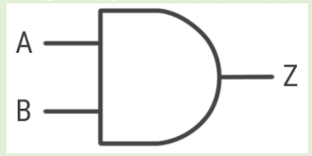
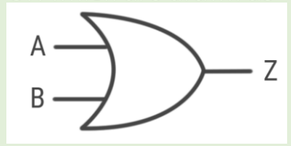
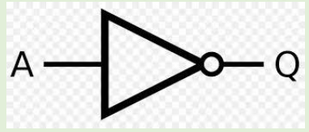
Computer Science Yr 11 Curriculum Map

	Autumn term	Spring term	Summer term
Unit of Work	**Algorithms**	**Boolean Logic**	**Paper 2 Recap**
Fertile Questions	How can I create a program to organise and find data from a list?	How is the CPU and programs able to make decisions based around binary inputs?	How can I learn independent revision skills to support my understanding of paper 2 content

Key Concepts	• Computational thinking is a skill that allows users to define and refine problems
	  ○ Abstraction- Looking at only the important information to the problem and ignoring additional details
	  ○ Decomposition- Breaking down a large problem into smaller more manageable chunks
	  ○ Algorithmic thinking- Thinking in logical steps to solve problems efficiently
	• Use a variety of tools to plan, interpret, correct or complete algorithms including:
	  ○ Pseudocode- Not real code but reads like a set of instructions
	  ○ Flowcharts- using specific symbols to show the order in a program
	  ○ Structure diagrams- used to plan out larger coded projects to think about separate parts of a program
	• Searching algorithms are used to find data in a set
	  ○ Linear search- When looking for a piece of data this search will start at the first item and check to see if it matches. If it does, the search is over as the item has been found, if it is not then the search goes to the next item beside it and checks. This is continued until the item is found or the list ends
	  ○ Binary search- Can only be completed if the list is in numerical or alphabetical order. When looking for a piece of data this search will start at the middle item and check to see if it matches. If it does, the search is over, if not it will check to see if the item is higher or lower than the searched item. If it is higher, it will discard the second half and then continue the steps above.
	• Sorting algorithms are used to organise data into numerical or alphabetical order.
	  ○ Bubble sort- The first two items are compared. If they are not in order they are swapped, if they are in order they are left. Then goes to the next two items and repeats the process. Once it reaches the end of the
		• Logic gates are used within the CPU by taking inputs, applying the logic operator and then determining the binary output. Logic gates include:
		  ○ AND which are drawn as
		 
		  ○ OR which are drawn as
		 
		  ○ NOT which are drawn as
		 
		• AND gates must have both inputs as 1 to have an output of 1
		• OR gates can have either input or both as 1 to have an output of 1
		• NOT gates will take the input and switch it
		• Logic circuits can be used by using a combination of logic gates
		• Truth tables are used to track and record all possible inputs in a logic circuit and determine the corresponding output
			• No new concepts for this topic as we revisit content learnt in year 10 and 11

	first pass, if any swaps were made it will go through the process again. ○ Merge sort- First the list of data is split until each item is in a list of one. Each list is then merged together and the data is ordered. ○ Insertion sort- Two lists are created, a sorted list and an unsorted list. Starting with the second item, it is compared to the item in the sorted list and inserted into the correct position. This continues until all unsorted items have been sorted.		
Assessment	• 35 mark exam questions	• 15 mark exam questions	• Mini mocks completed once a fortnight to review material on paper 2
Experiences	• Creating flowcharts, structure diagrams and writing in pseudocode • Physically acting out search and sort algorithms to understand how they work • Creating exemplar search and sort algorithms to refer back to when revising	• Drawing out logic gates • Building logic circuits by combining logic gates • Filling in truth tables	• Create a coded program that acts as a revision game about paper 2 content that you and others could use to revise • Students have reteach sessions each lesson and then have the remainder of the time to code and create their game
Skills	• Answering practice exam questions to build skills and confidence in what these questions look like • Building revision skills through developing materials in lesson that can be used to revise from later	• Deducing possible outputs from given inputs and logic gates	• Developing practical coding skills useful for paper 2 whilst reviewing paper 2 content in an engaging way
Unit of Work	**Programming Fundamentals (Lessons 9-19)**	**Programming Languages and IDEs**	
Fertile Questions	How can I develop my coding skills to include additional techniques to create a robust and engaging program?	How do computers understand human languages?	
Key Concepts	• You can use file handling operations in Python to open, read, write and close files • Different types of data can be stored in a record • SQL can be used to search for data using the commands SELECT, FROM, WHERE o SELECT – is used to select which fields of data will be outputted o FROM – is used to select the database o WHERE – is used to set parameters for the data you'd like to obtain • 1D arrays allow you to store multiple data entries under one identifier, however all the data needs to be of the same type. • 2D arrays allow you to create a table of data entries under of identifier, again all the data needs to be of the same type. • Subprograms are used to create a section of code that can be called upon later in your program. There are two types of subprograms: o Procedures – do not return a value, may or may not have parameters o Functions – must return a value, must have parameters	• Humans code in high-level languages and computers can only understand low-level languages • A high level language is one that reads closer to English to its easier for humans to understand • A low level language is one that computers understand so is binary • A translator is needed to convert the high level language humans write in to a low level language that the computer can understand. • There are two types of translators that can be used: o Interpreter- works by translating line by line and will stop when an error occurs. Does not make an executable file. o Complier- translate the entire program which can take longer but creates an executable file which doesn't need to be translated after it has been created. • Integrated Development Environments (IDEs) provide tools and facilities to support programmers in writing code. They include features such as: o Editors- a space to write your program o Error diagnostics- when a syntax error	

