Curriculum Map: Computing, Year 12

	Autumn	Spring	Summer
Content Declarative knowledge 'I Know'	1.1.1 Structure and function of the processor 1.2.1 Operating Systems - 1 1.1.2 Types of processors 1.2.3 Introduction to programming 1.4.1 Data Types 2.2.1 Programming techniques 1.4.2 Data Structures	1.1.3 Input, output and storage 1.4.2 Data Structures 2.2.1 Programming techniques (b) Recursion (Year 13 topic) 2.3.1 Algorithms 2.2.2 Software Development **NEA – preparation starts 1.4.1 Data Types 1.2.1 Operating Systems- 2 2.1.1 Thinking abstractly 2.1.2 Thinking ahead 2.1.3 Thinking procedurally	**NEA – Analysis phase 1.2.2 Applications generation 1.3.1 Databases 2.1.4 Thinking logically 1.4.3 Boolean Algebra 1.3.2 Networks 1.3.3 Web Technologies 1.5.1 Computing related legislation 1.5.2 Ethical, moral and cultural **NEA- more prep 1.2.4 Types of Programming Language – OOP (Y13 topic) ** Software design using UML (extra to spec)
Skills Procedural Knowledge 'I know how to'	**1.1.1 Structure and function of the processor** (a) The Arithmetic and Logic Unit; ALU, Control Unit and Registers (Program Counter; PC, Accumulator. ACC, Memory Address Register; MAR, Memory Data Register; MDR, Current Instruction Register; CIR). Buses: data, address and control: how this relates to assembly language programs. (b) The fetch-decode-execute cycle, including its effect on registers. (c) The factors affecting the performance of the CPU, clock speed, number of cores, cache. (d) Von Neumann, Harvard and contemporary processor architecture. **1.2.1 Operating Systems** (a) The need for, function and purpose of operating systems. (b) Memory Management (paging, segmentation and virtual memory). (c) Interrupts, the role of interrupts and Interrupt Service Routines (ISR), role within the Fetch-Decode-Execute Cycle. (d) Scheduling: round robin, first come first served, multi-level feedback queues, shortest job first and shortest remaining time. (h) Virtual machines, any instance where software is used to take on the function of a machine including executing intermediate code or running an operating system within another. **1.1.2 Types of processors** (a) The differences between and uses of CISC and RISC processors.	**1.1.3 Input, output and storage** (a) How different input, output and storage devices can be applied to the solution of different problems. (b) The uses of magnetic, flash and optical storage devices. (c) RAM and ROM. (d) Virtual storage. **1.4.2 Data Structures** (b) The properties of stacks and queues. **2.2.1 Programming techniques** (b) Recursion, how it can be used and compares to an iterative approach (Year 13 topic) **2.3.1 Algorithms** (a) Analysis and design of algorithms for a given situation. (b) Standard algorithms (bubble sort, insertion sort, binary search and linear search). (c) Implement bubble sort, insertion sort. (d) Implement binary and linear search. (e) Representing, adding data to and removing data from queues and stacks. (f) Compare the suitability of different algorithms for a given task and data set **2.2.2 Software Development (a)** Understand the waterfall lifecycle, agile methodologies, extreme programming, the spiral model and rapid application development. (b) The relative merits and drawbacks of different methodologies and when they might be used. (c) Writing and following algorithms.	**** NEA- Analysis phase** • Finish Problem definition • Establish stakeholders • Finish research and investigation • Conduct fact finding • Prototype **1.2.2 Applications generation** (a) The nature of applications, justifying suitable applications for a specific purpose. (b) Utilities. (c) Open-source vs closed source. (d) Translators: Interpreters, compilers and assemblers. **1.3.1 Databases** (a) Relational database, flat file, primary key, foreign key, secondary key, entity relationship modelling. (b) Methods of capturing, selecting, managing and exchanging data. **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. **1.4.3 Boolean Algebra** (a) Define problems using Boolean logic. (b) Manipulate Boolean expressions, including the use of Karnaugh maps to simplify Boolean expressions. (c) Use logic gate diagrams and truth tables. **1.3.2 Networks** (a) Characteristics of networks and the importance of protocols and standards.

