Curriculum Map

Sandringham School
'Everybody can be Somebody'

Subject: *Computer Science: Paper 2 – Algorithms and programming* **Year group:** *12*

Time period	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
Content *Declarative Knowledge –* *'Know What'*	*2.1.2 -Thinking ahead* (a) inputs and outputs (b) preconditions for devising a solution to a problem. *2.2.1 Programming techniques* (e) Use of an IDE to develop/debug a program (a) Programming constructs: sequence, iteration ,branching *2.1.4 -Thinking logically* (a) Decision making (b) Conditions that affect decision making (c) Program flow decisions *2.2.1 Programming techniques* b) Recursion	*2.1.3 - Thinking Procedurally* (a) (b) (c) & (d) Procedural languages paradigm used for problem solving *2.2.1 Programming techniques* (c) Global and local variables. (d) Modularity, functions and procedures, parameter passing by value and by reference *2.2.2 Computational methods* (a) Features that make a problem solvable by computational methods (b) Problem Recognition (c) Problem Decomposition (d) Use of divide and conquer	*1.4.2 – Data structures* (a) Arrays (of up to 3 dimensions), records, lists and tuples *2.3.1 - Algorithms* (a) Analysis and design of algorithms (b) The suitability of different algorithms for given data sets (c) Measures and methods to determine the efficiency of different algorithms through the Big O notation (d) Comparison of the complexity of algorithms.	*2.3.1 - Algorithms* (d) Standard algorithms (Bubble sort, insertion sort, merge sort, quick sort, binary search and linear search) (c) Algorithms for the main data structures, (Stacks, queues, trees, linked lists, depth-first (post-order) and breadth-first traversal of trees)	*1.3.4 – Web technologies* (b) Search engine indexing (c) PageRank Algorithm 1.*2.3 - Software Development* (a) The different software methodologies, waterfall lifecycle, agile methodologies, extreme programming, the spiral model and rapid application development. (b) The relative merits and drawbacks of different methodologies	*Component 3* *Non Exam Assessment – programming project* Introduction to the programming project
Skills *Procedural Knowledge –* *'Know How'*	*2.1.2 -Thinking ahead* a) Identify the inputs and outputs for a given situation. b) Determine the preconditions for devising a solution to a problem. *2.2.1 Programming techniques* e) Use an IDE to develop/debug a program-students are introduced to the C# programming language developed through Visual Studio IDE. They use practical programming exercises to understand the	*2.1.3 - Thinking Procedurally* a) Identify the components of a problem. (b) Identify the components of a solution to a problem. (c) Determine the order of the steps needed to solve a problem. (d) Identify sub-procedures necessary to solve a problem. *2.2.1 Programming techniques* (c) (d) Programming exercises involving functions, procedures and	*1.4.2 – Data structures* a) Students will be given problems to solve that require the use of a tuple, 1D or 2D array or a dynamic list. They must use their knowledge and design and create solutions to these problems. The must know how to create and manipulate arrays. Lists and tuples though slicing, sorting, searching and extracting the data necessary to their given problem from the data	*2.3.1 - Algorithms* d) Students will look at the standard algorithms for searching and sorting data sets (Bubble sort, insertion sort, merge sort, quick sort, binary search and linear search) and compare their time and space complexity to gain a further understanding as to the efficiency of the algorithms for given data sets. Students will implement these algorithms into program code to further	*1.3.4 – Web technologies* (b) (c) Students will know how search engine indexing collects, parses and stores web data to facilitate fast and accurate information retrieval. Students will also investigate the PageRank algorithm understanding how it ranks web pages based on given criteria. 1.*2.3 - Software Development*	*Component 3* *Non Exam Assessment – programming project* Students will be expected to (3.1) analyse, (3.2) design, (3.3) develop, (3.4) test, (3.5) evaluate and document a program written in a suitable programming language. The underlying approach to the project is to apply the principles of computational thinking to a practical coding

