# KS4 Computer Science – Crucial Knowlegde Glossary

| No. | Spec | Section | Sub-topic | Term | Definition |
|---|---|---|---|---|---|
| 1. | **1.1.1** | 1.1 Systems architecture | Architecture of the CPU | CPU | **Central Processing Unit:** "The main part of the computer (the brain), consisting of the registers, ALU and control unit." |
| 2. | **1.1.1** | 1.1 Systems architecture | Architecture of the CPU | Fetch-execute cycle | "The complete process of retrieving an instruction from store, decoding it and carrying it out. Also known as the instruction cycle." |
| 3. | **1.1.1** | 1.1 Systems architecture | Architecture of the CPU | ALU | **Arithmetic Logic Unit:** "Performs calculations e.g. x = 2 + 3 and logical comparisons e.g. IF x > 3 in the CPU." |
| 4. | **1.1.1** | 1.1 Systems architecture | Architecture of the CPU | CU | **Control Unit:** "Decodes instructions. Sends signals to control how data moves around the CPU." |
| 5. | **1.1.1** | 1.1 Systems architecture | Architecture of the CPU | Cache | "Memory in the processor providing fast access to frequently used instructions and data." |
| 6. | **1.1.1** | 1.1 Systems architecture | Architecture of the CPU | Register | "Tiny areas of extremely fast memory located in the CPU normally designed for a specific purpose, where data or control information is stored temporarily e.g. the MAR, MDR etc." |
| 7. | **1.1.1** | 1.1 Systems architecture | Architecture of the CPU | Von Neumann architecture | "Traditional computer architecture that forms the basis of most digital computer systems. Instructions are fetched, decoded and executed one at a time." |
| 8. | **1.1.1** | 1.1 Systems architecture | Architecture of the CPU | MAR | **Memory Address Register:** "Holds the address of data ready for use by the memory data register, or the address of an instruction passed from the program counter. Step 2 of the fetch, decode, execute cycle." |
| 9. | **1.1.1** | 1.1 Systems architecture | Architecture of the CPU | MDR | **Memory Data Register:** "Holds the data fetched from or to be written to the memory. Step 3 of the fetch, decode, execute cycle." |
| 10. | **1.1.1** | 1.1 Systems architecture | Architecture of the CPU | Program Counter | "Holds the address of the next instruction to be executed. Step 1 of the fetch, decode, execute cycle." |
| 11. | **1.1.1** | 1.1 Systems architecture | Architecture of the CPU | Accumulator | "Holds the result of calculations." |
| 12. | **1.1.2** | 1.1 Systems architecture | CPU performance | Clock speed | "Measured in Hertz, the clock speed is the frequency at which the internal clock generates pulses. The higher the clock rate, the faster the computer may work. The "clock" is the electronic unit that synchronizes related components by generating pulses at a constant rate." |
| 13. | **1.1.2** | 1.1 Systems architecture | CPU performance | Cache size | "A part of the main store between the central processor and the rest of the memory. It has extremely fast access, so sections of a program and its associated data are copied there to take advantage of its short fetch cycle. The larger the size of the cache the more that can be copied and stored here without having to go back to slower main memory (RAM), this has a significant impact on the speed of processing." |

| | | | | | |
|---|---|---|---|---|---|
| 14. | **1.1.2** | 1.1 Systems architecture | CPU performance | Cores | "A part of a multi-core processor. A multi-core processor is a single component with two or more independent actual CPUs, which are the units responsibly for the fetch-decode-execute cycle." |
| 15. | **1.1.3** | 1.1 Systems architecture | Embedded systems | Embedded system | "A computer which has been built to solve a very specific program and is not easily changed. For example the operating system placed inside a washing machine, microwave or set of traffic lights." |
| 16. | **1.2.1** | 1.2 Memory and storage | Primary storage (Memory) | Primary storage | "At GCSE level you can think of primary storage comprising of Random Access Memory (RAM) and Read Only Memory (ROM). It holds data and instructions which the CPU can much more easily and quickly access than from secondary storage devices." |
| 17. | **1.2.1** | 1.2 Memory and storage | Primary storage (Memory) | RAM | **Random Access Memory:** "Volatile (data lost when power is off) Read and write. Purpose: temporary store of currently executing instructions and their data. E.g. applications and the operating system in use." |
| 18. | **1.2.1** | 1.2 Memory and storage | Primary storage (Memory) | ROM | **Read Only Memory:** "Non-volatile (data retained when power is off) Read only. Purpose: stores instructions for starting the computer called the bootstrap." |
| 19. | **1.2.1** | 1.2 Memory and storage | Primary storage (Memory) | Virtual memory | "Using part of the hard disk as if it were random access memory. Allows more applications to be open than physical memory could hold." |
| 20. | **1.2.2** | 1.2 Memory and storage | Secondary storage | Secondary storage | "Permanent storage of instructions and data not in use by the processor. Stores the operating system, applications and data not in use. Read/write and non-volatile." |
| 21. | **1.2.2** | 1.2 Memory and storage | Secondary storage | Optical storage | "CD/R, CD/RW, DVD/R, DVD/RW Use: music, films and archive files. Low capacity. Slow access speed. High portability. Prone to scratches. Low cost." |
| 22. | **1.2.2** | 1.2 Memory and storage | Secondary storage | Magnetic storage | "Hard disk drive. Use: operating system and applications. High capacity. Medium data access speed. Low portability (except for portable drives). Reliable but not durable. Medium cost." |
| 23. | **1.2.2** | 1.2 Memory and storage | Secondary storage | Solid state storage | "Memory cards & solid state hard drive (SSD). Use: digital cameras and smartphones. Medium capacity. High portability. Reliable and durable. No moving parts. Fast data access speed. High cost." |
| 24. | **1.2.2** | 1.2 Memory and storage | Secondary storage | Storage capacity | "The amount of data a storage device is able to store. " |
| 25. | **1.2.2** | 1.2 Memory and storage | Secondary storage | Storage speed | "The read/write access speed of a storage device." |
| 26. | **1.2.2** | 1.2 Memory and storage | Secondary storage | Storage portability | "How easy it is to transport a given storage medium. E.g. Solid state and optical storage and designed to be highly portable, whereas more traditional magnetic storage is designed to stay in place." |
| 27. | **1.2.2** | 1.2 Memory and storage | Secondary storage | Storage durability | "How resistant to damage and wear a tear a storage device is. Devices with low durability will wear out easily over time." |
| 28. | **1.2.2** | 1.2 Memory and storage | Secondary storage | Storage reliability | "A relative measure of how confidant you can be that a given storage device will correctly allow you to write, read, delete and modify data." |

