2.1 – Algorithms		C&D	Rev Guide
2.1.1 Computational thinking			
	Principles of computational thinking:		p42
	o Abstraction	Open	p42
	o Decomposition	Open	p42
	o Algorithmic thinking	Open	p42
2.1.2 Designing, creating and refining algorithms			
	Identify the inputs, processes, and outputs for a problem	Open	-
	Structure diagrams	Open	-
	Create, interpret, correct, complete, and refine algorithms using:	Open	-
	o Pseudocode	^	p43
	o Flowcharts	^	p44
	o Reference language/high-level programming language	^	-
	Identify common errors	Open	-
	Trace tables	Open	p73
2.1.3 Searching and sorting algorithms			
	Standard searching algorithms:	-	p45
	o Binary search	Open	p45
	o Linear search	Open	p45
	Standard sorting algorithms:	^	p46
	o Bubble sort	Open	p46
	o Merge sort	Open	p47
	o Insertion sort	Open	p48

2.2 – Programming fundamentals		C&D	Rev Guide
2.2.1 Programming fundamentals			
	The use of variables, constants, operators, inputs, outputs and assignments	Open	p52
	The use of the three basic programming constructs used to control the flow of a program:	Open	p54
	o Sequence	^	-
	o Selection	^	p54-55
	o Iteration (count- and condition-controlled loops)	^	p55-56
	The common arithmetic operators	Open	p51
	The common Boolean operators AND, OR and NOT	Open	p57

2.2.2 Data types		C&D	Rev Guide
	The use of data types:	Open	p50
	o Integer	^	p50
	o Real	^	p50
	o Boolean	^	p50
	o Character and string	^	p50
	o Casting	^	p50
2.2.3 Additional programming techniques		**C&D**	**Rev Guide**
	The use of basic string manipulation	Open	p53
	The use of basic file handling operations:	Open	p63
	o Open	^	p63
	o Read	^	p63
	o Write	^	p63
	o Close	^	p63
	The use of records to store data	Open	p64
	The use of SQL to search for data	Open	p65
	The use of arrays (or equivalent) when solving problems, including both one-dimensional (1D) and two-dimensional arrays (2D)	Open	p61-62
	How to use sub programs (functions and procedures) to produce structured code	Open	p66-67
	Random number generation	Open	p60

2.3 – Producing robust programs		C&D	Rev Guide
2.3.1 Defensive design			
	Defensive design considerations:	Open	p70
	o Anticipating misuse	Open	p70
	o Authentication	^	p70
	Input validation	^	p70
	Maintainability:	Open	p69
	o Use of sub programs	^	p69
	o Naming conventions	^	p69
	o Indentation	^	p69
	o Commenting	^	p69
2.3.2 Testing			
	The purpose of testing	Open	p71

	Types of testing:	^	p72
	o Iterative	^	p72
	o Final/terminal	^	p72
	Identify syntax and logic errors	Open	p71
	Selecting and using suitable test data:	Open	p72
	o Normal	^	p72
	o Boundary	^	p72
	o Invalid/Erroneous	^	p72
	Refining algorithms	Open	-

2.4 – Boolean logic		**C&D**	**Rev Guide**
2.4.1 Boolean logic			
	Simple logic diagrams using the operators AND, OR and NOT	Open	p57-59
	Truth tables	Open	p57-59
	Combining Boolean operators using AND, OR and NOT	Open	p57-59
	Applying logical operators in truth tables to solve problems	Open	p57-59

2.5 – Programming languages and Integrated Development Environments		**C&D**	**Rev Guide**
2.5.1 Languages			
	Characteristics and purpose of different levels of programming language:	Open	p74
	o High-level languages	^	p74
	o Low-level languages	^	p74
	The purpose of translators	Open	p74
	The characteristics of a compiler and an interpreter	Open	p74
2.5.2 The Integrated Development Environment (IDE)			
	Common tools and facilities available in an Integrated Development Environment (IDE):	Open	p75
	o Editors	^	p75
	o Error diagnostics	^	p75
	o Run-time environment	^	p75
	o Translators	^	p75