	(b) Multicore and Parallel systems. **1.2.3 Introduction to programming** (a) Procedural programming language techniques: •• program flow •• variables and constants •• procedures and functions •• arithmetic, Boolean and assignment operators •• string handling •• file handling. •• Assembly language (including following and writing simple programs with Little Man Computer). **1.4.1 Data Types** (a) Primitive data types, integer, real/floating point, character, string and Boolean. **2.2.1 Programming techniques** (a) Programming constructs: sequence, iteration, branching. (b) Global and local variables. (c) Modularity, functions and procedures, parameter passing by value and reference. (d) Use of an IDE to develop/debug a program. **1.4.2 Data Structures** (a) Arrays (of up to 3 dimensions), records, lists, tuples.	(d) Different test strategies, including black and white box testing and alpha and beta testing. (e) Test programs that solve problems using suitable test data and end user feedback, justify a test strategy for a given situation. ****NEA – preparation starts** • Submit proposal • Define requirements • Research and investigation **1.4.1 Data Types** (b) Represent positive integers in binary. (c) Use of sign and magnitude and two's complement to represent negative numbers in binary. (d) Addition and subtraction of binary integers. (e) Represent positive integers in hexadecimal. (f) Convert positive integers between binary hexadecimal and denary. (g) Positive and negative real numbers using normalised floating point representation. (h) How character sets (ASCII and UNICODE) are used to represent text. **1.2.1** (e) Distributed, embedded, multi-tasking, multi-user and real time operating systems. (f) BIOS. (g) Device drivers. **2.1.1 Thinking abstractly** (a) The nature of abstraction. (b) The need for abstraction. (c) The differences between an abstraction and reality. (d) Devise an abstract model for a variety of situations. **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. (c) The need for reusable program components. **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.	(b) Internet structure: •• The TCP/IP stack. •• DNS. •• Protocol layering. •• LANs and WANs. •• Packet and circuit switching. (c) Client-server and peer to peer **1.3.3 Web Technologies** (a) HTML, CSS and JavaScript. See AS appendix 5d. (b) Lossy v lossless compression. **1.5.1 Computing related legislation** (a) The Data Protection Act 1998. (b) The Computer Misuse Act 1990. (c) The Copyright Design and Patents Act 1988. (d) The Regulation of Investigatory Powers Act 2000. **1.5.2 Ethical, moral and cultural issues** (a) The individual moral, social, ethical and cultural opportunities and risks of digital technology: •• Computers in the workforce. •• Automated decision making. •• Artificial intelligence. •• Environmental effects. •• Censorship and the Internet. •• Monitor behaviour. •• Analyse personal information. •• Piracy and offensive communications. •• Layout, colour paradigms and character sets. ****NEA- more prep** **1.2.4 Types of Programming Language** (e) Object-oriented languages **(Y13 topic)** • Software design using UML **(extra to spec)**
Strategies Conditional Knowledge 'I know when to'	Which kind of devices would use Von Neumann architecture and which ones would use Harvard architecture? Which kind of devices would use RISC architecture and which ones would use CISC?	When to use stacks and when to use queues in program design? What are the scenarios where the different sorting and searching algorithms are applicable/suitable?	How Karnaugh maps are used to simplify Boolean expressions. Recommend client server or peer to peer architecture for given networking scenario.

	What are the design implications of using global variables? When to use static data structures as opposed to dynamic data structures? Deciding when to use casting to convert from one data type to another.	Be able to recommend / compare different methodologies for a given software problem. Be able to recommend / compare different types of OS for a given scenarios.	Able to consider the legal /moral/environmental implications of a new / existing computer system and/or the use of electronic devices.