| | | | | | |
|---|---|---|---|---|---|
| 29. | **1.2.2** | 1.2 Memory and storage | Secondary storage | Storage cost | “The relative price of a storage device e.g. per Megabyte of data” |
| 30. | **1.2.3** | 1.2 Memory and storage | Units | Bit | “The smallest unit of storage in a computer system, represented by either a binary 1 or 0.” |
| 31. | **1.2.3** | 1.2 Memory and storage | Units | Nibble | “Half a byte / 4 bits.” |
| 32. | **1.2.3** | 1.2 Memory and storage | Units | Byte | “A collection of eight bits.” |
| 33. | **1.2.3** | 1.2 Memory and storage | Units | Kilobyte | “1 Kilobyte (KB) is 1024 Bytes. For the purpose of calculations in an exam you can assume 1000.” |
| 34. | **1.2.3** | 1.2 Memory and storage | Units | Megabyte | “1 Megabyte (MB) is 1024 Kilobytes (KB). For the purpose of calculations in an exam you can assume 1000.” |
| 35. | **1.2.3** | 1.2 Memory and storage | Units | Gigabyte | “1 Gigabyte (GB) is 1024 Megabytes (MB). For the purpose of calculations in an exam you can assume 1000.” |
| 36. | **1.2.3** | 1.2 Memory and storage | Units | Terabyte | “1 Terabyte (TB) is 1024 Gigabytes (GB). For the purpose of calculations in an exam you can assume 1000.” |
| 37. | **1.2.3** | 1.2 Memory and storage | Units | Petabyte | “1 Petabyte (PB) is 1024 Terabytes (TB). For the purpose of calculations in an exam you can assume 1000.” |
| 38. | **1.2.4** | 1.2 Memory and storage | Data storage (Numbers) | Denary numbers | “A numerical system of notation which uses 10 as its base. The 10 Decimal base digits are 0-9.” |
| 39. | **1.2.4** | 1.2 Memory and storage | Data storage (Numbers) | Binary numbers | “Binary describes a numbering scheme in which there are only two possible values for each digit: 0 and 1. The term in computing refers to any digital encoding system in which there are exactly two possible states. E.g. in memory, storage, processing and communications, the 0 and 1 values are sometimes called “low” and “high”, respectively.” |
| 40. | **1.2.4** | 1.2 Memory and storage | Data storage (Numbers) | Binary arithmetic | “The process of adding together two of more positive 8-bit binary numbers (0-255).” |
| 41. | **1.2.4** | 1.2 Memory and storage | Data storage (Numbers) | Overflow | “The generation of a number that is too large to be represented in the device meant to store it.” |
| 42. | **1.2.4** | 1.2 Memory and storage | Data storage (Numbers) | Hexadecimal | “A numerical system of notation which uses 16 rather than 10 as its base. The 16 Hex base digits are 0-9 and the letters A-F.” |
| 43. | **1.2.4** | 1.2 Memory and storage | Data storage (Numbers) | Binary shifts | “Allows you to easily multiple and divide base-2 binary numbers. A left shift multiplies by 2 and a right shift divides by 2. |
| 44. | **1.2.4** | 1.2 Memory and storage | Data storage (Characters) | Character set | “The set of symbols that may be represented in a computer at a particular time. These symbols, called characters, can be letters, digits, spaces or punctuations marks, the set includes control characters.” |
| 45. | **1.2.4** | 1.2 Memory and storage | Data storage (Characters) | ASCII | “America Standard Code for Information Interchange: “A character set devised for early telecommunication systems but proved to be ideal for computer systems. ASCII codes use 7-bits giving 32 control codes and 96 displayable characters (the 8th bit is often used for error checking).” |

| | | | | | |
|---|---|---|---|---|---|
| 46. | **1.2.4** | 1.2 Memory and storage | Data storage (Characters) | Unicode | “Standard character set that replaces the need for all the different character sets. It incorporates characters from almost all the world’s languages. It is a 16-bit extension of ASCII.” |
| 47. | **1.2.4** | 1.2 Memory and storage | Data storage (Images) | Pixels | “A pixel is the smallest unit of a digital image or graphic that can be displayed and represented on a digital display device. A pixel is represented by a dot or square on a computer monitor display screen.” |
| 48. | **1.2.4** | 1.2 Memory and storage | Data storage (Images) | Metadata | “A set of data that describes and gives information about other data.” |
| 49. | **1.2.4** | 1.2 Memory and storage | Data storage (Images) | Colour depth | “Also known as bit depth, is either the number of bits used to indicate the colour of a single pixel, in a bitmapped image or video frame buffer, or the number of bits used for each colour component of a single pixel.” |
| 50. | **1.2.4** | 1.2 Memory and storage | Data storage (Images) | Resolution | “The number of pixels (individual points of colour) contained on a display monitor, expressed in terms of the number of pixels on the horizontal axis and the number on the vertical axis.” |
| 51. | **1.2.4** | 1.2 Memory and storage | Data storage (Images) | Image quality | “The overall detail of an image, this is affected by the colour depth and resolution.” |
| 52. | **1.2.4** | 1.2 Memory and storage | Data storage (Images) | Image file size | “The file size of an image is increased when either its resolution (width & height in pixels) or its colour depth (number of bits needed to store a single pixel) increases.”<br>File size of an image = colour depth x image height (px) x image width (px) |
| 53. | **1.2.4** | 1.2 Memory and storage | Data storage (Sound) | Sample rate | “The number of samples taken per second, measured in Hertz (Hz).” |
| 54. | **1.2.4** | 1.2 Memory and storage | Data storage (Sound) | Sample duration | “How many seconds of audio the sound file contains.” |
| 55. | **1.2.4** | 1.2 Memory and storage | Data storage (Sound) | Sample bit depth | “The number of bits available to store each sample e.g. 16-bit” |
| 56. | **1.2.4** | 1.2 Memory and storage | Data storage (Sound) | Playback quality | “The finished quality of the digital sound file. This is effected by the sample rate and bit-depth. The higher the number the better the quality. The higher the number the larger the file size. CD quality is 44,100 samples per second.” |
| 57. | **1.2.4** | 1.2 Memory and storage | Data storage (Sound) | Sound file size | “The overall size of a sound file is found by the following formula:<br>Sample rate x duration (s) x bit depth” |
| 58. | **1.2.5** | 1.2 Memory and storage | Compression | Compression | “The process of reducing the size of a file in terms of its storage size.” |
| 59. | **1.2.5** | 1.2 Memory and storage | Compression | Lossy compression | “A compression scheme where their generally involves a loss of resolution in parts of the image where experiences shows that it will be least noticed.” |
| 60. | **1.2.5** | 1.2 Memory and storage | Compression | Lossless compression | “A compression scheme that allows the original images to be recreated.” |
| 61. | **1.3.1** | 1.3 Computer networks, connections and protocols | Networks and topologies | LAN | **Local Area Network:** “Small geographic area. All the hardware for the LAN is owned by the organisation using it. Wired with UTP cable, fibre optic cable or wireless using routers and Wi-Fi access points.” |
