2.1 Elements of computational thinking Understand what is meant by computational thinking		C&D	Isaac	Textbook
2.1.1 Thinking abstractly				
	(a) The nature of abstraction.	Open	Open	
	(b) The need for abstraction.	Open	^	
	(c) The differences between an abstraction and reality	Open	^	
	(d) Devise an abstract model for a variety of situations.	Open	^	
2.1.2 Thinking ahead				
	(a) Identify the inputs and outputs for a given situation.	Open	Open	
	(b) Determine the preconditions for devising a solution to a problem.	Open	^	
	(c) The nature, benefits and drawbacks of caching	Open	^	
	(d) The need for reusable program components.	Open	^	
2.1.3 Thinking procedurally				
	(a) Identify the components of a problem	Open	Open	
	(b) Identify the components of a solution to a problem.	Open	^	
	(c) Determine the order of the steps needed to solve a problem	Open	^	
	(d) Identify sub-procedures necessary to solve a problem.	Open	^	
2.1.4 Thinking logically				
	(a) Identify the points in a solution where a decision has to be taken.	Open	Open	
	(b) Determine the logical conditions that affect the outcome of a decision.	Open	^	
	(c) Determine how decisions affect flow through a program.	Open	^	
2.1.5 Thinking concurrently				
	(a) Determine the parts of a problem that can be tackled at the same time.	Open	Open	
	(b) Outline the benefits and trade offs that might result from concurrent processing in a particular situation.	Open	^	

2.2 Problem solving and programming How computers can be used to solve problems and programs can be written to solve them (Learners will benefit from being able to program in a procedure/imperative language and object oriented language.)		C&D	Isaac	Textbook
2.2.1 Programming techniques				
	(a) Programming constructs:			
	• sequence	Open	Open	
	• iteration	^	Open	
	• branching	^	Open	

	(b) Recursion, how it can be used and compares to an iterative approach.	Open	Open	
	(c) Global and local variables.	Open	Open	
	(d) Modularity, functions and procedures, parameter passing by value and by reference.	Open	Open	
	(e) Use of an IDE to develop/debug a program.	Open	Open	
	(f) Use of object oriented techniques.	Open	Open	
2.2.2 Computational methods				
	(a) Features that make a problem solvable by computational methods.	Open	Open	
	(b) Problem recognition.	Open	Open	
	(c) Problem decomposition	Open	Open	
	(d) Use of divide and conquer.	Open	Open	
	(e) Use of abstraction	Open	Open	
	(f) Learners should apply their knowledge of:			
	• backtracking	Open	Open	
	• data mining	Open	^	
	• heuristics	Open	^	
	• performance modelling	Open	^	
	• pipelining	Open	^	
	• visualisation to solve problems.	Open	^	

2.3 Algorithms **The use of algorithms to describe problems and standard algorithms**		**C&D**	**Isaac**	**Textbook**
2.3.1 Algorithms				
	(a) Analysis and design of algorithms for a given situation.	Open		
	(b) The suitability of different algorithms for a given task and data set, in terms of execution time and space.	Open	Open	
	(c) Measures and methods to determine the efficiency of different algorithms, Big O notation (constant, linear, poly	Open	Open	
	(d) Comparison of the complexity of algorithms.	Open	Open	
	(e) Algorithms for the main data structures:	Open		
	• stacks	Open	Open	
	• queues	Open	Open	
	• trees	Open	Open	
	• linked-lists	Open	Open	
	• depth-first (post-order) traversal of trees	Open	Open	
	• breadth-first (post-order) traversal of trees	Open	Open	

	(f) Standard algorithms:	Open		
	bubble sort	Open	Open	
	insertion sort	Open	Open	
	merge sort	Open	Open	
	quick sort	Open	Open	
	Dijkstra's shortest path	Open	Open	
	A* algorithm	Open	Open	
	binary search	Open	Open	
	linear search	Open	Open	