Key Questions	What is the purpose and function of the core components of a processor? What is the role and components of the ALU? What is the purpose and function of registers within the processor, including the PC, accumulator, MAR, MDR and CIR? What is the purpose, function and role of the data, address, and control buses in the processor? How assembly language makes use of registers, and how data and addresses are transferred between registers? What is the purpose and stages within the FDE cycle? How and when are the registers used within this cycle, and how and where are data and addresses transmitted to/from in each part of this cycle. How the performance of the CPU can be affected by many factors. How and why the performance is affected by the clock speed, the number of cores and the size, speed and cache. What are the different approaches that the architectures take to storing instructions and data in memory and the benefits of each approach? How to control the flow of a program (sequence, iteration and selection). What is the purpose and function of both variables and constants? How to read, trace and write code that makes use of both variables and constants. What are the benefits of using constants over variables? What is the role of sub-programs (procedures and functions) in a program, how these can be used to reduce the amount of code and improved the efficiency? What are the differences between procedures and functions? How to read, write and trace programs using both procedures and functions.	What is the range of input, output and storage devices? Candidates do not need to understand how the input and output devices work but must be able to recommend appropriate devices for specific situations and be able to justify choices made. Understand that there are different types of storage device. Candidates need to know about the characteristics of each type (magnetic, optical and flash) and understand the benefits and drawbacks of each and be able to recommend an appropriate type of device for a given situation and justify the choice. Understand the purpose of ROM and RAM within a computer system, their characteristics, and the role they play in the running of a range of different computers e.g., mobile devices, embedded systems etc. Why there is a need for virtual storage, how virtual storage works and the benefits and drawbacks of using virtual storage. Why an operating system is required, along with the different tasks it performs within a computer system (e.g., resource management, file management, interrupt handling, security, providing a platform for software to run, providing a user interface and providing utilities). How operating systems manage memory. Candidates need to understand the need for, purpose and function of paging to divide memory into usable fixed-size pages and how these aid in the transfer of memory for example virtual memory. Candidates need to understand what is meant by segmentation and how memory is divided into segments to allow access to memory. Candidates need to understand what is meant by virtual memory and why this is needed in a computer system. Candidates need to understand how paging is used in virtual memory, and the benefits and drawbacks of having and using virtual memory in a computer system.	What is meant by a database. Candidates should be familiar with basic database terminology such as fields, records and tables. Candidates should know the difference between a flat file and a relational database, and be able to explain the benefits and limitations of each approach. Candidates should have experience of setting up and using both a flat file, and relational database. What is meant by a primary key, foreign key and secondary key and how each are used in a database. Candidates should be able produce and follow Entity Relationship (ER) diagrams which include 1:1, 1:M and M:M relationships. Candidates should be able to identify how tables should be linked. Discuss a range of methods for capturing data (such as forms, OCR, OMR and sensors) selecting data (such as Query By Example and SQL), managing data (such as changing data by manipulating it – e.g. arithmetic functions, adding, editing, deleting the data) and exchanging data (with common formats such as CSV, JSON and XML). Candidates won't be specifically asked about any one of these methods but may be asked to discuss/justify suitable methods as part of a more open question. Understand the purpose of applications, and should have knowledge and experience of a range of different application software (for example database, word processor, web browser, graphics manipulation etc.). Candidates should be able to recommend the use of specific and generic applications for given scenarios, justifying their use and function(s) for a scenario. Understand the purpose and role of utility software in a computer system. Candidates should be familiar with a range of utility software (e.g. disk defragmentation, file management, device driver, system clean-up, security etc.) Be able to explain the differences between open and closed source software, the benefits and drawbacks to creator and user of each of the licensing models and be able to recommend which is used (with justification) for a specific scenario.

	How to use a range of arithmetic (+, -, /, *, MOD, DIV) operators, Boolean (AND, OR, NOT, ==, >, <, =, >=, <=, !