| 62. | **1.3.1** | 1.3 Computer networks, connections and protocols | Networks and topologies | WAN | **Wide Area Network:** “Small geographic area. All the hardware for the LAN is owned by the organisation using it. Wired with UTP cable, fibre optic cable or wireless using routers and Wi-Fi access points.” |

| | | | | | |
|---|---|---|---|---|---|
| 63. | **1.3.1** | 1.3 Computer networks, connections and protocols | Networks and topologies | Client-server network | “A client makes requests to the server for data and connections. A server controls access and security to one shared file store. A server manages access to the internet, shared printers and email services.  A server runs a backup of data.” |
| 64. | **1.3.1** | 1.3 Computer networks, connections and protocols | Networks and topologies | Peer-to-peer network | “All computers are equal. Computers serve their own files to each other. Each computer is responsible for its own security and backup. Computers usually have their own printer.” |
| 65. | **1.3.1** | 1.3 Computer networks, connections and protocols | Networks and topologies | Wireless access point | “A networking hardware device that allows a Wi-Fi device to connect to a wired network.” |
| 66. | **1.3.1** | 1.3 Computer networks, connections and protocols | Networks and topologies | Router | “A router sends data between networks. It is needed to connect a local area network to a wide area network. It uses the IP address on a device to route traffic to other routers.” |
| 67. | **1.3.1** | 1.3 Computer networks, connections and protocols | Networks and topologies | Switch | “A switch sends data between computers on a local area network. It uses the NIC address on a device to route traffic.” |
| 68. | **1.3.1** | 1.3 Computer networks, connections and protocols | Networks and topologies | NIC | **Network Interface Card/Controller:** “A computer hardware component that connects a computer to a computer network.” |
| 69. | **1.3.1** | 1.3 Computer networks, connections and protocols | Networks and topologies | Transmission media | “The physical media over which data is transmitted, e.g. twisted copper cable, fibre optic etc. ” |
| 70. | **1.3.1** | 1.3 Computer networks, connections and protocols | Networks and topologies | The Internet | “The Internet is a worldwide collection of interconnected computer networks.  It is an example of a WAN, albeit the very largest one which exists!” |
| 71. | **1.3.1** | 1.3 Computer networks, connections and protocols | Networks and topologies | DNS | **Domain Name System:** “The Internet’s equivalent of a phone book.  They maintain a directory of domain names and translate them to Internet Protocol (IP) addresses.  This is necessary because, although domain names are easy for people to remember, computers or machines access websites based on IP addresses.” |
| 72. | **1.3.1** | 1.3 Computer networks, connections and protocols | Networks and topologies | Hosting | “Websites stored on dedicated servers. Reasons include: Websites need to be available 24/7. Accessed by thousands of users at a time. Strong protection from hackers. They need an IP address that doesn’t change.” |
| 73. | **1.3.1** | 1.3 Computer networks, connections and protocols | Networks and topologies | The Cloud | “Remote servers that store data that can be accessed over the internet. Advantages: Access anytime, anywhere from any device. Automatic backup. Collaborate on files easily.” |
| 74. | **1.3.1** | 1.3 Computer networks, connections and protocols | Networks and topologies | Web server | “A program that uses HTTP (Hypertext Transfer Protocol) to serve the files that form Web pages to users, in response to their requests, which are forwarded by their computers' HTTP clients. Dedicated computers and appliances may be referred to as Web servers as well.” |
| 75. | **1.3.1** | 1.3 Computer networks, connections and protocols | Networks and topologies | Client | “A client can be thought of as computing device which requests or is using the services from some remote / connected server.” |
| 76. | **1.3.1** | 1.3 Computer networks, connections and protocols | Networks and topologies | Network topology | “The physical or logical arrangement of connected devices on a network e.g. Computers, switches, routers, printers, servers etc.” |

| | | | | | |
|---|---|---|---|---|---|
| 77. | **1.3.1** | 1.3 Computer networks, connections and protocols | Networks and topologies | Star topology | “Computers connected to a central switch. If one computer fails no others are affected. If the switch fails all connections are affected.” |
| 78. | **1.3.1** | 1.3 Computer networks, connections and protocols | Networks and topologies | Mesh topology | “Switches (LAN) or routers (WAN) connected so there is more than one route to the destination. e.g. The Internet More resilient to faults but more cable needed.” |
| 79. | **1.3.2** | 1.3 Computer networks, connections and protocols | Wired and wireless networks, protocols and layers | Wired connection | “Any physical connection made between two or more devices e.g. Copper wire, Ethernet cables, fibre optics etc.” |
| 80. | **1.3.2** | 1.3 Computer networks, connections and protocols | Wired and wireless networks, protocols and layers | Ethernet | “A standard for networking local area networks using protocols. Frames are used to transmit data. A frame contains the source and destination address, the data and error checking bits. Uses twisted pair and fibre optic cables. A switch connects computers together.” |
| 81. | **1.3.2** | 1.3 Computer networks, connections and protocols | Wired and wireless networks, protocols and layers | Wireless connection | “Any connection made between two or more devices which does not involve the need for a physical connection e.g. Wi-Fi, 4G, Bluetooth etc.” |
| 82. | **1.3.2** | 1.3 Computer networks, connections and protocols | Wired and wireless networks, protocols and layers | Wi-Fi | “Wireless connection to a network. Requires a wireless access point or router. Data is sent on a specific frequency. Each frequency is called a channel.” |
| 83. | **1.3.2** | 1.3 Computer networks, connections and protocols | Wired and wireless networks, protocols and layers | Bluetooth | “A method of exchanging data wirelessly over short distances, (much shorter than Wi-Fi). Examples of typical Bluetooth use could be, headphones, car mobiles etc.” |
| 84. | **1.3.2** | 1.3 Computer networks, connections and protocols | Wired and wireless networks, protocols and layers | Encryption | “Encoding readable data called plaintext into unreadable data called ciphertext. Only the intended recipient can decode the data using a key. Protects communications from hackers.” |
