TAUNTON SCHOOL

Year 10 Syllabus in a nutshell

COMPUTER SCIENCE

Year 10 Syllabus in a nutshell – Computer Science

Computer science is the study of the foundational principles and practices of computation and computational thinking and their application in the design and development of computer systems. Learning computational thinking involves learning to program, that is to write computer code, because this is the means by which computational thinking is expressed. IGCSE Computer Science enables learners to develop an interest in computing and to gain confidence in computational thinking and programming. They develop their understanding of the main principles of problem-solving using computers. Learners apply their understanding to develop computer-based solutions to problems using algorithms and a high-level programming language. They also develop a range of technical skills, as well as the ability to test effectively and to evaluate computing solutions. This qualification will help learners appreciate current and emerging computing technologies and the benefits of their use. They learn to recognise the ethical issues and potential risks when using computers. IGCSE Computer Science is an ideal foundation for further study in Computer Science. Understanding the principles of Computer Science provides learners with the underpinning knowledge required for many other subjects in science and engineering, and the skills learnt can also be used in everyday life.

Syllabus:

https://www.cambridgeinternational.org/programmes-and-qualifications/cambridge-igcse-9-1-computer-science-0984/

Text Book:

Cambridge iGCSE Computer Science Coursebook. Sarah Lawrey And Donald Scott

Year 10 Theory (2 Single Lessons Each Week)

	Autumn 1	**Autumn 2**	**Spring 1**	**Spring 2**	**Summer 1**	**Summer 2**
Theme	Data Representation	Communication And Internet Technologies	Hardware	Input and Output Devices	Software And Security	Ethics
Syllabus Reference	1.1	1.2	1.3	1.3	1.4	1.5
Assessment	Data Representation Assessment Test	Communication And Internet Technologies Assessment Test	Hardware Assessment Test	Input and Output Devices Assessment Test	Software A Security Assessment Test	Ethics Assessment Test
Text Book Reference	Chapter 1	Chapter 2	Chapters 3, 6, 7, 8	Chapters 5, 7, 8	Chapter 2 a 9	Chapter 10

Year 10 Practical (1 Double Lesson Each Week)

	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
Topic	Practical Programming					
Software To Be Used And Skills To Be Developed	Python					
Assessment	Python Practical Programming Assessments					

Year 11 Theory (2 Single Lessons Each Week)

	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
Theme	Algorithm design and programming	System Design And Databases	Revision	Revision	Exam Practice	
Syllabus Reference	2.1	2.2				
Assessment	Algorithms Assessment	System Design And Databases Assessment				
Text Book Reference	Chapters 11, 12, 13	Chapter 15				

Year 11 Practical (1 Double Lesson Each Week)

	Autumn 1	Autumn 2	Spring 1	Spring 2	Summer 1	Summer 2
Theme	Problem Solving and Programming					
Software To Be Used And Skills To Be Developed	Python					
Assessment	Problem Solving And Programming	Problem Solving And Programming	Problem Solving And Programming Developing Solutions For Pre Release Materials	Problem Solving And Programming Developing Solutions For Pre Release Materials	Exam	

Data representation	Binary systems • recognise the use of binary numbers in computer systems • convert positive denary integers into binary and positive binary integers into denary (a maximum of 16 bits will be used) • show understanding of the concept of a byte and how the byte is used to measure memory size • use binary in computer registers for a given application (such as in robotics, digital instruments and counting systems) Hexadecimal • represent positive numbers in hexadecimal notation • show understanding of the reasons for choosing hexadecimal notation to represent numbers • convert positive hexadecimal integers to and from denary (a maximum of four hexadecimal digits will be required) • convert positive hexadecimal integers to and from binary (a maximum of 16 bit binary numbers will be required) • represent numbers stored in registers and main memory as hexadecimal • identify current uses of hexadecimal numbers in computing, such as defining colours in Hypertext Markup Language (HTML), Media Access Control (MAC) addresses, assembly languages and machine code, debugging Data storage • show understanding that sound (music), pictures, video, text and numbers are stored in different formats • identify and describe methods of error detection and correction, such as parity checks, check digits, checksums and Automatic Repeat reQuests (ARQ) • show understanding of the concept of Musical Instrument Digital Interface (MIDI) files, JPEG files, MP3 and MP4 files • show understanding of the principles of data compression (lossless and

	lossy) applied to music/video, photos and text files
Communication and internet technologies	**Data transmission** • show understanding of what is meant by transmission of data • distinguish between serial and parallel data transmission • distinguish between simplex, duplex and half-duplex data transmission • show understanding of the reasons for choosing serial or parallel data transmission • show understanding of the need to check for errors • explain how parity bits are used for error detection • show understanding of the use of serial and parallel data transmission, in Universal Serial Bus (USB) and Integrated Circuit (IC) **Security aspects** • show understanding of the security aspects of using the Internet and understand what methods are available to help minimise the risks • show understanding of the Internet risks associated with malware, including viruses, spyware and hacking • explain how anti-virus and other protection software helps to protect the user from security risks **Internet principles of operation** • show understanding of the role of the browser • show understanding of the role of an Internet Service Provider (ISP) • show understanding of what is meant by hypertext transfer protocol (http and https) and HTML • distinguish between HTML structure and presentation • show understanding of the concepts of MAC address, Internet Protocol (IP) address, Uniform Resource Locator (URL) and cookies
Hardware and software	**Logic gates** • use logic gates to create electronic circuits

