Curriculum Map 2021/2022

ST MARY'S CE HIGH SCHOOL

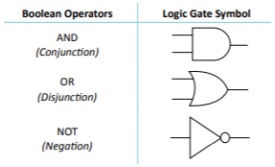
YEAR 11 COMPUTER SCIENCE

The Computer Science curriculum encourages students to develop their understanding and application of the core concepts in computer science. Students also analyse problems in computational terms and devise creative solutions by designing, writing, testing and evaluating programs.

	Autumn 1a	Autumn 1b	Spring 2a	Spring 2b	Summer 3a	Summer 3b

CONTENT
 	
Declarative / core / powerful Knowledge – 'Know What'
	Algorithms:
	- Computational Thinking
	  ○ Principles of computational thinking
	- Designing, creating and refining algorithms
	  ○ Identify the inputs, processes and outputs for a problem
	  ○ Structure diagrams
	  ○ Create, interpret, correct, complete and refine algorithms using: Pseudocode, Flowcharts, Reference Language
	  ○ Identify common errors
	  ○ Trace tables
	- Searching and sorting algorithms
		Programming Fundamentals:
		- Programming fundamentals
		  - The use of variables, constants, operators, inputs, outputs and assignments
		  - The use of the three basic programming constructs used to control the flow of a program:
		  ○ Sequence
		  ○ Selection
		  ○ Iteration (count- and condition-controlled loops)
		  - The common arithmetic operators
		  - The common Boolean operators AND, OR and NOT
		  - The use of data types:
		  ○ Integer
		  ○ Real
		  ○ Boolean
		  ○ Character and string
			Producing Robust Programs:
			- Defensive Design
			  - Defensive design considerations:
			  ○ Anticipating misuse
			  ○ Authentication
			  - Input validation
			  - Maintainability:
			  ○ Use of sub programs
			  ○ Naming conventions
			  ○ Indentation
			  ○ Commenting
			- Testing
			  - The purpose of testing
			  - Types of testing:
			  ○ Iterative
			  ○ Final/terminal
			  - Identify syntax and logic errors
			  - Selecting and using suitable test data:
			  ○ Normal
			  ○ Boundary
			  ○ Invalid/Erroneous
			  - Refining algorithms
			  -
				Boolean Logic
				- Boolean Logic
				  - Simple logic diagrams using the operators AND, OR and NOT
				  - Truth tables
				  - Combining Boolean operators using AND, OR and NOT
				  - Applying logical operators in truth tables to solve problems
					Programming Languages and Integrated Development Environments:
					- Languages
					  - Characteristics and purpose of different levels of programming language:
					  ○ High-level languages
					  ○ Low-level languages
					  - The purpose of translators
					  - The characteristics of a compiler and an interpreter
					  - Common tools and facilities available in an Integrated Development Environment (IDE):
					  ○ Editors
					  ○ Error diagnostics
					  ○ Run-time environment
					  ○ Translators
						Revision and exam techniques revisited in preparation for final exams

	○ Standard searching algorithms (Binary & Linear) ○ Standard sorting algorithms (Bubble, Merge & Insertion)	○ Casting - The use of basic string manipulation - The use of basic file handling operations: ○ Open ○ Read ○ Write ○ Close - The use of records to store data - The use of SQL to search for data - The use of arrays (or equivalent) when solving problems, including both one-dimensional (1D) and two-dimensional arrays (2D) - How to use sub programs (functions and procedures) to produce structured code - Random number generation				
Skills *Procedural Knowledge – 'Know How'*	Understanding of these principles and how they are used to define and refine problems. Produce simple diagrams to show: -The structure of a problem -Subsections and their links to other subsections	Practical use of the techniques in a high-level language within the classroom Understanding of each technique and recognise the following operators: - == equal to - != not equal to - < less than	Understanding of the issues a programmer should consider to ensure that a program caters for all likely input values Understanding of how to deal with invalid data in a program Authentication to confirm the identity of a user	Knowledge of the truth tables for each logic gate and ability to recognise them Understanding of how to create, complete or edit logic diagrams and truth tables for given scenarios	The differences between high- and low-level programming languages The need for translators The differences, benefits and drawbacks of using a compiler or an interpreter Knowledge of the tools that an IDE provides	

	Complete, write or refine an algorithm using the techniques listed Identify syntax/logic errors in code and suggest fixes Create and use trace tables to follow an algorithm Understand the main steps of each algorithm and any pre-requisites of an algorithm Apply the algorithm to a data set and identify an algorithm if given the code or pseudocode for it	- <= less than or equal to - > greater than - >= greater than or equal to - + addition - - subtraction - * multiplication - / division - MOD modulus - DIV quotient - ^ exponentiation (to the power) Practical use of the data types in a high-level language within the classroom Ability to choose suitable data types for data in a given scenario Understand that data types may be temporarily changed through casting, and where this may be useful Practical use of the additional programming techniques in a high-level language within the classroom Ability to manipulate strings, including: - Concatenation - Slicing	Practical experience of designing input validation and simple authentication (e.g. username and password) Understand why commenting is useful and apply this appropriately The difference between testing modules of a program during development and testing the program at the end of production Syntax errors as errors which break the grammatical rules of the programming language and stop it from being run/translated Logic errors as errors which produce unexpected output Normal test data as data which should be accepted by a program without causing errors Boundary test data as data of the correct type which is on the very edge of being valid Invalid test data as data of the correct data type which	Ability to work with more than one gate in a logic diagram  Truth Tables: AND A | B | A AND B 0 | 0 | 0 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1 OR A | B | A OR B 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1 NOT A | NOT A 0 | 1 1 | 0	How each of the tools and facilities listed can be used to help a programmer develop a program Practical experience of using a range of these tools within at least one IDE	

		Arrays as fixed length or static structures Use of 2D arrays to emulate database tables of a collection of fields, and records The use of functions and procedures and where to use them effectively The use of the following within functions and procedures: - local variables/constants - global variables/constants - arrays (passing and returning) SQL commands: - SELECT - FROM - WHERE Be able to create and use random numbers in a program	should be rejected by a computer system Erroneous test data as data of the incorrect data type which should be rejected by a computer system Ability to identify suitable test data for a given scenario, create and complete a test plan			
Key Questions	What are the different types of computational thinking and what do they do?	What are the different fundamentals used within programming? What are the three basic programming constructs	What are the defensive design considerations that need to be considered? What is input validation and why is it used?	What is the difference between the different Boolean logic operators? How do we use truth tables?	What are the different characteristics between high- and low- level programming languages? What is the purpose of translators?	

	What do the different symbols of a flow diagram mean? What is the difference between a Binary and a Linear search? What do the following searches do: Bubble, Merge & Insertion?	used to control the flow of a program? What are the common arithmetic operators used in programming? What are the differences between the different Boolean operators? What are the different data types and what do they do? How can we manipulate data? What are the basic file handling operations? How can we use records to store data? What does SQL stand for? How can we use arrays to solve problems?	How do we maintain programs? What is the purpose of testing and what are the different types of testing? What is the difference between syntax and logic errors? How do we refine algorithms?	How do we calculate the number of columns and rows needed for a truth table? How do we combine Boolean operators? How can we apply logical operators to truth tables?	What are the characteristics of a compiler? What are the characteristics of an interpreter? What are the common tools and facilities available within an Integrated Development Environment?	
Assessment	End of Algorithms assessment including previous modules	End of Programming Fundamentals assessment including previous modules	End of Producing Robust Programs assessment including previous modules	End of Boolean assessment including previous modules	End of Programming languages and Integrated Development Environments assessment including previous modules	