| 85. | **1.3.2** | 1.3 Computer networks, connections and protocols | Wired and wireless networks, protocols and layers | IP address | **Internet Protocol Address:** “A unique string of numbers separated by full stops that identifies each computer using the Internet Protocol to communicate over a network.” |
| 86. | **1.3.2** | 1.3 Computer networks, connections and protocols | Wired and wireless networks, protocols and layers | MAC address | **Media Access Control Address:** “A unique identifier assigned to network interfaces for communications at the data link layer of a network segment. MAC addresses are used as a network address for most network technologies, including Ethernet and Wi-Fi.” |
| 87. | **1.3.2** | 1.3 Computer networks, connections and protocols | Wired and wireless networks, protocols and layers | Standards | “The field of Computer Science is full of standards. They provide us with various rules for different areas of computing. Standards allow hardware and software to interact across the different manufacturers / producers.” |
| 88. | **1.3.2** | 1.3 Computer networks, connections and protocols | Wired and wireless networks, protocols and layers | Protocol | “A set of rules that allow two devices to communicate.” |

| | | | | | |
|---|---|---|---|---|---|
| 89. | **1.3.2** | 1.3 Computer networks, connections and protocols | Wired and wireless networks, protocols and layers | TCP/IP | **Transmission Control Protocol / Internet Protocol:** "TCP provides an error free transmission between two routers. IP routes packets across a wide area network." |
| 90. | **1.3.2** | 1.3 Computer networks, connections and protocols | Wired and wireless networks, protocols and layers | HTTP | **Hypertext Transfer Protocol:** "A client-server method of requesting and delivering HTML web pages. Used when the information on a web page is not sensitive or personal." |
| 91. | **1.3.2** | 1.3 Computer networks, connections and protocols | Wired and wireless networks, protocols and layers | HTTPS | **Hypertext Transfer Protocol Secure:** "Encryption and authentication for requesting and delivering HTML web pages. Used when sensitive form or database data needs to be transferred. e.g. passwords and bank account details." |
| 92. | **1.3.2** | 1.3 Computer networks, connections and protocols | Wired and wireless networks, protocols and layers | FTP | **File Transfer Protocol:** "Used for sending files between computers, usually on a wide area network. Typically used for uploading web pages and associated files to a web server for hosting." |
| 93. | **1.3.2** | 1.3 Computer networks, connections and protocols | Wired and wireless networks, protocols and layers | POP | **Post Office Protocol:** "Used by email clients to retrieve email from an email server." |
| 94. | **1.3.2** | 1.3 Computer networks, connections and protocols | Wired and wireless networks, protocols and layers | IMAP | **Internet Message Access Protocol:** "Used by mail clients to manage remote mailboxes and retrieve email from a mail server." |
| 95. | **1.3.2** | 1.3 Computer networks, connections and protocols | Wired and wireless networks, protocols and layers | SMTP | **Simple Mail Transfer Protocol:** "Sends email to an email server." |
| 96. | **1.3.2** | 1.3 Computer networks, connections and protocols | Wired and wireless networks, protocols and layers | Protocol layering | "The concept of a protocol not simply being a set of rules but those rules being built up into very specific layers and those rule layers behind built on top of each other in a deliberate order creating a layered protocol stack. This results in the rules of a protocol being executed in a specific sequence as you move through the protocol stack." |
| 97. | **1.4.1** | 1.4 Network security | Threats to computer systems and networks | Malware | "Software written to cause loss of data, encryption of data, fraud and identity theft: virus, worm, trojan, ransomware and spyware." |
| 98. | **1.4.1** | 1.4 Network security | Threats to computer systems and networks | Social engineering | "Most vulnerabilities are caused by humans. Not locking computers. Using insecure passwords. Not following/poor company network policies. Not installing protection software. Not being vigilant with email/files received. Not encrypting sensitive data." |
| 99. | **1.4.1** | 1.4 Network security | Threats to computer systems and networks | Phishing | "Sending emails purporting to be from reputable companies to induce people to reveal personal information." |
| 100. | **1.4.1** | 1.4 Network security | Threats to computer systems and networks | Brute-force attack | "A trial and error method of attempting passwords. Automated software is used to generate a large number of guesses." |

| | | | | | |
|---|---|---|---|---|---|
| 101. | **1.4.1** | 1.4 Network security | Threats to computer systems and networks | Denial of service attack | “Flooding a server with so much traffic it is unable to process legitimate requests.” |
| 102. | **1.4.1** | 1.4 Network security | Threats to computer systems and networks | Data interception and theft | “Stealing computer-based information.” |
| 103. | **1.4.1** | 1.4 Network security | Threats to computer systems and networks | SQL injection | “A hacking technique used to view or change data in a database by inserting SQL code instead of data into a text box on a form.” |
| 104. | **1.4.2** | 1.4 Network security | Identifying and preventing vulnerabilities | Penetration testing | “Testing designed to check the security and vulnerabilities of a system.” |
| 105. | **1.4.2** | 1.4 Network security | Identifying and preventing vulnerabilities | Anti-malware software | “Antimalware software protects against infections caused by many types of malware, including viruses, worms, Trojan horses, rootkits, spyware, key loggers, ransomware and adware.” |
| 106. | **1.4.2** | 1.4 Network security | Identifying and preventing vulnerabilities | Firewall | “A computer application used in a network to prevent external users gaining unauthorised access to a computer system.” |
| 107. | **1.4.2** | 1.4 Network security | Identifying and preventing vulnerabilities | User access level | “The amount of access a given user is allowed to a computer. On a network most users will have restricted access. Whereas a systems administer or network technician would be allowed much greater access with fewer restrictions.” |
| 108. | **1.4.2** | 1.4 Network security | Identifying and preventing vulnerabilities | Password | “A secret word or phrase that must be used to gain access to a computer / program / interface / system.” |
| 109. | **1.4.2** | 1.4 Network security | Identifying and preventing vulnerabilities | Physical security | “Any form of real world physical security to help protect data and systems e.g. Alarms, locks, security patrols etc.” |
| 110. | **1.5.1** | 1.5 Systems software | Operating systems | Systems software | “A generic umbrella term for all the software which typically ships with a new computer in order to make it work. It covers Operating Systems, Utility Software, Device Drivers etc.” |
