Year 12			***Curriculum Checkpoints: What do students know and what can they do?***			
Computer Science Comp 1			Developing	Securing	Mastering	Excelling
AF1	1.1 The characteristics of contemporary processors, input, output and storage devices	1.1.1 Structure and function of the processor	Know the names and definitions of all registers and components of the CPU. Explain the different factors affecting the performance of the CPU.	Explain the Fetch-Decode-Execute cycle using most registers and components in the correct order. Describes the process of pipelining and how it improves processor efficiency.	Explain in detail the Fetch-Decode-Execute cycle using all registers and components in the correct order. Described the differences between Von Neumann and Harvard architecture.	Explain in increasing complexity the Fetch-Decode-Execute cycle including components outside of the CPU to explain who all processing is carried out. Can justify when pipelining is appropriate to be used or not using real world examples. Evaluate the use of Von Neumann, Harvard and contemporary architecture and suggest the most appropriate for a scenario.
		1.1.2 Types of processor	Identify the features of a CISC and RISC processors. Explain the term GPU and its uses in processing graphics.	Explain the terms CISC and RISC and give some examples of their uses. Describe the term GPU and its uses, including those not related to graphics.	Explain the differences between a CISC and RISC processor. Explain the differences multicore and parallel systems. Describe the differences between a GPU and CPU and select appropriate processors for a computer system and its needs.	Explain and justify the appropriate use of a CISC or RISC processor for a given scenario. Explain and justify the appropriate use of a multicore or parallel system for a given scenario.
		1.1.3 Input, output and storage	Explain the terms input, output and storage. Describe the different storage technologies and be able to identify an number of devices for each technology. Explain the difference between RAM and ROM.	Identify appropriate input, output and storage devices required for a range of real world scenarios. Explain the advantages and disadvantages of the different storage technologies and how they would suit different user needs.	Identify and justify appropriate input, output and storage devices required for a range of real world scenarios. Explain the advantages and disadvantages of virtual storage and how it would suit different user needs.	Explain in detail the complex processing and tasks that take place within a computer system from the initial input of a user to the final output of a process and all the components required along the way. Explain in detail why the over reliance on virtual memory would require improvements in RAM.
AF2	1.2 Software and software development	1.2.1 Systems Software	Explain the purpose of an operating system and list the different types of operating systems. Explain the differences between paging and segmentation. List some of the scheduling algorithms use to process data and instructions. Explain what the BIOS and device drivers are.	Explain the different types of operating systems. Identify where paging and segmentation have been used in the memory management of a system. List all of the scheduling algorithms used to process data and instructions. Describe the stages of BIOS in order to turn a computer on. List some device drivers needed for a computer to work.	Explain what an interrupt and Interrupt Service Routine are. Explain the uses of paging, segmentation and virtual storage. Describe the scheduling algorithms used to process data and instructions. Justify why the BIOS needs to be stored on ROM rather than RAM. Explain what a device driver is and why they are needed in a computer system.	Explain in detail how an interrupt is detected during the Fetch-Decode-Execute cycle and the role of the Interrupt Service Routine in solving it. Explain what happens if additional interrupts arise while the Interrupt Service Routine is dealing with another interrupt. Describe the advantages and disadvantages of different operating systems and be able to suggest an appropriate one for a given scenario.
		1.2.2 Applications Generation	Explain the terms utilities, open source and close source software. Describe the 3 different types of translators and the coding language they would be used on. Identify the correct order of stages required to convert source code to machine code.	Explain a number of different utilities used by a computer system and their purpose. Explain the advantages and disadvantages of open source and closed source software. Explain the differences between an interpreter and compiler. Explain the terms linkers, loaders and libraries.	Suggest appropriate utilities to be used to improve a computer systems performance. Suggest if a piece of software should be distributed as open or closed source based on a set of requirements. Justify the most appropriate translator to use for a given scenario. Describe each stage required to covert source code into machine code.	Justify appropriate utilities to be used to improve a computer systems performance for a given scenario. Justify the most appropriate format for software to be distributed based on a given scenario. Describe how linkers, loaders and libraries are incorporated into the code to create the final executable file.
		1.2.3 Software Development	Name the 5 different software development models. Trace through an algorithm and identify what is happening in the algorithm.	Explain the 5 different software development models. Trace through algorithms and correct both syntax and logic errors they might contain. Write simple algorithms to solve mathematical or logic problems.	Describe the advantages and disadvantages of the different software development models. Trace through algorithms and complete unfinished code in order to achieve the desired outcome. Write algorithms to solve mathematical and logic problems.	Justify the most appropriate software development model for a given scenario. Write complex algorithms to solve increasingly complex real world problems making use of subroutines.

