Paper 2 Computational thinking, algorithms and programming (J277/02)				
2.1 Algorithms				
2.1.1 Computational Thinking	Understand the principles of computational thinking (abstraction, decomposition, algorithmic thinking)			
2.1.2 Designing, Creating and Refining Algorithms	Design and refine algorithms using: Pseudocode, Flowcharts, Reference language / Python			
	Identify common errors and use Trace tables			
2.1.3 Searching and Sorting Algorithms	Understand and apply searching and sorting algorithms			
	Searching Algorithms: Linear, Binary			
	Sorting Algorithms: Bubble, Merge, Insert			
2.2 Programming fundamentals				
2.2.1 Programming Constructs	Use variables, constants, operators, inputs, outputs, and assignments in programming			
	Use of the 3 programming concepts: Sequence, Selection and Iteration (Count and Condition controlled Loops)			
	Arithmetic Operators (= - * / MOD DIV ^)			
	Comparison Operators (== != < <= > >=)			
	Boolean Operators (AND, OR, NOT)			
2.2.2 Data Types	Understand and apply different data types (integer, real, Boolean, character, string, casting)			
2.2.3 Additional Programming Techniques	Use basic string manipulation (concatenation, slicing)			
	File handling (Open, Read, Write, Close)			
	Use of SQL to search for data (SELECT FROM WHERE)			
	Use of Arrays/Lists when solving problems			
	Use of Sub programs (functions and procedures)			
	Random number generation			
2.3 Producing Robust Programs				
2.3.1 Defensive Design	Understand defensive design considerations (Anticipating misuse, authentication)			
	Input validation			

	Maintainability: Use of sub programs, indentation, commenting, naming conventions			
2.3.2 Testing	Understand the purpose of testing and the different types (Iterative, Final/terminal)			
	Select suitable test data (normal, boundary, Invalid/erroneous)			
2.4 Boolean Logic				
2.4.1 Boolean Logic	Simple logic diagrams using operators (AND, OR, NOT)			
	Complete Truth Tables			
	Combining Boolean operators using AND, OR,NOT			
	Apply Boolean operators (AND, OR, NOT) in truth tables to solve problems			
2.5 Programming Languages and IDEs				
2.5.1 Languages	Characteristics and purpose of high- and low-level languages			
	Purpose of translators			
	Characteristics of a compiler and an interpreter			
2.5.2 Integrated Development Environments (IDEs)	Use IDE tools (editors, error diagnostics, runtime environment, translators)			