| 111. | **1.5.1** | 1.5 Systems software | Operating systems | Operating system | “A sub-category of systems software. An operating system allows the user to install applications which then can interact with the hardware underneath via the operating system software. Most common operating systems are Windows, Linux, Unix, MacOS, iOS.” |
| 112. | **1.5.1** | 1.5 Systems software | Operating systems | User interface | “The means by which the user and a computer system interact, in particular the use of input devices and software.” |
| 113. | **1.5.1** | 1.5 Systems software | Operating systems | Memory management | “The process of the operating system deciding what should be in memory at any given time. Responsible for loading data and programs into and out of memory when required.” |
| 114. | **1.5.1** | 1.5 Systems software | Operating systems | Multitasking | “Running more than one application at a time by giving each one a slice of processor time.” |
| 115. | **1.5.1** | 1.5 Systems software | Operating systems | Peripheral management | “The process of your operating system dealing with requests / input / output to and from any connected peripheral devices such as a mouse, keyboard, webcam, speaker, scanner, printer etc.” |

| | | | | | |
|---|---|---|---|---|---|
| 116. | **1.5.1** | 1.5 Systems software | Operating systems | Driver | “Translates commands from the operating system into hardware specific commands that a device understands. e.g. A printer driver tells the printer how to print a document from the operating system.” |
| 117. | **1.5.1** | 1.5 Systems software | Operating systems | User management | “Operating system provides for: Allowing different people to log into the same computer with a username and password. Remembering personal settings. Managing access rights to files.” |
| 118. | **1.5.1** | 1.5 Systems software | Operating systems | File management | “Operating system provides: Access permissions for files (read and write). Opening files in associated programs. Moving, deleting and renaming files. Presenting a folder structure to the user.” |
| 119. | **1.5.2** | 1.5 Systems software | Utility software | Utility software | “A systems program that performs some specific task in the operation of the computer, for example file backup, virus checking or a compression program.” |
| 120. | **1.5.2** | 1.5 Systems software | Utility software | Encryption software | “Turns plaintext data into unreadable ciphertext data using a key. Protects data from being read by hackers.” |
| 121. | **1.5.2** | 1.5 Systems software | Utility software | Defragmentation software | “Different sized files saved on disk are deleted over time creating gaps on the disk. New files fill up the gaps, but may need more space than the gap provides resulting in fragments of the file being spread across the disk. Defragmentation rearranges parts of files back to contiguous space. Makes access quicker.” |
| 122. | **1.5.2** | 1.5 Systems software | Utility software | Data compression software | “Reduces the size of a file. Takes up less disk space. Quicker to download over the internet. Compressed files must be extracted before they can be read.” |
| 123. | **1.6.1** | 1.6 Ethical, legal, cultural and environmental concerns | Ethical, legal, cultural and environmental impact | Ethical issues | “The ethical and moral issues which have come about in modern society due to the increase use of computer science and its related technologies. e.g. Losing/changing jobs. Efficiency: robots work 24/7.<br>Access to IT is not equal (digital divide). Invasion of privacy. Responsibility for content on the internet. ” |
| 124. | **1.6.1** | 1.6 Ethical, legal, cultural and environmental concerns | Ethical, legal, cultural and environmental impact | Legal issues | “The legal issues which have come about in modern society due to the increase use of computer science and its related technologies. e.g. Copyright and ownership of digital content, different laws in different countries (crime may be committed in a certain country, but the people committing the crime could be physically located in another), hacking, piracy. ” |
| 125. | **1.6.1** | 1.6 Ethical, legal, cultural and environmental concerns | Ethical, legal, cultural and environmental impact | Cultural issues | “The cultural moral issues which have come about in modern society due to the increase use of computer science and its related technologies. e.g. Censorship to prevent political unrest and preserve culture. Geography & economy of a country affects access to networks and power. Increased mobile technology impacts on how people communicate: cyberbullying. ” |
| 126. | **1.6.1** | 1.6 Ethical, legal, cultural and environmental concerns | Ethical, legal, cultural and environmental impact | Environmental issues | “The environmental issues which have come about in modern society due to the increase use of computer science and its related technologies. e.g. Manufacturing computers uses fossil fuels. Limited number of natural resources. Data centres use |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 2% of global energy. Computers contain hazardous materials, often shipped to other countries for disposal. ” |
| 127. | **1.6.1** | 1.6 Ethical, legal, cultural and environmental concerns | Ethical, legal, cultural and environmental impact | Privacy issues | “The privacy issues which have come about in modern society due to the increase use of computer science and its related technologies. e.g. Increase in always on, voice activated devices in the home. Rise in CCTV. Rise in social networking and GPS tracking.” |
| 128. | **1.6.1** | 1.6 Ethical, legal, cultural and environmental concerns | Ethical, legal, cultural and environmental impact | The Data Protection Act 2018 | “Legislation which protects individuals from unreasonable use of their personal data. Updated in 2018 to encompass all the new requirements of the General Data Protection Regulation.” |
| 129. | **1.6.1** | 1.6 Ethical, legal, cultural and environmental concerns | Ethical, legal, cultural and environmental impact | Computer Misuse Act 1990 | “Legislation which defines electronic vandalism, unauthorised access to computer systems and theft of information.” |
| 130. | **1.6.1** | 1.6 Ethical, legal, cultural and environmental concerns | Ethical, legal, cultural and environmental impact | Copyright Design and Patents Act 1998 | “Legislation which gives creators of literacy, dramatic, musical and artistic works the right to control the ways in which their material may be used.” |
| 131. | **1.6.1** | 1.6 Ethical, legal, cultural and environmental concerns | Ethical, legal, cultural and environmental impact | Software licences | “A set of binding legal terms which often come with a commercial software application, they will dictate how you can use the software e.g. personal use only, company use, installed on just a single computer etc. |