		1.2.4 Types of Programming Language	Explain the terms procedural language and object-oriented language. Identify the 4 different modes of addressing. Explain the functions of the 11 assembly language instructions used in LMC.	Demonstrate how to write code in either a procedural language or object-oriented language and identify the characteristics of that language. Describe the 4 different modes of addressing. Be able to read and create simple assembly language programmes in LMC	Demonstrate how to write code in both a procedural language and object-oriented language and identify the characteristics used in each language. Demonstrate how to use the 4 different modes of addressing on a data set. Be able to read, create and explain assembly language programmes in LMC which make use of some branching instructions.	Demonstrate how to write code in both a procedural language and object-oriented language and identify the characteristics used in each language. Explain why there are 4 different modes of addressing and how they can have an impact on the running of a program. Be able to read, create and explain assembly language programmes in LMC which make use of a loop instruction.
AF3	1.3 Exchanging data	1.3.1 Compression, Encryption and Hashing	Explain lossy and lossless compression. Explain run length and dictionary encoding. Explain what encryption is and why it is need. List different uses of the hashing algorithm.	Describe the differences between lossy and lossless compression. Explain run length and dictionary encoding and how they are effective as lossless compression methods. Explain what symmetric and asymmetric encryption are. Explain what hashing is and list some different uses.	Explain the most appropriate compression type for different file types and formats. Demonstrate how run length and dictionary encoding would be performed on a text file. Describe the differences between symmetric and asymmetric encryption. Explain what hashing is and list a number of different uses.	Justify the most appropriate compression type for a given scenario. Justify why either run length or dictionary encoding are the most appropriate format for compressing text files. Justify the appropriate use of either symmetric or asymmetric encryption for a given scenario. Explain the difference between encryption and hashing and justify which you would use in different situations.
		1.3.2 Databases	Explain basic database terms, for example; entity, attribute, flat file, relational, etc. List a number of data capture methods for a database. Attempts to normalise a database using the normalisation rules. Knows the structure of an SQL statement. Has an understanding of transaction processing and can explain record locking.	Explain database terms in some detail with the addition of primary key, secondary key, and normalisation. Attempt to create an entity relationship diagram. Explain a number of data capture methods for a database and how they work. Normalise a database to 1NF using the normalisation rules. Produce a simple SQL statement to find information in a database. Explain transaction processing making reference to record locking and redundancy.	Explain database terms in detail with the addition of entity relationship modelling. Create an accurate entity relationship diagram for 3 given tables. Justify the most appropriate data capture method for a given scenario. Normalise a database to 2NF using the normalisation rules. Produce a SQL statement to find information in a database and print out only select attributes in an order. Explain transaction processing making reference to record locking, redundancy and ACID.	Explain in detail all database terminology and be able to justify why databases are used to store data. Create an accurate entity relationship diagram for a scenario which requires additional tables to be added. Normalise a database to 3NF using the normalisation rules. Produce SQL statements that can search, add, edit, and delete records within a database. Explain transaction processing making reference to record locking, redundancy and ACID.
		1.3.3 Networks	Explain the different hardware required set up a network and connect to the internet. List the different network topologies. Explain the difference between a LAN and a WAN. List different network security threats and one prevention method for each threat. Explain the characteristics of a client-server and peer to peer network.	Draw and annotate a network diagram for a given scenario explaining each piece of hardware and its use. Explain the different network topologies. Justify the most appropriate network type for a given scenario. Explain different network security threats and one prevention method for each threat. Explain the difference between packet switching and circuit switching.	Suggest an appropriate network structure including the topology and all hardware for a given scenario. Explain all network security threats in detail and a number prevention methods for each threat. Describe the difference between a client-server and peer to peer network. Describe the roles of the four layers in the TCP/IP protocol stack. Explain the terms domain name and IP address and how domain names are organised.	Justify an appropriate network structure including the topology and all hardware for a given scenario. Explain and justify appropriate security threats and their preventions for a given scenario. Describe the advantages and disadvantages of client-server and peer to peer networks. Explain the importance of network protocols and standards and be able to suggest the most appropriate protocol to be used in difference scenarios. Explain the purpose and function of the Domain Name System.

		1.3.4 Web Technologies	Understand the term HTML and its role in the World Wide Web. Recall some basic HTML tags and their functions. Understand how webpages are indexed by a search engine.	Explain how HTML and CSS are used to create webpages. Recall most HTML tags and their functions and be able to add inline CSS formatting. Explain the process of how webpages are indexed by a search engine.	Explain how HTML, CSS and JavaScript are used to create webpages. Recall all HTML tags and their functions and be able to add external CSS formatting. Explain the PageRank algorithm and how it is used by search engines. Identify the different uses of client- and server-side processing.	Explain in increasing detail how HTML, CSS and JavaScript are used to create webpages. Accurately recreate the source code of a webpage from seeing it in the browser. Interpret and apply the PageRank algorithm to a given scenario. Describe which is the most appropriate processing, with client- or server-side, for a given scenario.
