## Topic 1: Problem solving

| **2013 specification** | |  | **2016 specification** | |
| --- | --- | --- | --- | --- |
| **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, pseudo-code, structured English, written descriptions, program code) |  | **1.1 Algorithms** | 1.1.1 understand what an algorithm is, what algorithms are used for and be able to interpret algorithms (flowcharts, pseudo- code, 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) |  | 1.1.2 understand how to create an algorithm to solve a particular problem, making use of programming constructs (sequence, selection, iteration) and using appropriate conventions (flowchart, pseudo-code, written description, draft program code) |
|  | 1.1.3 be able to describe the purpose of a given algorithm and explain how a simple algorithm works |  | 1.1.3 understand the purpose of a given algorithm and how an algorithm works |
|  | 1.1.4 be able to identify the correct output of an algorithm for a given set of data |  | 1.1.4 understand how to determine the correct output of an algorithm for a given set of data |
|  | 1.1.5 be able to identify and correct errors in algorithms |  | 1.1.5 understand how to identify and correct errors in algorithms |
|  | 1.1.6 be able to code an algorithm into a high-level language |  | 1.1.6 understand how to code an algorithm in 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 |  | 1.1.7 understand how the choice of algorithm is influenced by the data structures 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 |  |  | 1.1.8 understand how standard algorithms (bubble sort, merge sort, linear search, binary search) work |
|  | 1.1.9 understand factors that affect the efficiency of an algorithm |  |  | 1.1.9 be able to evaluate the fitness for purpose of algorithms in meeting specified requirements efficiently using logical reasoning and test data |
| **1.2**  **Decomposition** | 1.2.1 be able to analyse a problem, investigate requirements (inputs, outputs, processing, initialisation) and design solutions |  | **1.2 Decomposition and abstraction** | 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 |  |  | 1.2.2 be able to decompose a problem into smaller sub-problems |
|  |  |  |  | 1.2.3 understand how abstraction can be used effectively to model aspects of the real world |
|  |  |  |  | 1.2.4 be able to program abstractions of real-world examples |

# Topic 2: Programming

| **2013 specification** | |  | **2016 specification** | |
| --- | --- | --- | --- | --- |
| **Subject content:** | |  | **Students should:** | |
| **2.1 Develop code** | 2.1.1 be able to write programs in a high-level programming language |  | **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 |  | 2.1.2 understand the benefit of producing programs that are easy to read and be able to use techniques (comments, descriptive names, variables, constants, subprograms, 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) |  | 2.1.3 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 |  | 2.1.4 be able to design and use test plans and test data (normal, boundary, erroneous) |
|  | 2.1.5 be able to interpret error messages and identify, locate and fix errors in a program |  | 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) |  | 2.1.6 be able to determine 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) |  | 2.1.7 be able to determine the strengths and weaknesses of a program and suggest improvements |
|  | 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 Constructs** | 2.2.1 understand the structural components of a program (variable and type declarations, command sequences, selection, iteration, data structures, subprograms) |
|  | 2.2.2 be able to use sequencing, selection and repetition constructs in their programs |  |  | 2.2.2 be able to use sequencing, selection and iteration 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 Data types and structures** | 2.3.1 understand the need for, and understand how to 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) |  |  | 2.3.2 understand the need for, and understand how to use, data structures (records, one-dimensional arrays,  two-dimensional arrays) |
|  | 2.3.3 understand the need for, and be able to manipulate, strings |  |  | 2.3.3 understand the need for, and how to manipulate, strings |
|  | 2.3.4 understand the need for, and be able to use, variables and constants |  |  | 2.3.4 understand the need for, and how to use, variables and constants |
|  | 2.3.5 understand the need for, and be able to use, global and local variables |  |  | 2.3.5 understand the need for, and how to use, global and local variables when implementing subprograms |
| **2.4**  **Input/output** | 2.4.1 be able to write code that accepts and responds appropriately to user input |  | **2.4 Input/output** | 2.4.1 understand how to write code that accepts and responds appropriately to user input |
|  | 2.4.2 understand the need for, and be able to implement, validation |  |  | 2.4.2 understand the need for, and how 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 |  |  | 2.4.3 be able to write code that reads/writes from/to a text file |
|  | 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 Operators** | 2.5.1 understand the purpose of, and how to use, arithmetic operators (add, subtract, 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) |  |  | 2.5.2 understand the purpose of, and how 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) |  |  | 2.5.3 understand the purpose of, and how to use, logic 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 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) |  |  | 2.6.2 understand the concept of passing data into and out of subprograms (procedures, functions) |
|  | 2.6.3 be able to create subprograms that perform generalisation |  |  | 2.6.3 be able to create subprograms that use parameters |