| 132. | **1.6.1** | 1.6 Ethical, legal, cultural and environmental concerns | Ethical, legal, cultural and environmental impact | Open source | “Users can modify and distribute the software. Can be installed on any number of computers. Support provided by the community. Users have access to the source code. May not be fully tested.” |
| 133. | **1.6.1** | 1.6 Ethical, legal, cultural and environmental concerns | Ethical, legal, cultural and environmental impact | Proprietary | “Users cannot modify the software. Copyright protected. Usually paid for. Licensed per user or per computer. Support provided by developers. Users do not have access to the source code. Fully tested and supported by developers.” |
| 134. | **2.1.1** | 2.1 Algorithms | Computational thinking | Computational thinking | “The thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out.” |
| 135. | **2.1.1** | 2.1 Algorithms | Computational thinking | Abstraction | “The process of separating ideas from specific instances of those ideas at work. Computational structures are defined by their meanings, while hiding away the details of how they work. Abstraction tries to factor out details from a common pattern so that programmers can work close to the level of human thoughts, leaving out details which matter in practice, but are immaterial to the problem being solved.” |
| 136. | **2.1.1** | 2.1 Algorithms | Computational thinking | Decomposition | “The process by which a complex problem or system is broken down into parts that are easier to conceive, understand, program and maintain.” |
| 137. | **2.1.1** | 2.1 Algorithms | Computational thinking | Algorithmic thinking | “A way of getting to a solution by identifying the steps needed.” |
| 138. | **2.1.2** | 2.1 Algorithms | Designing, creating and refining algorithms | Problem inputs | “Any information or data which goes into a system.” |

| | | | | | |
|---|---|---|---|---|---|
| 139. | **2.1.2** | 2.1 Algorithms | Designing, creating and refining algorithms | Problem processes | "Anything which happens to data during a system running e.g. performing calculations." |
| 140. | **2.1.2** | 2.1 Algorithms | Designing, creating and refining algorithms | Problem outputs | "Any information of data which leaves a system." |
| 141. | **2.1.2** | 2.1 Algorithms | Designing, creating and refining algorithms | Structure diagram | "A diagram which looks like an upside down tree, with one node at the top (root) and many below. It is used when designing solutions to problems in order to help break a large problem down into a number of small parts." |
| 142. | **2.1.2** | 2.1 Algorithms | Designing, creating and refining algorithms | Psuedocode | "A language independent description of the steps of an algorithm. Intended for humans to express and design algorithms before coding." |
| 143. | **2.1.2** | 2.1 Algorithms | Designing, creating and refining algorithms | Flowchart | "A method of designing algorithms before coding using symbols." |
| 144. | **2.1.2** | 2.1 Algorithms | Designing, creating and refining algorithms | Trace table | "A technique used to test algorithms, in order to make sure that no logical errors occur while the algorithm is being processed. The table usually has one column for each variable. Each row of the table shows how the various values held in variables change as the algorithm is running." |
| 145. | **2.1.3** | 2.1 Algorithms | Searching and sorting algorithms | Searching algorithms | "An algorithm which attempts to find a given value in a data set." |
| 146. | **2.1.3** | 2.1 Algorithms | Searching and sorting algorithms | Binary search | "A particularly efficient search method. It only works if records in the file are in sequence. A binary search involvers accessing the middle record in the file and determining if the target record has been found or, if not, if it is before or after in the sequence. This process is repeated on the part of the file where the target record is expected, until it is found." |
| 147. | **2.1.3** | 2.1 Algorithms | Searching and sorting algorithms | Linear search | "Involves examining each entry in turn in the file until the time is found or the end of the file is reached. Unless the file is in some useful order a serial search has to be used." |
| 148. | **2.1.3** | 2.1 Algorithms | Searching and sorting algorithms | Sorting algorithm | "An algorithm which attempts to sort an unordered set of values." |
| 149. | **2.1.3** | 2.1 Algorithms | Searching and sorting algorithms | Bubble sort | "A simple algorithm popular with inexperienced programmers. It is inefficient when sorting large amounts of data as the time taken is related to the square of the number of items. If 10 items take 1ms then 100 times will take 100ms (this is 10 times the number of items and so the time will be 102 or 100 times longer)." |
| 150. | **2.1.3** | 2.1 Algorithms | Searching and sorting algorithms | Merge sort | "A type of divide and conquer algorithm that was incited by John von Neumann. First the list is divided into the smallest unit (1 element), then each element is compared with the adjacent list to sort and merge the two adjacent lists. Finally all elements are sorted and merged." |

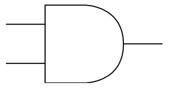
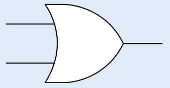
| | | | | | |
|---|---|---|---|---|---|
| 151. | **2.1.3** | 2.1 Algorithms | Searching and sorting algorithms | Insertion sort | “A simple sorting algorithm that builds the final sorted array (or list) one item at time. It is much less efficient on large lists than more advanced algorithms such as quicksort, heapsort, or merge sort.” |
| 152. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Variable | “A value that can change, depending on conditions or on information passed to the program.” |
| 153. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Constant | “A value that cannot be altered by the program during normal execution, i.e., the value is constant.” |
| 154. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Operator | “A generic term in Computer Science. An operator tells how you to manipulate or interpret values. Categories of operators you will need to know are: Arithmetic, Boolean, Comparison.” |
| 155. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Assignment | “Giving a variable or constant a value. e.g. counter = 0” |
| 156. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Programming construct | “Lines / block of code which perform a distinct function. The three basic programming constructs are: Sequence, Selection, Iteration.” |
| 157. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Sequence | “One of the 3 basic programming constructs. Instructions happening one after the other in order is sequence.” |
| 158. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Selection | “One of the 3 basic programming constructs. Instructions which can evaluate a Boolean expression and then branch the code to one or more alternatives paths is branching / selection.” |