	facilities of this IDE. They acquire further knowledge of the IDE by designing, implementing and debugging program code. a) Programming exercises involving branching (IF, nested IF, SELECT/CASE statements) to reiterate and consolidate their theoretical knowledge in a practical application. Programming exercises involving iteration (FOR,WHILE,REPEAT UNTIL) *2.1.4 -Thinking logically* a) Identify the points in a solution where a decision has to be taken. (b) Determine the logical conditions that affect the outcome of a decision. (c) Determine how decisions affect flow through a program. *2.2.1b) Recursion* How it can be used and compares to an iterative approach. Programming exercises involving recursion e.g. factorial.	parameters. Understanding parameter passing by value and reference through practical activities. Look at how we use and define local and global variables in our programs. Programming exercises involving subroutines understanding where it is necessary to create a function with a return value and where it is necessary to write a procedure. *2.2.2 Computational methods* (a) (b) (c) & (d) Students will know how to break down given problems into manageable solvable solutions, they will apply the techniques of abstraction and decomposition to the problem. They will then create solutions to the given problems through program code solutions.	structure. Students will know the differences between the data structures in which is static and dynamic. They will utilise this skill in problem solving by identifying the correct data type necessary for their given solution. *2.3.1 - Algorithms* a) Analysis and design of algorithms for a given situation and determine b) The suitability of different algorithms for a given task and data set, in terms of execution time and space. (c) Understand the measures and methods to determine the efficiency of different algorithms, Big O notation (constant, linear, polynomial, exponential and logarithmic complexity). (d) Compare the complexity of algorithms in terms of worst case big O notation time and space complexity.	understand their ability to search or sort data. Students need to understand the Big O notation for the searching and sorting algorithms for larger and smaller data sets. (c) Students will know how the different algorithms for each of the data structures is designed and implemented. They will need to be able to read, trace and write code to implement features of these data structures. Students will also understand the time and space complexity of the algorithms and compare the time and space complexity of each in relation to the size of the dataset.	(a) Understand the models that can be followed to produce a software system; the waterfall lifecycle, agile methodologies (specifically extreme programming); the spiral model and rapid application development). (b) need to understand the tasks, processes, benefits and drawbacks of each model and the similarities and differences between each. They need to understand where each model is most suitable to use, and be able to justify the use in a situation.	problem. Students are expected to apply appropriate principles from an agile development approach to the project development.
Key Questions	What is computational thinking? How do we apply decomposition and abstraction to a given problem? What are the inputs, pre-conditions, processes and outputs for a given problem? What are the programming constructs used in a any language? What are	Where do I use local and global variables? Why is it poor programming practice to use Global variables? What is a subroutine? What is the difference between a function and a procedure? What is the function definition? What are parameters? Why do	What is an array? What is a list? What is a tuple? What is a static and dynamic data structure? When do I use either a static or dynamic data structure? What is time complexity? What is space complexity? Why do we measure	What are the different searching algorithms? What is divide and conquer? When is it suitable to use a linear search? Binary search? What are the different sorting algorithms? What is the time complexity of the different searching	What is a search engine? What is a web crawler? Why do we use a web crawler? What is search engine indexing? What is page rank? What factors affect a page's rank in the page rank algorithm? What are the different types of software	What is a problem definition? What or who is a stakeholder? What do we mean by the complexity of our project? How do I describe the essential features of a computational?

Curriculum Map

Sandringham School
'Everybody can be Somebody'

	the main components of an IDE? When is recursion used to solve a problem. How do I identify the base case for a recursive function? How do I design and implement a recursive function?	subroutines use parameters? When do we pass parameters by reference or by value? What are the benefits of using functions in writing larger program code? What is modularity of code? Why is it useful? When do we use the procedural paradigm programming model? What if we are writing large programs?	algorithms in terms of their space and time complexity? What is best, average and worse case. What is a data set? What is big data? What does big O notation use the worst case? What do we mean by time complexity using constant, linear, logarithmic polynomial, exponential and factorial? What are tractable and intractable problems?	and sorting algorithms? Why is time complexity important in relation to the data set given for each of the algorithms? What are the different data structures used to hold data and programs in memory? How is each of the data structures designed and implements as algorithms? How do I implement it? What is the time and space complexity of the data structure.	development methodologies used? Where do we need to use these in a given situation? What are the benefits and drawbacks of each model and where is it appropriate to use these?	solution explaining these choices. Explain the limitations of the proposed solution. Justify the solution requirements? What is success criteria? What is the iterative development process? What is iterative and final testing? What is algorithmic design? How does my success criteria inform my final product?
Assessment	End of unit tests, Past exam questions to consolidate learning Exam style HBL questions				End of unit tests, Exam style HBL questions Past exam questions to consolidate learning Practical activities using HTML, CSS And Javascript. Trial exams Programming project analysis	
Literacy/Numeracy/ SMSC/Character	Computational literacy Scaffolded answers to LAQ, guided through AO1, AO2 and AO3 evaluative skills	Computational literacy Scaffolded answers to LAQ, guided through AO1, AO2 and AO3 evaluative skills Mathematical computation Data handling Linear Algebra Discrete mathematics	Computational literacy Exemplar modelling of answers Understanding of key word definitions. Scaffolded answers to LAQ, guided through AO1, AO2 and AO3 evaluative skills Mathematical computation Data handling Linear Algebra Discrete mathematics Graph theory	Programming language literacy Computational literacy Exemplar modelling of answers Understanding of key word definitions. Scaffolded answers to LAQ, guided through AO1, AO2 and AO3 evaluative skills Mathematical computation Data handling Linear Algebra Discrete mathematics Graph theory	Programming language literacy Computational literacy Exemplar modelling of answers Understanding of key word definitions. Scaffolded answers to LAQ, guided through AO1, AO2 and AO3 evaluative skills	Programming language literacy Computational literacy Exemplar modelling of answers Understanding of key word definitions. Scaffolded answers to LAQ, guided through AO1, AO2 and AO3 evaluative skills