AF4	1.4 Data types, data structures and algorithms	1.4.1 Data Types	List and define primitive data types. Attempt to covert between binary, hexadecimal and denary, however there are still some errors. Use sign and magnitude to represent negative numbers in binary. Describe what ASCII and Unicode are.	Explain all primitive data types and why they are needed. Covert between binary, hexadecimal and denary. Represent fractions in fixed point binary. Use sign and magnitude and two's complement to represent negative numbers in binary. Perform binary addition on binary integers. Explain why there is more than one character set and the differences between ASCII and Unicode.	Perform binary subtraction using the method of two's complement and binary addition on the number to be subtracted. Represent positive and negative numbers with a fraction in float point form. Explain underflow and overflow and when they might occur. Perform logical, arithmetic and circular shifts on binary data. Explain the AND,OR and XOR bitwise operations.	Normalise un-normalised floating point numbers with both positive and negative mantissas. Add and subtract floating point numbers. Perform bitwise operations AND, OR and XOR. Use appropriate masks to manipulate bits
		1.4.2 Data Structures	Explain the concept of data structures. Describe the concept of an abstract data type. Describe how a list may be implemented as either a static or dynamic structure. Explain the concept and uses of a stack. Describe a hash table and its uses. Explain the typical uses for graphs.	Describe arrays up to 3 dimensions, tuples and records. Explain the concept and uses of a queue. Show how items may be added to or deleted from a list. Describe the creation and maintenance of data with in a stack. Apply the hashing algorithm to data sets. Explain the terms graph, weighted graph, vertex/node, edge/arc, undirected graph and directed graph. Define a binary tree as a rooted tree.	Describe the creation and maintenance of data within a queue (linear, circular, priority). Describe the linked list data structure. Describe and apply the following operations to a stack: push, pop, peek, test for empty and test for full. Explain what is meant by a collision and how collisions are handled using rehashing. Explain how an adjacency matrix and an adjacency list may be used to represent a graph. Create and traverse a binary tree.	Describe and apply the following to the three types of queue; adding an item, removing an item, testing for an empty queue, test for a full queue. Show how to create, traverse add data to and remove data from a linked list. Explain how a stack frame is used with subroutine calls to store return addresses, parameters and local variables. Explain the concept of a dictionary and be able to apply it to a data set. Compare the use of adjacency matrices and adjacency lists. Create, search and traverse a binary search tree.
		1.4.3 Boolean Algebra	Construct a truth table for a variety of logic gates. Draw and interpret logic gate circuit diagrams involving multiple gates. Know there are a number of rules to follow when simplifying Boolean expressions. Explain what a Karnaugh map is used for. Complete the groupings on a given Karnaugh map.	Compare a truth table for a given logic gate circuit. Write a Boolean expression for a given logic circuit. Explain most of the rules for simplifying Boolean expressions. Create a Karnaugh map for a Boolean expression with 2 variables. Recognise and trace the logic of a half adder and full adder circuit.	Draw a logic gate circuit for a given Boolean expression. Explain all of the rules for simplifying Boolean expressions and begin to use the rules to simplify some Boolean expressions. Create a Karnaugh map for a Boolean expression with 3 variables. Construct the circuit for a flip-flop and explain why they are used.	Define problems using Boolean Logic. Can use all the rules of simplification to accurately simplify Boolean expressions. Create a Karnaugh map for a Boolean expression with 4 variables. Explain the use of the edge-triggered D-type flip-flop as a memory unit.
AF5	1.5 Legal, moral, cultural and ethical issues	1.5.1 Computing related legislation	Understand developments in technology have had a big impact on companies and individuals and how they can use technology.	Explain the four main computing related legislation; The Data Protection Act 1998, The Computer Misuse Act 1990, The Copyright Designs and Patents Act 1988 and The Regulation of Investigatory Powers Act 2000.	Compare the different computer related legislation and identify where there might be overlaps in them. Explain the penalties for breaking each of the four computing related legislation.	Explain and justify the computer related legislation broken, and the possible penalties, for a given scenario.