| 159. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Count controlled iteration | “An iteration which loops a fixed number of times. The count is kept in a variable called an index or counter. When the index reaches a certain value (the loop bound) the loop will end. Count-controlled repetition is often called definite repetition because the number of repetitions is known before the loop begins executing.” |
| 160. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Condition controlled iteration | “A way for computer programs to repeat one or more various steps depending on conditions set either by the programmer initially or real-time by the actual program.” |
| 161. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Arithmetic operator | “+, -, /, *, ^. Used in mathematical expressions e.g. num1 + num2 = sum” |
| 162. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Boolean operator: AND | “A logical operator used within a program. AND works by only returning TRUE if both values being compared are TRUE.” |
| 163. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Boolean operator: OR | “A logical operator used within a program. OR works by returning TRUE as long as either value being compared is TRUE.” |
| 164. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Boolean operator: NOT | “A logical operator used within a program. NOT works by returning FALSE if the input is TRUE, and returning TRUE if the input is FALSE.” |
| 165. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Comparison operator: == | “One of the standard comparison operators you can use in virtually all programming languages to carry out computing-related mathematics: |

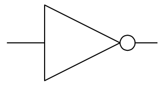
| | | | | | |
|---|---|---|---|---|---|
| | | | | | == is the standard symbol used to represent ‘equal to’ ” |
| 166. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Comparison operator: != | “One of the standard comparison operators you can use in virtually all programming languages to carry out computing-related mathematics:<br>!= is the standard symbol used to represent ‘not equal to’ ” |
| 167. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Comparison operator: < | “One of the standard comparison operators you can use in virtually all programming languages to carry out computing-related mathematics:<br>< is the standard symbol used to represent ‘less than’ ” |
| 168. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Comparison operator: <= | “One of the standard comparison operators you can use in virtually all programming languages to carry out computing-related mathematics:<br><= is the standard symbol used to represent ‘less than or equal to’ ” |
| 169. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Comparison operator: > | “One of the standard comparison operators you can use in virtually all programming languages to carry out computing-related mathematics:<br>> is the standard symbol used to represent ‘greater than’ ” |
| 170. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Comparison operator: >= | “One of the standard comparison operators you can use in virtually all programming languages to carry out computing-related mathematics:<br>>= is the standard symbol used to represent ‘greater than or equal to’ ” |
| 171. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Arithmetic operator: + | “One of the standard arithmetic operators you can use in virtually all programming languages to carry out computing-related mathematics:<br>+ is the standard symbol used for addition.” |
| 172. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Arithmetic operator: - | “One of the standard arithmetic operators you can use in virtually all programming languages to carry out computing-related mathematics:<br>- is the standard symbol used for subtraction.” |
| 173. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Arithmetic operator: * | “One of the standard arithmetic operators you can use in virtually all programming languages to carry out computing-related mathematics:<br>* is the standard symbol used for multiplication.” |
| 174. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Arithmetic operator: / | “One of the standard arithmetic operators you can use in virtually all programming languages to carry out computing-related mathematics:<br>/ is the standard symbol used for real division.” |
| 175. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Arithmetic operator: MOD | “One of the standard arithmetic operators you can use in virtually all programming to carry out integer division:<br>MOD gives you remainder left over e.g. 10 MOD 3 would give you 1.” |
| 176. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Arithmetic operator: DIV | “One of the standard arithmetic operators you can use in virtually all programming to carry out integer division:<br>DIV gives you the number of times a number fits into another number e.g. 10 MOD 3 would give you 3.” |
| 177. | **2.2.1** | 2.2 Programming fundamentals | Programming fundamentals | Arithmetic operator: ^ | “One of the standard arithmetic operators you can use in virtually all programming languages to carry out computing-related mathematics: |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | ^ is the standard symbol used for exponent." |
| 178. | **2.2.2** | 2.2 Programming fundamentals | Data types | Data type | "The basic data types provided by a programming language as building blocks. Most languages allow more complicated composite types to be recursively construction starting from basic types. E.g. char, integer, float, Boolean. As an extension a 'string' data type is constructed behind the scenes of many char data types." |
| 179. | **2.2.2** | 2.2 Programming fundamentals | Data types | Integer | "A data type used to store positive and negative whole numbers." |
| 180. | **2.2.2** | 2.2 Programming fundamentals | Data types | Real | "A data type used to store an approximation of a real number in a way that can support a trade-off between range and precision. A number is, in general, represented approximately to a fixed number of significant digits and scaled using an exponent." |
| 181. | **2.2.2** | 2.2 Programming fundamentals | Data types | Boolean | "Used to store the logical conditions TRUE / FALSE. Often translated to On/Off, Yes/No etc." |
| 182. | **2.2.2** | 2.2 Programming fundamentals | Data types | Character | "A single alphanumeric character or symbol." |
| 183. | **2.2.2** | 2.2 Programming fundamentals | Data types | String | "A sequence of alphanumeric characters and or symbols. e.g. a word or sentence." |
| 184. | **2.2.2** | 2.2 Programming fundamentals | Data types | Casting | "Converting a variable from one data type to another. e.g. variable entered as a string, but needs to be an integer for calculation. age = INPUT("Enter your age: ") age = INT(age)" |