=) operators and assignment operator (=). How to read, trace and write programs using these operators. How to read, trace and write code using a range of string handling functions (selecting substrings, converting to upper/lowercase, converting between characters and their ASCII values. How to write programs that write to and read from text files. How to read/write/trace/amend simple programs in procedural languages. What is the purpose and need for assembly language? How to read, write, trace, and amend programs written in the Little Man Computer language. What are the differences between the CISC and RISC processors and the key features and benefits of each, and the relative benefits of each architecture? What is meant by a parallel system and the benefits and limitations of parallel processing. How parallel processing can be achieved through different (i.e. multiple processors in the same computer or distributed or multiple cores in a CPU or GPU). What are the benefits of a multicore system in terms of parallel processing and running multiple programs at the same time. How to use programming data types such as integer, real, Boolean, character, string etc., choosing appropriate data types for a situation or given data, and converting from one data type to another (casting) when required. Be able to describe what is meant by arrays (up to 3 dimensions), records, lists and tuples. Candidates are expected to be able recognise when they can be used and incorporate them in their programs to store data. Understand the purpose and use of a record structure to store data of different data types in a program. Candidates	Understand the purpose of interrupts within a computer system. Why an interrupt might be generated, and what happens within CPU and memory in order to call an interrupt service routine. Understand the need for scheduling of tasks by an operating system and the benefits that scheduling brings. Candidates need to understand that there are different scheduling algorithms, which each have benefits and drawbacks for tasks with specific characteristics. Understand how the following scheduling algorithms work; round robin, first come first served, multi-level feedback queue, shortest job first and shortest remaining time. What are the different (and often overlapping) classifications of operating systems (distributed, embedded, multi-tasking, multi-user and real time), including the key features of each. Candidates should be able to recommend (and justify) a type of operating system for a given scenario. What is the role of the BIOS in a computer system, and the steps that the BIOS goes through to start a computer? What is meant by 'device drivers' and why they are needed for communication between hardware and the operating system. Be able to describe what is meant by a virtual machine, how they can be used to execute intermediate code, how they can be used to run a software driven machine inside a physical machine and the benefits and drawbacks of each approach. How and why computers store data as binary, and that a binary number can have a variety of different interpretations depending on what is being stored (e.g. numeric, text, image, sound). Be able to convert positive whole numbers to binary and from binary to denary. How to store negative numbers using Sign and Magnitude and Two's Complement. Candidates should be able to convert denary numbers to sign and magnitude, and two's complement – and vice-versa.	Understand the need for translators when writing programs. Candidates need to have knowledge of the differences in operation of interpreters and compilers, from these they need to be able to assess the benefits and drawbacks of using each type, and recommend with justification which should be used in a specific scenario. Candidates need to understand the role of an assembler and how it differs from interpreters and compilers. Understand the definition and purpose of a network. Candidates need to understand the purpose of, and importance of using, protocols. Candidates should be able to discuss examples of protocols that may be used in a network/ the internet (but will not be asked to recall information about any specific protocol). Candidates should understand the term standard, and the purpose and need for standards in a network (or any situation where data is transferred). What is the purpose and benefits of layering protocols, particularly within the TCP/IP stack? Candidates need to know the different layers within the TCP/IP stack and the purpose of each. Candidates need to understand how data is transmitted on the Internet, the use of IP addresses and packets in the transfer of data. (NB: Candidates are not expected to be familiar with the OSI model). Candidates are expected to understand the terms LAN and WAN. Candidates need to understand how the Domain Name System is used to find the IP address of a URL. Candidates need to understand the purpose, function, benefits and drawbacks of both packet and circuit switching. Candidates need to understand the difference between a client-server and peer-to-peer network. Candidates need to know the benefits and drawbacks of each type of network and be able to recommend one for a given scenario. What is the purpose of HTML, CSS and JavaScript. Candidates need to know when each language/markup would be used, and what its purpose and function is. Candidates should have experience of writing webpages using HTML, CSS and JavaScript. Candidates need to be able to recognise the code in Appendix 5d, and be able to read,

	should have experience of using records to store, search, manipulate and retrieve data. Understand the purpose and use of a list to store data in a program. Candidates should have experience of using lists to store, search, manipulate and retrieve data. Understand the purpose and use of tuples to store data in a program. Candidates should have experience of using tuples to store, search, manipulate and retrieve data. Candidates need to understand the behaviour of stacks and queues (i.e. LIFO and FIFO). Be able to understand the constructs of sequence, iteration and branching. Candidates must be able to use these constructs independently of each other, and combine them to produce a solution. These include the selection statements of if (include elseif and else) and select case statements. These include both condition-based iteration (e.g. while, repeat until) and count controlled iteration (e.g. for) – as well as how condition based can be used as count controlled iteration. Be able to read code using these constructs, create code using these constructs and trace code (for example using a trace table). To understand the use and need for variables in a program, and must understand the difference, benefits and drawbacks of both global and local variables. Candidates must be able to recognise where local and global variables are used, and the impact that these have on the program, for example the amount of memory used by the program. Candidates need to understand how a program using global variables can be changed to use local variables – and vice-versa. Understand what is meant by modular code, and how this can be produced using functions and procedures. Candidates need to understand the differences between functions and procedures and how each is used within a program. Candidates need to be able to read, trace and write code using functions and procedures. Understand the purpose and use of parameters within a program, and how they are used in functions and procedures. Candidates will need to be able to read, trace and write code that makes use of parameters.	