	• When a subprogram is made the variables that exist in that subprogram are called local variables as they do not exist in the main program. Variables that are declared outside of the subprogram are called global variables as they exist throughout the program	occurs it will highlight the line that had the error and suggest a solution to the error ○ Run-time environment- a separate window that executes your code so the programmer can see how it would look to a user ○ Translator- translates the high level language into a low level language for the programmer	
Assessment	• Coded game incorporating skills learnt throughout the topic • 35 mark exam questions	• 35 mark exam questions	
Experiences	• Use structure diagrams to plan and organise a large coded game • How to use sub-programs and arrays in their program • How to add in time and randomness to their program • Writing programs onto paper following exam criteria to prepare for exams	• Use and compare different IDEs to review tools they provide and understand the support they provided • Create a simple program to provide a user with information about high level and low level languages • Create a simple program to provide a user with information about the different tools IDEs provide	
Skills	• Creating new programs in IDLE • Create variables and constants to be later called upon • Importing libraries like time and random to use in their programs • How to code and use subprograms • How to code and use arrays • Save and open IDLE files • How to send files over email	• Understand the difference and support provided when using an IDE • Practice using high level languages when coding programs	
Unit of Work	**Producing Robust Programs**	**Paper 1 Recap**	
Fertile Questions	How can I create a program that is resistant to user misuse?	How can I learn independent revision skills to support my understanding of paper 1 content	
Key Concepts	• Users can accidentally or maliciously exploit input values in a program that can cause disruption to the program. Defensive design is used to ensure this doesn't happen. • Anticipating misuse- using input validation checks in a program after an input to ensure the user has followed the instruction. If not it causes the user to repeat the input until they do. • Authentication- confirming the identity of the user using a username and password to ensure they are accessing information they are entitled to. • Creating maintainable code so that when the programmer returns to it they remember where they are and what they were working on. This includes: ○ Use of subprograms – coding something once and then calling upon it throughout to avoid making mistakes or unnecessarily repeating yourself ○ Sensible names for variables and constants so others and the programmer can remember what they are storing	• No new concepts for this topic as we revisit content learnt in year 10	

	○ Indentation- To understand the flow of the program and when certain parts are skipped ○ Commenting- Allows the user or other programmers to understand the intention of certain parts of the code • Throughout programming testing should be occurring to ensure all parts of the program are running as intended. This includes two types of testing: ○ Iterative testing- testing as you go ○ Final testing- testing the program at the end • Through testing different errors will occur. These include: ○ Syntax errors- where the code language has been written incorrectly causing the program to break. These are easier to identify as it causes an error message ○ Logic errors- where the code runs but not as it was intended to by the programmer. These are harder to identify but using a trace table can support this. • Trace tables- used to follow the flow of a program and examine all values throughout the program. Helpful in identifying logic errors. • When testing a range of test data should be used to identify all possible errors. This includes: ○ Normal data- data you'd expect to be accepted in the program ○ Boundary data- data on the end of what is accepted and should still be accepted ○ Invalid data- data that is the correct type but outside the range of what is accepted and should be rejected by the program ○ Erroneous data- incorrect data type and should be rejected by the program		
Assessment	• Coded program with input validations and maintainability • 35 mark exam questions	• Mini mocks completed once a fortnight to review material on paper 1	
Experiences	• Taking previous coded game from a topic before and adding in input validations and maintainability to improve the program • Building test tables to record results of testing data • Fill in trace tables for programs to identify where errors have occurred	• Create a coded program that acts as a revision game about paper 1 content that you and others could use to revise • Students have reteach sessions each lesson and then have the remainder of the time to code and create their game	
Skills	• Building stronger coding practice to include maintainability in programs • Strengthening testing skills by using formal processes to record and track errors	• Developing practical coding skills useful for paper 2 whilst reviewing paper 1 content in an engaging way	