Qualification content

Topic 1: Problem solving

Students are expected to develop a set of computational thinking skills that enable them to understand how computer systems work, and design, implement and analyse algorithms for solving problems.

Students should be given repeated opportunities to tackle computational problems of various sorts, including some substantial problem-solving tasks.

Subject content	Students should:
1.1 Algorithms	1.1.1 Understand what an algorithm is, what algorithms are used for and be able to interpret algorithms [flowcharts, pseudocode, structured English, written descriptions, program code]*
	1.1.2 Be able to create an algorithm to solve a particular problem, making use of programming constructs [sequence, selection, repetition] and using an appropriate notation [flowchart, written description, program code]
	1.1.3 Be able to describe the purpose of a given algorithm and explain how a simple algorithm works
	1.1.4 Be able to identify the correct output of an algorithm for a given set of data
	1.1.5 Be able to identify and correct errors in algorithms
	1.1.6 Be able to code an algorithm into a high-level language
	1.1.7 Understand how the choice of algorithm is influenced by the data structure and data values that need to be manipulated
	1.1.8 Understand how standard algorithms [quick sort, bubble sort, selection sort, linear search, binary search, breadth first search, depth first search, maximum/minimum, mean, count] work
	1.1.9 Understand factors that affect the efficiency of an algorithm
1.2 Decomposition	1.2.1 Be able to analyse a problem, investigate requirements [inputs, outputs, processing, initialisation] and design solutions
	1.2.2 Be able to decompose a problem into smaller sub-problems

* See *Appendix 2* for key definitions related to this content.

Topic 2: Programming

Learning to program is a core component of a computer science course. Students should be competent at reading and writing programs and be able to reason about code. They must be able to apply their skills to solve real problems and produce robust programs.

Students should be given repeated opportunities to develop and practise their programming skills.

Subject content	Students should:
2.1 Develop code	2.1.1 Be able to write programs in a high-level programming language
	2.1.2 Understand the benefit of producing programs that are easy to read, and be able to use techniques [comments, descriptive variable names, indentation] to improve readability and to explain how the code works
	2.1.3 Be able to differentiate between types of error in programs [logic, syntax, runtime]
	2.1.4 Be able to design and use test plans and test data
	2.1.5 Be able to interpret error messages and identify, locate and fix errors in a program
	2.1.6 Be able to identify what value a variable will hold at a given point in a program [trace table]
	2.1.7 Be able to make effective use of tools offered in an integrated development environment [watcher, break points, single-step, step-throughs]
	2.1.8 Be able to evaluate the strengths and weaknesses of a program and suggest improvements
	2.1.9 Be able to work safely, respectfully, responsibly and securely when using computers
2.2 Constructs	2.2.1 Be able to identify the structural components of a program [variable and type declarations, initialisations, command sequences, conditionals, repetition, data structures, subprograms]
	2.2.2 Be able to use sequencing, selection and repetition constructs in their programs
2.3 Data types and structures	2.3.1 Understand the need for and be able to select and use data types [integer, real, Boolean, char]
	2.3.2 Understand the need for and be able to select and use data structures [one-dimensional arrays, two-dimensional arrays]
	2.3.3 Understand the need for and be able to manipulate strings
	2.3.4 Understand the need for and be able to use variables and constants
	2.3.5 Understand the need for and be able to use global and local variables

Subject content	Students should:
2.4 Input/output	2.4.1 Be able to write code that accepts and responds appropriately to user input
	2.4.2 Understand the need for and be able to implement validation
	2.4.3 Be able to write code that outputs information to a screen and understand and use Cartesian x/y coordinates
	2.4.4 Be able to design and code a user interface [textual, graphical]
	2.4.5 Be able to write code that opens/closes, reads/writes, deletes, inserts, appends from/to a file
2.5 Operators	2.5.1 Understand the purpose of and be able to use arithmetic operators [plus, minus, divide, multiply, modulus, integer division]
	2.5.2 Understand the purpose of and be able to use relational operators [equal to, less than, greater than, not equal to, less than or equal to, greater than or equal to]
	2.5.3 Understand the purpose of and be able to use Boolean operators [AND, OR, NOT]
2.6 Subprograms	2.6.1 Understand the benefits of using subprograms and be able to write code that uses user-written and pre-existing [built-in, library] subprograms
	2.6.2 Understand the concept of passing data into and out of subprograms [procedures, functions, return values]
	2.6.3 Be able to create subprograms that perform generalisation

Topic 3: Data

Computers are able to store and manipulate large quantities of data. They use binary to represent different types of data.

Students are expected to learn how different types of data are represented in a computer. They should be given the opportunity to gain practical experience of using SQL to handle data stored in a structured database.