Be able to perform addition and subtraction on integer binary numbers. (These numbers could be positive or negative using two's complement representation.) Understand the purpose and potential uses of hexadecimal for example where and why they are used instead of binary and the benefits of using hexadecimal over alternatives such as binary. Candidates should be able to convert denary numbers to hexadecimal and vice-versa and from binary to hexadecimal and vice-versa. How (positive and negative) real numbers are represented in a binary floating-point representation, and should be able to convert between a denary number and a real binary number. (NB the representation used for the exam is the mantissa and exponent both represented using two's complement.) Candidates should understand the need for normalised floating-point numbers. Candidates should be able to normalise a floating-point number. How characters are represented in binary. Candidates should understand the need for a character set and how a computer makes use of a character set. Candidates should be aware of the ASCII and UNICODE character sets and be able to explain the differences between these and the benefits of each. Candidates should be able to use a character set, or part of a character set, to translate characters into binary and vice-versa. (Candidates are not expected to memorise any values in a character set) What is abstraction, it's purpose in the design and creation of computer programs. Candidates need to know about the benefits of abstraction and be able to apply these benefits to specific scenarios. Candidates may be given a scenario and be asked how abstraction can be applied to it, or how it has already been applied. Candidates need an understanding of the differences between reality and abstraction. Understand that situations require inputs and output, and that outputs can be both digital or in a hard copy	write, amend and interpret code using HTML, CSS and JavaScript. What is the need for compression (when transferring data over a network). Candidates need to understand the difference between lossy and lossless compression, and the benefits and drawbacks of each type. Candidates need to be able to recommend a type of compression for a given scenario. What are AND, OR, NOT and XOR gates? Candidates should be familiar with the logic of each Boolean operator, and the truth tables. Candidates should be able to construct logic gate diagrams from a Boolean expression and vice-versa. Candidates should be able to construct truth tables from Boolean expressions and logic gate diagrams. Understand that Boolean expressions can be simplified and should have experience of simplifying expressions using Karnaugh maps. Candidates should be able to create, complete and interpret Karnaugh maps to simplify Boolean expressions. What is the need for, and purpose of laws relating to the use of computers? What is the purpose and role of the Data Protection Act. Candidates will need to understand the different rules that are within the DPA and how these impact the use of computers and the storage of data by organisations. This should include what organisations can and cannot do. Candidates need to understand the purpose and principles of the Computer Misuse Act, including the actions that it prohibits. Understand the purpose and principles of the Copyright and Patents Act, including the actions that it prohibits. Understand the purpose and principles of the Regulation of Investigatory Powers Act, and what this allows in interception and monitoring of electronic communication. Understand how the regulations impact organisations and the use of computers and electronic communication. We are aware the law is constantly changing and some of the mentioned laws/acts (most notably the DPA) are likely to change over the course of the specification. Answers will be accepted that use an interpretation of the law based on

	Understand the difference between passing a parameter by value and by reference, they need to understand the benefits and drawbacks of each, recommending which should be used for a given situation. Candidates need to be able to read, trace and write code that makes use of parameters passed both by value and by reference. Understand how an IDE can be used to produce code, and understand the range of features and tools that are within an IDE that can be used to help produce and debug a program.	format. Candidates may be given a description, diagram, or code for a scenario, and they will need to demonstrate an understanding of what inputs and outputs are needed, and/or are used in that specific scenario. Be able to determine what else they need to know before they can produce a solution, for example what information is missing and what else will affect that solution. Understand the purpose, benefits and drawbacks of reusable program components. Candidates should understand how these components can be reused, and for a given scenario/program they will need to be able to identify the subprograms that will be needed. Candidates may then be required to write code for these reusable components. Be able to deconstruct a program and identify the component parts that will make it up, for example listing the parts or completing a structure diagram. Candidates may be given some component parts and be asked to complete these from a written description or pseudocode for a program. Be able to identify the steps that will need to take place to complete the algorithm or program, and be able to write these in a suitable format, or put a given list into the correct order to produce a working program. Candidates may need to write pseudocode or draw a flow chart to show this sequencing of steps. For a given scenario, candidates need to be able to identify where sub-procedures may be used, and then write appropriate pseudocode for these