		1.5.2 Moral and ethical Issues	Explain the terms ethical, moral, environmental and cultural issues and be able to link them to technology use.	Explain how digital technology has impacted the work place and the impacts it has had for employers, employees and the environment. Explain what is meant by the term censorship and what this has to do with the internet.	Evaluate the use of artificial intelligence and the positive and negative impacts of it on society and the workforce. Discuss the cultural opportunities and impacts of technology on monitoring behaviour and analysing personal data.	Evaluate all of ethical, moral, environmental and cultural issues for a given digital technology scenario and give a balanced argument for how this might be solved.
Computer Science Comp 2			Developing	Securing	Mastering	Excelling

AF1	2.1 Elements of computational thinking	2.1.1 Thinking abstractly	Understand the nature of and need for abstraction. Define the term abstraction. Explain the need of abstraction in a given situation. E.g. saves memory.	Describe the differences between an abstraction and reality in a given scenario. Be able to identify features that have been abstracted or can be abstracted form a real model.	Create an abstract model for a variety of situations. Can assess the effectiveness of abstracted features of a solution.	Implement a solution containing only relevant details of the problem ignoring anything unnecessary.
		2.1.2 Thinking ahead	Identify the inputs and outputs for a given scenario. Know the processes that will take place in a given scenario.	Determine the preconditions for devising a solution to a problem. Examine the affect of these preconditions and the impacts they may have on solution development.	Assess the need of reusable components. Describe the advantages of using reusable components within a solution.	Understand the nature, benefits and drawbacks of caching. Assess how caching compares with other concepts such as concurrent processing.
		2.1.3 Thinking procedurally	Identify the components of a problem. Identify the components of a solution to a problem.	Determine the order of the steps needed to solve a problem.	Know how to develop a solution of a problem after identifying all components of a problem.	Identify sub-procedures necessary to solve a problem and the importance of them within a solution.
		2.1.4 Thinking logically	Identify the points in a solution where a decision has to be taken.	Determine the logical conditions that affect the outcome of a decision.	Determine how decisions affect flow through a program	Apply logical thinking techniques in problem solving within a given scenario. Identify decisions, inputs, outputs and how these may affect the flow of a program.
		2.1.5 Thinking concurrently	Define the term concurrent processing. Give a suitable example of how concurrent processing works.	Explain how concurrent processing can take place in a given scenario, determining which parts of a program can be tackled at the same time.	Determine the benefits are trade-offs of concurrent processing in a given scenario.	Evaluate and compere the use of concurrent processing with other concepts such as caching, searching & sorting algorithms.
AF2	2.2 Problem solving and programming	2.2.1 Programming techniques	Define and explain the three programming constructs. Define the terms local and global variables and identify when used within a program. Can define and identify parameters being passed. Define and identify where iteration & recursion has been used.	Define IDE tools and explain how they can be used to help a programmer develop a solution. Can differentiate between general and debugging IDE tools. Can define and determine if parameters are being passed by value or by reference. Describe the pros and cons of using iteration & recursion.	Understand what is meant by modularity and how to implement within a program. Know the benefits of using sub programs. Explain how procedural programming and OOP compere. Know how to implement sub programs within Python. Compere iteration & recursion and identify which would be suitable to use in a given scenario.	Explain the use of all OOP techniques and be able to write pseudocode using OOP, including; classes, objects, methods, attributes, inheritance, instantiation encapsulation and polymorphism. Can write OOP using the constructor method. Have practical coding experience of OOP, iteration & recursion. Can trace a program including recursion.
		2.2.2 Computational methods	Know what features of a problem make it solvable by computational methods. Categorise different types of problems and solutions. Define what is meant by abstraction, decomposition and problem recognition.	Know the correct definitions of computational methods; Backtracking, data mining, heuristics, performance modelling, pipelining and visualisations. Be able to use abstraction, decomposition and problem recognition in a given scenario.	Understand the concept and application of the "divide and conquer" approach. Apply all computational methods in a given scenario.	Evaluate the effectiveness of each computational method. Apply and explain how computational methods are implemented within other concepts such as; algorithms and programming.
AF3	2.3 Algorithms	2.3.1 Algorithms	Analyse the suitability of different algorithms for a given scenario and data set. Define constant, linear, polynomial, exponential and logarithmic functions. Know how to perform searching and sorting algorithms on data.	Perform the Dijkstra's shortest path algorithm and the A* algorithm. Use big O Notation to compare the time and complexity of algorithms. Trace searching and sorting algorithms. Perform the push() and pop() methods on stack and queue data structures	Know how to traverse a graph using breath and depth first traversal. Trace breath and depth first algorithms. Know how to implement a linked list in a given scenario. Complete and traverse a binary tree. Know what is meant by pre-order, in-order and post-order traversal.	Know how to implement all sorting and searching algorithms in Python & Pseudocode. Know how to implement data structures such as stacks & queues in Python & Pseudocode.