GCSE OCR Computer Science

Unit	YEAR 11
2.1 Algorithms	
2.1.1 Computational Thinking	Define the term 'abstraction'
	Define the term 'decomposition'
	Define the term 'algorithmic thinking'
	Explain how decomposing a problem makes it easier to solve
	Explain why abstraction is used
	Give an example which demonstrates the term 'abstraction'
	Give an example which demonstrates the term 'decomposition'
2.1.2 Designing, Creating and Refining algorithms	Define the term 'algorithm'
	Draw the shape for a process in a flowchart
	Draw the shape for a decision in a flowchart
	Draw the shape for input/output in a flowchart
	Fill out a trace table for a simple program
	Fill out a trace table for a program with a loop
	Complete a program with some statements missing
	Given a jumbled-up program, reorder the program instructions to fix it
	Draw a flowchart for a simple program
	Draw a flowchart for a program with a for loop
	Draw a flowchart for a program with a while loop
	Draw a flowchart that contains a sub program
	Draw a structure diagram
	Identify where functions or procedures can be used in a program
2.2 Programming Fundamentals	
	Re-write a series of 'if-elseif' statements as a switch-case statement
	Define the term 'iteration' and give an example of what this would look like in code
	Define the term 'sequence' and give an example of how it is used in programming
	Define the term 'selection' and give two examples of selection in programming
	List the comparison operators
	List arithmetic operators and how to write these in ERL/Python

2.2.1 Programming Fundamentals	Explain what the MOD operation does, giving an example
	Explain what the DIV operation does, giving an example
	Give an example of how the AND operator can be used
	Give an example of how the OR operator can be used
	Give an example of how the NOT operator can be used
	State the name for the operators: AND, OR and NOT
	Explain the difference between a 'while' loop and a 'repeat until' loop
	Define the term 'variable'
	Explain the difference between a counter-controlled loop and condition-controlled loop
	Explain why creating a constant instead of a variable can be beneficial and give an example
2.2.2 Data Types	State five data types that are commonly included in a programming language
	Explain how the substring() method works, giving an example
	Explain the need for casting
	Give two examples of data, stating which data type would be used
	Identify a string in ERL and program code
	Describe concatenation and give an example
2.2.3 Additional Programming Techniques	Be able to concatenate strings
	Be able to slice strings
	Explain how SQL is used to search for data
	Describe when and how you might use an array
	Write an SQL commant that uses SELECT FROM WHERE
	State the difference between a function and a procedure
	Describe the advantages to using sub programs
	Be able to use an array and explain why they are useful
	Be able to use 2D arrays and explain, using an example, what 2D arrays are used for
	Define the terms 'field' and 'record'
	Be able to read and understand a program that uses basic file handling: open, read, write and close operations
	Be able to write a program that uses basic file handling: open, read, write and close operations
	Be able to create and use random numbers in a program

	Explain the difference between local and global variables and constants
	Create an example which demonstrates the scope of a local and global variable
	Pass an array as a parameter to a sub program, and return the array
2.3 Producing Robust Programs	
2.3.1 Defensive Design	Explain the phrase 'defensive design'
	State four techniques to create maintainable code
	Explain why creating maintainable code is desireable
	Write a comment in ERL and Python
	Explain why writing comments is useful
	State the naming conventions of variables and constants
	Explain why using sub programs makes code more maintainable
	Explain what is meant by code reuse
	Describe what types of validation could be used to check for a valid password
	Explain why validation should be used and give examples of when this is necessary
	Explain the difference between validation and verification
	Describe authentication and give examples of where this is applied
	Explain how a validation could be applied to user input of a username
	Explain issues that could occur if invalid data is entered into a program without any validation
2.3.2 Testing	Explain the difference between whitebox testing and blackbox testing
	Using an example, identify normal, boundary and invalid data to be used for a test
	Explain why testing needs to make use of different types of test data
	Compare the use of iterative testing and final/terminal testing
	Describe a syntax error and give a common example
	Describe a runtime error and give a common example
	Describe a logic error and give a common example
	Create a simple test plan, showing which data is to be used and what type of test data this is
	Give examples of erroneous data and explain why this should be rejected by the program

	Explain the difference between invalid data and erroneous data
	State which type of error prevents the program from being run
2.4 Boolean Logic	
2.4.1 Boolean Logic	Draw the logic gate for AND
	Draw the logic gate symbol for NOT
	Draw the logic gate symbol for OR
	Draw the truth table for the AND gate
	Draw the truth table for the OR gate
	Draw the truth table for the NOT gate
	State the type of gate that requires only one input
	Combine multiple gates together and state the logic equation for the final output
	Draw the truth table for a logic diagram with three inputs
2.5 Programming Languages and IDEs	
2.5.1 Languages	Give two examples of a high-level language
	Describe the term 'low-level language'
	Explain why programmers prefer to write in high-level languages
	Describe the difference between high and low-level lanugages
	Explain the purpose of a translator and why it is needed
	Give an example of a low-level language instruction
	Explain the role of a compiler and how it works
	Explain the role of an interpreter and how it works
	Compare the benefits and drawbacks of using a compiler vs an interpreter
2.5.2 The IDE	Explain the benefits to a programmer of using an IDE
	Explain how a programmer would use error diagnostics to help them write a program
	Explain the term 'run-time environment'
	Explain the term 'editor' and how it is used
	Explain how a translator can be included as part of an IDE and why this is useful