# Topic 3: Data

| **2013 specification** | |  | **2016 specification** | |
| --- | --- | --- | --- | --- |
| **Subject content:** | | **Students should:** | |
| **3.1 Binary** | 3.1.1 understand that computers use binary to represent data and instructions | **3.1 Binary** | 3.1.1 understand that computers use binary to represent data (numbers, text, sound, graphics) and program instructions |
|  | 3.1.2 understand how computers represent and manipulate numbers (unsigned integers, signed integers [sign and magnitude, two’s complement] real numbers) |  | 3.1.2 understand how computers represent and manipulate numbers (unsigned integers, signed integers [sign and magnitude, two’s complement]) |
|  | 3.1.3 be able to convert between binary and denary whole numbers (0-255) and vice versa |  | 3.1.3 be able to convert between binary and denary whole numbers (0–255) |
|  | 3.1.4 be able to perform binary arithmetic (add, subtract, multiply) and understand the concept of overflow |  | 3.1.4 understand how to perform binary arithmetic (add, shifts [logical and arithmetic]) 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 |  | 3.1.5 understand why hexadecimal notation is used and be able to convert between hexadecimal and binary |
| **3.2 Data representation** | 3.2.1 understand how computers encode characters (ASCII, Unicode) | **3.2 Data representation** | 3.2.1 understand how computers encode characters using ASCII |
|  | 3.2.2 understand how bitmap images are represented in binary (pixels, resolution, colour depth) |  | 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 |  |  | 3.2.3 understand how sound, an analogue signal, 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 |  |  | 3.2.4 understand the limitations of binary representation of data (sampling frequency, resolution) when constrained by the number of available bits |
| **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 Data storage and compression** | 3.3.1 understand how 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 |  |  | 3.3.2 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 |  |  | 3.3.3 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 |  |  | 3.3.4 understand that file storage is measured in bytes and be able to calculate file sizes |
| **3.4 Encryption** | 3.4.1 understand the need for data encryption |  | **3.4 Encryption** | 3.4.1 understand the need for data encryption |
|  | 3.4.2 understand how a Caesar cipher algorithm works |  |  | 3.4.2 understand how a Caesar cipher algorithm works |
| **3.5 Databases** | 3.5.1 understand the characteristics of structured and unstructured data |  | **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) |  |  | 3.5.2 understand that data can be decomposed, organised and managed in a structured database (tables, records, fields, relationships, keys) |
|  | 3.5.3 understand the need for and be able to use SQL statements |  |  |  |