sub-procedures, making use of parameters where appropriate. Given a structure diagram, they will need to interpret, or complete to identify these sub-procedures. Understand that decisions are made within programs, and they need to be able to identify where these decisions will take place within an algorithm or program, and be able to understand what these decisions are and the impact of these decisions on the algorithm/program and the next (and final) outcomes	when the specification was started or when the examination was sat. Understand what is meant by moral, social, ethical and cultural issues in relation to the use of computers. Understand how the use of computers, and the increasing use of computers in the work force has moral, social, ethical and cultural implications and risks to a variety of people such as the employees, employers, society and organisations. Understand how the use of computers to make decisions automatically has moral, social, ethical and cultural implications and risks to a variety of people such as those people who make the decisions, the people the decisions affect, and the need for additional collection of information to ensure the decisions are accurate and valid. Understand how the development of artificial intelligence has moral, social, ethical and cultural impacts on a variety of people. How the environmental effects of computers (such as disposal, energy use) have moral, social, ethical and cultural implications. How the Internet and censorship on the Internet has moral, social, ethical and cultural implications. What are the moral, social, ethical and cultural implications of using computers to monitor behaviour (such as CCTV, tracking phone calls, GPS, monitoring emails)? What are the moral, social, ethical and cultural implications of using computers to analyse personal information (such as the gathering, storing and analysing of medical records) How different cultures impact on the use of and creation of computers and programs. How colours have different meanings in different cultures for example red means danger in one culture, and luck in another. Candidates need to consider how these will impact the creation of computer applications.

		from the algorithm/program. Candidates need to understand that there can be many different routes through a program, and understand how decisions influence these routes and outcomes. Understand the different models that can be followed to produce a program (explicitly the waterfall lifecycle, agile methodology, extreme programming, the spiral model and rapid application development). Candidates need to understand the tasks, processes, benefits and drawbacks of each model and the similarities and differences between each. Candidates need to understand where each model is most suitable to use, and be able to justify the use in a situation. Candidates need to be able to write algorithms using flow charts, pseudocode and/ or program code. Candidates need to be able to follow the code as shown in the OCR pseudocode guide, but are not expected to write code in this. Candidate's code is not expected to be syntactically correct, but must use appropriate code structures. Be able to use black box testing, white box testing, alpha testing and beta testing whilst producing their own programs. Candidates need to understand how each testing strategy can be used in a situation, and the benefits and drawbacks of each method, and apply this to a given situation to recommend appropriate testing strategies. Have experience of using suitable test data to test their own programs. Understand the use of test data and apply this to a given program. Understand how dry runs can be used in the development and testing of programs, and be able to use dry runs to test given code. Understand the need for and importance of end user feedback. Be able to write algorithms using flow charts, pseudocode and program code. Candidates need to be able to follow the code as shown in the OCR pseudocode guide, but are not expected to write code in this syntax. Candidate's code is not expected to be syntactically correct, but must use appropriate code structures.	

		Understand the need for standard sorting algorithms. Candidates need to understand how the sorting algorithms bubble and insertion work and the situations when each can, and cannot be used. Candidates need to be able to use the algorithms to sort data, and complete, write and correct algorithms to perform each sorting algorithm. Understand the need for standard searching algorithms. Candidates need to understand how the searching algorithms binary and linear work and the situations when each can, and cannot be used. Candidates need to be able to use the algorithms to search data sets for specific values that may, or may not exist in the data set. Candidates need to understand when each searching algorithm can, and cannot be used. Candidates need to be able to complete, write and correct algorithms to perform each searching algorithm. Experience of using the data structures stacks and queues. Candidates need to understand the differences and similarities between stacks and queues. Candidates need to be able to add and remove data from both stacks and queues. Candidates need to understand how pointers are used within stacks and queues. Candidates need to understand how stacks and queues can be implemented in a computer system, for example through the use of an array with pointers. Be able to read, correct and write algorithms to add and remove data items, and manipulate data items in a stack and queue. Understand how the choice of algorithm can be affected by the data set. Candidates need to understand the impact of specific algorithms on speed and memory use. Be aware of how and when a program can use more memory, or can take longer to run and be able to compare algorithms (not expected to know about Big O notation)	
Assessment topics	October baseline test, Dec progress test	Progress tests: Jan/Feb, April	Y12 PPE

Cross curricular links/Character Education	Maths	Maths, Electronics, Problem solving, Mirroring the practices in the real world of software development	Environmental concerns Business Studies/Economics English language Problem solving Resilience Mirroring the practices in the real world of software development