Subject content	Students should:
3.1 Binary	3.1.1 Understand that computers use binary to represent data and instructions
	3.1.2 Understand how computers represent and manipulate numbers [unsigned integers, signed integers (sign and magnitude, Two's complement) real numbers]
	3.1.3 Be able to convert between binary and denary whole numbers (0-255) and vice versa
	3.1.4 Be able to perform binary arithmetic [add, subtract, multiply] and understand the concept of overflow
	3.1.5 Understand why hexadecimal notation is used and be able to convert between hexadecimal and binary and vice versa
3.2 Data representation	3.2.1 Understand how computers encode characters [ASCII, Unicode]
	3.2.2 Understand how bitmap images are represented in binary [pixels, resolution, colour depth]
	3.2.3 Understand how analogue data [sound, temperature, light intensity] is represented in binary
	3.2.4 Understand the limitations of binary representation of data [quantisation, sampling frequency] and how bit length constrains the range of values that can be represented
3.3 Data storage and Compression	3.3.1 Understand and be able to convert between the terms 'bit, nibble, byte, kilobyte (KB), megabyte (MB), gigabyte (GB), terabyte (TB)'
	3.3.2 Understand the need for data compression and methods of compressing data [lossless, lossy] and that JPEG and MP3 are examples of lossy algorithms
	3.3.3 Understand how a lossless, run-length encoding [RLE] algorithm works
	3.3.4 Understand that file storage is measured in bytes and that data transmission is measured in bits per seconds, and be able to calculate the time required to transmit a file and storage requirements for files
3.4 Encryption	3.4.1 Understand the need for data encryption
	3.4.2 Understand how a Caesar cipher algorithm works

Subject content	Students should:
3.5 Databases	3.5.1 Understand the characteristics of structured and unstructured data
	3.5.2 Understand that data can be decomposed and organised in a structured database [tables, records, fields, relationships, keys]
	3.5.3 Understand the need for and be able to use SQL statements *

* See *Appendix 2* for key definitions related to this content.

Topic 4: Computers

Students must be familiar with the hardware and software components that make up a computer system and recognise that computers come in all shapes and sizes from embedded microprocessors to distributed clouds.

Students should be given the opportunity to gain practical experience of interpreting instructions written in assembly language.

Subject content	Students should:
4.1 Machines and computational models	4.1.1 Understand the concept of a computer as a hardware machine or as a virtual machine
	4.1.2 Understand that there is a range of computational models [sequential, parallel, multi-agent]
	4.1.3 Understand the input-process-output model
4.2 Hardware	4.2.1 Understand the function of hardware components of a computer system [processor (CPU), memory, secondary storage, input devices, output devices] and how they work together
	4.2.2 Understand the concept of a stored program and the role of components of the processor [control unit (CU), arithmetic/logic unit (ALU), registers, clock, address bus, data bus] in the fetch-decode-execute cycle
	4.2.3 Understand the function of assembly code and be able to interpret a block of assembly code using a given set of commands*
	4.2.4 Understand how data is stored on physical devices [magnetic, optical, solid state]
	4.2.5 Understand how microcontrollers can be programmed to control actuators and take input from sensors
4.3 Logic	4.3.1 Be able to construct truth tables for a given logic statement [AND, OR, NOT]
	4.3.2 Be able to produce logic statements for a given problem
4.4 Software	4.4.1 Understand what an operating system is and the functions of an operating system [file management, input/output, resource allocation, process management, network management, user management]
	4.4.2 Understand that application software such as a web browser, word processor, spreadsheet or apps are computer programs
	4.4.3 Understand how software can be used to simulate and model aspects of the real world and be able to create software models
4.5 Programming languages	4.5.1 Understand what is meant by high-level and low-level programming languages and assess their suitability for a particular task
	4.5.2 Understand what is meant by a compiler and an interpreter

* See *Appendix 2* for key definitions related to this content.

Topic 5: Communication and the internet

Computer networks and the internet are now ubiquitous. Many computer applications in use today would not be possible without networks. Students should understand the key principles behind the organisation and of computer networks. Ideally, they should be able to experiment by setting up a simple network.

Students should be given the opportunity to gain practical experience of creating web pages.

Subject content	**Students should:**
5.1 Networks	5.1.1 Understand why computers are connected in a network
	5.1.2 Understand the different types of networks [LAN, WAN, PAN, VPN]
	5.1.3 Understand the network media [copper cable, fibre optic cable, wireless]
	5.1.4 Understand that network data speeds are measured in bits per second [Mbps, Gbps]
	5.1.5 Understand the role of and need for network protocols
	5.1.6 Understand that data can be transmitted over networks using packets [TCP/IP]
	5.1.7 Understand the need to detect and correct errors in data transmission [check sums]
	5.1.8 Understand the concept of and need for network addressing and host names [MAC addresses]
	5.1.9 Understand characteristics of network topologies [bus, ring, star, mesh]
5.2 The internet and the world wide web	5.2.1 Understand what is meant by the internet and how the internet is structured [IP addressing, routers, connecting backbone, domain names]
	5.2.2 Understand what is meant by the world wide web (WWW) and components of the WWW [web server URLs, ISP, HTTP, HTTPS, HTML]
	5.2.3 Be able to use HTML and CSS to construct web pages [formatting, links, images, media, layout, styles, lists]
	5.2.4 Understand the client-server model, the difference between client-side and server-side processing and the role of cookies

Topic 6: The bigger picture

Students should be aware of emerging trends in computing technology and recognise that computing has an impact on nearly every aspect of the world in which they live.

Subject content	Students should:
6.1 Emerging trends, issues and impact	6.1.1 Be aware of current and emerging trends in computing technology [quantum computing, DNA computing, artificial intelligence (AI), nano technology]
	6.1.2 Be aware of the impact of computing on individuals, society and the environment
	6.1.3 Be aware of ethical and legal issues arising from the use of computers
	6.1.4 Be aware of ownership issues relating to computing [intellectual property, patents, licensing, open source and proprietary software]