# Topic 4: Computers

| **2013 specification** | |  | **2016 specification** | |
| --- | --- | --- | --- | --- |
| **Subject content:** | |  | **Students should:** | |
| **4.1 Machines and computational modelling** | 4.1.1 understand the concept of a computer as a hardware machine or as a virtual machine |  | **4.1 Machines and computational modelling** | 4.1.1 understand the input-process-output model |
|  | 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 Hardware** | 4.2.1 understand the function of the hardware components of a computer system (CPU, main memory, secondary storage, input and 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 |  | 4.2.2 understand the function of different types of main memory (RAM, ROM, cache) |
|  | 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 |  | 4.2.3 understand the concept of a stored program and the role of components of the CPU (control unit [CU], arithmetic/logic unit [ALU], registers, clock, address bus, data bus, control bus) in the fetch-decode-execute cycle (the Von Neumann model) |
|  | 4.2.4 understand how data is stored on physical devices (magnetic, optical, solid state) |  | 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 |  |  | 4.2.5 understand the concept of storing data in the ‘cloud’ and other contemporary secondary storage |
|  |  |  |  | 4.2.6 understand the need for embedded systems and their functions |
| **4.3 Logic** | 4.3.1 be able to construct truth tables for a given logic statement (AND, OR, NOT) |  | **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 |  |  | 4.3.2 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 Software** | 4.4.1 know what an operating system is and how it manages files, processes, hardware and the user interface |
|  | 4.4.2 understand that application software such as a web browser, word processor, spreadsheet or apps are computer programs |  |  | 4.4.2 understand the purpose and functions of utility software (managing, repairing and converting files; compression; defragmentation; backing up; anti-virus, anti-spyware) |
|  | 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 |  |  | 4.4.3 understand how software can be used to simulate and model aspects of the real world |
| **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 Programming languages** | 4.5.1 understand what is meant by high-level and low-level programming languages and understand their suitability for a particular task |
|  | 4.5.2 understand what is meant by a compiler and an interpreter |  |  | 4.5.2 understand what is meant by an assembler, a compiler and an interpreter when translating programming languages and know the advantages and disadvantages of each |

# Topic 5: Communication and the internet

| **2013 specification** | |  | **2016 specification** | |
| --- | --- | --- | --- | --- |
| **Subject content:** | |  | **Students should:** | |
| **5.1 Networks** | 5.1.1 understand why computers are connected in a network |  | **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) |  | 5.1.2 understand the different types of networks (LAN, WAN) and usage models (client-server, peer-to-peer) |
|  | 5.1.3 understand the network media (copper cable, fibre optic cable, wireless) |  | 5.1.3 understand wired and wireless connectivity |
|  | 5.1.4 understand that network data speeds are measured in bits per second (MBps, GBps) |  | 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 |  | 5.1.5 understand the role of, and need for, network protocols (Ethernet, Wi-Fi, TCP/IP, HTTP. HTTPS, FTP, email [POP3, SMTP, IMAP]) |
|  | 5.1.6 understand that data can be transmitted over networks using packets (TCP/IP) |  | 5.1.6 understand that data can be transmitted in packets using layered protocol stacks (TCP/IP) |
|  | 5.1.7 understand the need to detect and correct errors in data transmission (check sums) |  | 5.1.7 understand characteristics of network topologies (bus, ring, star, mesh) |
|  | 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 Network security** | 5.2.1 understand the importance of network security and be able to use appropriate validation and authentication techniques (access control, physical security and firewalls) |
|  |  |  |  | 5.2.2 understand security issues associated with the ‘cloud’ and other contemporary storage |
|  |  |  |  | 5.2.3 understand different forms of cyberattack (based on technical weaknesses and behaviour) including social engineering (phishing, shoulder surfing), unpatched software, USB devices, digital devices and eavesdropping |
|  |  |  |  | 5.2.4 understand methods of identifying vulnerabilities including penetration testing, ethical hacking, commercial analysis tools and review of network and user policies |
|  |  |  |  | 5.2.5 understand how to protect software systems from cyberattacks, including considerations at the design stage, audit trails, securing operating systems, code reviews to remove code vulnerabilities in programming languages and bad programming practices, modular testing and effective network security provision |
| **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) |  | **5.3 The internet and the world wide web** | 5.3.1 understand what is meant by the internet and how the internet is structured (IP addressing, routers) |
|  | 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) |  |  | 5.3.2 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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **2013 specification** | |  | **2016 specification** | |
| **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], nanotechnology) |  | **6.1 Emerging trends, issues and impact** | 6.1.1 understand the environmental impact of technology (health, energy use, resources) on society |
|  | 6.1.2 be aware of the impact of computing on individuals, society and the environment |  | 6.1.2 understand the ethical impact of using technology (privacy, inclusion, professionalism) on society |
|  | 6.1.3 be aware of ethical and legal issues arising from the use of computers |  | 6.1.3 understand the legal impact of using technology (intellectual property, patents, licensing, open source and proprietary software, cyber-security) on society |
|  | 6.1.4 be aware of ownership issues relating to computing (intellectual property, patents, licensing, open source and proprietary software) |  |  |