	• understand and define the functions of NOT, AND, OR, NAND, NOR and XOR (EOR) gates, including the binary output produced from all the possible binary inputs (all gates, except the NOT gate, will have 2 inputs only) • draw truth tables and recognise a logic gate from its truth table • recognise and use the following standard symbols used to represent logic gates: ○ NOT AND OR NAND NOR XOR • produce truth tables for given logic circuits, • produce a logic circuit to solve a given problem or to implement a given written logic statement **Computer architecture and the fetch-execute cycle** • show understanding of the basic Von Neumann model for a computer system and the stored program concept (program instructions and data are stored in main memory and instructions are fetched and executed one after another) • describe the stages of the fetch-execute cycle, including the use of registers and buses **Input devices** • describe the principles of operation (how each device works) of these input devices: 2D and 3D scanners, barcode readers, Quick Response (QR) code readers, digital cameras, keyboards, mice, touch screens, interactive whiteboards, microphones • describe how these principles are applied to real-life scenarios, for example: scanning of passports at airports, barcode readers at supermarket checkouts, and touch screens on mobile devices • describe how a range of sensors can be used to input data into a computer system, including light, temperature, magnetic field, gas, pressure, moisture, humidity, pH and motion

- describe how these sensors are used in real-life scenarios, for example: street lights, security devices, pollution control, games, and household and industrial applications

Output devices

- describe the principles of operation of the following output devices: inkjet, laser and 3D printers; 2D and 3D cutters; speakers and headphones; actuators; flat-panel display screens, such as Liquid Crystal Display (LCD) and Light-Emitting Diodes (LED) display; LCD projectors and Digital Light Projectors (DLP)
- describe how these principles are applied to real-life scenarios, for example: printing single items on demand or in large volumes; use of small screens on mobile devices

Memory, storage devices and media

- show understanding of the difference between: primary, secondary and off-line storage and provide examples of each, such as:
- primary: Read Only Memory (ROM) and Random Access Memory (RAM) secondary: hard disk drive (HDD) and Solid State Drive (SSD);
- off-line: Digital Versatile Disc (DVD), Compact Disc (CD), Blu-ray disc, USB flash memory and removable HDD
- describe the principles of operation of a range of types of storage device and media including magnetic, optical and solid state
- describe how these principles are applied to currently available storage solutions, such as SSDs, HDDs, USB flash memory, DVDs, CDs and Blu-ray discs
- calculate the storage requirement of a file

Operating systems

- describe the purpose of an operating system (Candidates will be required to understand the purpose and function of an operating system and why it is needed. They will not be required to

	understand how operating systems work.) • show understanding of the need for interrupts High- and low-level languages and their translators • show understanding of the need for both high-level and low-level languages • show understanding of the need for compilers when translating programs written in a high-level language • show understanding of the use of interpreters with high-level language programs • show understanding of the need for assemblers when translating programs written in assembly language
Security	**Security** • show understanding of the need to keep data safe from accidental damage, including corruption and human errors • show understanding of the need to keep data safe from malicious actions, including unauthorised viewing, deleting, copying and corruption • show understanding of how data are kept safe when stored and transmitted, including: • use of passwords, both entered at a keyboard and biometric • use of firewalls, both software and hardware, including proxy servers • use of security protocols such as Secure Socket Layer (SSL) and Transport Layer Security (TLS) • use of symmetric encryption (plain text, cypher text and use of a key) showing understanding that increasing the length of a key increases the strength of the encryption • show understanding of the need to keep online systems safe from attacks including denial of service attacks, phishing, pharming • describe how the knowledge from above can be applied to real-life scenarios including, for example, online banking, shopping
Ethics	**Ethics**

	• show understanding of computer ethics, including copyright issues and plagiarism • distinguish between free software, freeware and shareware • show understanding of the ethical issues raised by the spread of electronic communication and computer systems, including hacking, cracking and production of malware

Algorithm design and problem-solving	**Problem-solving and design** • show understanding that every computer system is made up of sub-systems, which in turn are made up of further sub-systems • use top-down design, structure diagrams, flowcharts, pseudocode, library routines and sub-routines • work out the purpose of a given algorithm • explain standard methods of solution • suggest and apply suitable test data • understand the need for validation and verification checks to be made on input data (validation could include range checks, length checks, type checks and check digits) • use trace tables to find the value of variables at each step in an algorithm • identify errors in given algorithms and suggest ways of removing these errors • produce an algorithm for a given problem (either in the form of pseudocode or flowchart) • comment on the effectiveness of a given solution **Pseudocode and flowcharts** • understand and use pseudocode for assignment, using ▯ • understand and use pseudocode, using the following conditional statements: • IF … THEN … ELSE … ENDIF • CASE … OF … OTHERWISE … ENDCASE • understand and use pseudocode, using the following loop structures: • FOR … TO … NEXT REPEAT … UNTIL • WHILE … DO … ENDWHILE

	• understand and use pseudocode, using the following commands and statements: • INPUT and OUTPUT (e.g. READ and PRINT) • totalling (e.g. Sum = Sum + Number) counting (e.g. Count = Count + 1) • understand and use standard flowchart symbols to represent the above statements, commands and structures
Programming	**Programming concepts** • declare and use variables and constants • understand and use basic data types: Integer, Real, Char, String and Boolean • understand and use the concepts of sequence, selection, repetition, totalling and counting • use predefined procedures/functions **Data structures; arrays** • declare and use one-dimensional arrays, for example: A[1:n] • show understanding of the use of one-dimensional arrays, including the use of a variable as an index in an array • read or write values in an array using a FOR … TO … NEXT loop
Databases	**Databases** • define a single-table database from given data storage requirements • choose and specify suitable data types • choose a suitable primary key for a database table • perform a query-by-example from given search criteria