for string handling in algorithms and programs.	I can identify different routines for string handling.				
			I can explain the use of different string handling techniques in algorithms and programs.				
			I can use a range of string handling techniques in algorithms and programs.				
	Mathematical Operations	Identify, explain and apply computing-related mathematical operations in algorithms and programs.	I can identify different mathematical operations which can be used in algorithms and programs.				
			I can explain the use of different mathematical operation in algorithms and programs.				
			I can use a range of mathematical operations in algorithms and programs.				
	Logical Operations	Identify, use and explain the logical operators AND, OR, NOT and XOR in algorithms and programs.	I can identify a range of different logical operations in algorithms and programs.				
			I can explain the use of different logical operations in algorithms and programs.				
			I can use a range of logical operations in algorithms and programs.				
	Sorting	Describe the characteristics of merge sort and bubble sort algorithms.	I can explain the difference between a merge and bubble sort algorithm.				
			I can describe how a merge sort algorithm works.				
			I can perform a merge sort algorithm on a given set of data.				
			I can describe how a bubble sort algorithm works.				
			I can perform a bubble sort algorithm on a given set of data.				
	Searching	Explain and use linear and binary search algorithms.	I can explain the difference between a linear and binary search algorithm.				
			I can perform a linear search on a given set of data.				
			I can explain what the term "divide and conquer" means.				

			I can perform a binary search on a given set of data.				
	Testing and Evaluating	Explain how an algorithm or program works and evaluate its fitness for purpose in meeting requirements.	I can explain how a given algorithm processes data to produce an outcome.				
			I can compare different algorithms and make judgements based on their efficiency.				
		Evaluate the efficiency of an algorithm or program using logical reasoning and test data	I can use logical reasoning to evaluate the efficiency of an algorithm or program.				
			I can use test data to evaluate the efficiency of an algorithm or program.				
			I can produce dry run algorithms using test data.				
			I can explain the outputs of dry run algorithms.				
Programming Languages	**Mark-up Languages**	Design, write, test and refine HTML pages using tags and their corresponding closures.	I can explain what a HTML tag is.				
			I can explain the purpose of commonly used HTML tags.				
			I can identify different HTML tags that have been used to produce a webpage.				
			I can use a range of HTML tags to mark up a document to specific requirements.				
	Object-Oriented Languages	Design, write, test and refine Java programs within the Greenfoot environment.	I can create new classes and extend existing classes.				
			I can create new and edit existing objects.				
			I can create new and edit existing worlds.				
			I can write and invoke methods.				
			I can change existing methods.				
			I can create new and edit existing properties.				
			I can add and remove objects from worlds.				
			I can use actors.				
			I can move objects around a world.				
			I can keyboard input.				
			I can add and play sounds.				
			I can implement and use parameter passing.				
			I can access one object from another.				
			I can implement object collision detection.				

			I can implement random number generation.				
			I can use the concept of inheritance and encapsulation.				
	Assembly Languages	Design, write, test and refine simple assembly programs.	I can design a simple assembly program.				
			I can write a simple assembly program which contains a range of mnemonics.				
			I can test a simple assembly program.				
			I can refine a simple assembly program.				
Data Structures and Data Types	**Implementing Data Structures**	Use one-dimensional and two dimensional arrays, files and records.	I can use one dimensional arrays in algorithms and programs to input, store, process and output data.				
			I can use two dimensional arrays in algorithms and programs to input, store, process and output data.				
			I can use records in algorithms and programs to input, store, process and output data.				
	Implementing Data Types	Use a variety of data types, including integer, boolean, real, character and string.	I can use the integer data type in algorithms and programs.				
			I can use the real data type in algorithms and programs.				
			I can use the character data type in algorithms and programs.				
			I can use the string data type in algorithms and programs.				
			I can use the boolean data type in algorithms and programs.				
			I can use different data types to hold data in variables, arrays and records in both algorithms and programs.				
	Variables and Constants	Assign, identify and explain the use of constants and variables in algorithms and programs.	I can assign, identify and use constants in programs and algorithms to store data that does not change.				
			I can explain where the use of constants and variables is appropriate.				
			I can assign, identify and use variables in programs and algorithms to store data that can change.				

			I can use variables and constants appropriately.				
		Describe the scope and lifetime of variables in algorithms and programs.	I can use local and global variables in algorithms and programs.				
			I can explain the difference in lifetime between local and global variables.				
Security Techniques	**Security Techniques**	Use appropriate security techniques, including validation and authentication.	I can use techniques that validate data in algorithms.				
			I can use techniques that authenticate information entered into an algorithm.				