| 185. | **2.2.3** | 2.2 Programming fundamentals | Additional programming techniques | String manipulation | "Commands and techniques which allow you to alter and extract information from textual strings e.g. .length .substring(x, i) .left(i) .right(i) .upper .lower ASC(…) CHR(…)" |
| 186. | **2.2.3** | 2.2 Programming fundamentals | Additional programming techniques | File handling: Open | "File handling is the process of dealing with input to and from files. Files first have to be opened, this creates a handle to the file and allows reading and writing." |
| 187. | **2.2.3** | 2.2 Programming fundamentals | Additional programming techniques | File handling: Read | "File handling is the process of dealing with input to and from files. Once a file has been opened it is possible to use commands to read its contents and return them to your program." |
| 188. | **2.2.3** | 2.2 Programming fundamentals | Additional programming techniques | File handling: Write | "File handling is the process of dealing with input to and from files. Once a file has be opened it is possible to use commands to write data to file from your program." |
| 189. | **2.2.3** | 2.2 Programming fundamentals | Additional programming techniques | File handling: Close | "File handling is the process of dealing with input to and from files. Once you are done reading / writing it is important to close a file, this releases the file handle and breaks the connection between it and your program." |

| | | | | | |
|---|---|---|---|---|---|
| 190. | **2.2.3** | 2.2 Programming fundamentals | Additional programming techniques | Record | “A data structure which consists of a collection of elements, typically in fixed number and sequence and typically indexed by names. The elements of records may also be called fields.”<br><br>“The record type is a data type that describes such values and variables. Most modern computer languages allow the programmer to define new record types. The definition includes specifying the data type of each field and an identifier by which it can be accessed.” |
| 191. | **2.2.3** | 2.2 Programming fundamentals | Additional programming techniques | SQL | “The language and syntax used to write and run database queries” |
| 192. | **2.2.3** | 2.2 Programming fundamentals | Additional programming techniques | SQL command: SELECT | “A key word in the SQL programming language used for the querying (retrieval) of data.” e.g.<br><br>SELECT Name, Age, Class<br>FROM Students_table<br>WHERE Gender = ‘Male’ |
| 193. | **2.2.3** | 2.2 Programming fundamentals | Additional programming techniques | SQL command: FROM | “A key word in the SQL programming language used to signify which table(s) we are using.” e.g.<br><br>SELECT Name, Age, Class<br>FROM Students_table<br>WHERE Gender = ‘Male’ |
| 194. | **2.2.3** | 2.2 Programming fundamentals | Additional programming techniques | SQL command: WHERE | “A key word in the SQL programming language used to filter the results of your query.” e.g.<br><br>SELECT Name, Age, Class<br>FROM Students_table<br>WHERE Gender = ‘Male’ |
| 195. | **2.2.3** | 2.2 Programming fundamentals | Additional programming techniques | Array | “A set of data items of the same type grouped together using a single identifier. Each of the data items is addressed by the variable name and a subscript.” |
| 196. | **2.2.3** | 2.2 Programming fundamentals | Additional programming techniques | Sub programs | “A block of code given a unique identifiable name within a program. Supports code reuse and good programming technique.” |
| 197. | **2.2.3** | 2.2 Programming fundamentals | Additional programming techniques | Procedure | “A block of code given a unique identifiable name within a program. A procedure can take either zero or more parameters when it is called. The procedure should be designed and written to perform one task or action which is clearly indicated by its name.” |

| | | | | | |
|---|---|---|---|---|---|
| 198. | **2.2.3** | 2.2 Programming fundamentals | Additional programming techniques | Function | “A block of code given a unique identifiable name within a program.  A function can take either zero or more parameters when it is called and should return a value.  The function should be designed and written to perform one task or action which is clearly indicated by its name.” |
| 199. | **2.2.3** | 2.2 Programming fundamentals | Additional programming techniques | Random number generation | “Most programming languages have built in functions or libraries that allow you to easily generate random numbers.  Creating truly random numbers is actually something quite difficult for a computer, and these algorithms are quite complex.” |
| 200. | **2.3.1** | 2.3 Producing robust programs | Defensive design | Defensive design | “Defensive design is the practice of planning for contingencies in the design stage of a project or undertaking.” |
| 201. | **2.3.1** | 2.3 Producing robust programs | Defensive design | Anticipating misuse | “The ability of a programmer to consider how the end user might accidently (or on purpose) break the program and then to write additional code to handle these situations.” |
| 202. | **2.3.1** | 2.3 Producing robust programs | Defensive design | Authentication | “Verifying a user identity before they can use a program with username and password. Strong passwords over a certain length with symbols and mixed case are advised.” |
| 203. | **2.3.1** | 2.3 Producing robust programs | Defensive design | Input validation | “Ensuring data input by the user meets specific criteria before processing. Range check. E.g. between 1 and 31. Type check. E.g. number not symbol. Presence check. E.g. data has been input. Format check. E.g. postcode is LLN(N) NLL. ” |
| 204. | **2.3.1** | 2.3 Producing robust programs | Defensive design | Maintainability | “A selection of techniques and methods that make code easy to debug, update and maintain.” |
| 205. | **2.3.1** | 2.3 Producing robust programs | Defensive design | Naming conventions | “Many programmers / organisations use certain naming conventions for their variables / contents / procedure names etc.<br><br>Camel case is a popular one used in industry where the first word of an identifier uses all lower case, with all subsequent words starting with a capital letter:<br><br>e.g. studentsFirstName” |
| 206. | **2.3.1** | 2.3 Producing robust programs | Defensive design | Indentation | “Indenting makes it easy to see where structures begin and end. Conditions and iterations should be indented. Code inside procedures and functions should be indented.” |
| 207. | **2.3.1** | 2.3 Producing robust programs | Defensive design | Commenting | “Used by a programmer to explains sections of code.  Ignored by the compiler.” |
| 208. | **2.3.2** | 2.3 Producing robust programs | Testing | Testing | “This involves testing the program under various conditions to make sure it is going to work. You need to think about what devices it could be used on and what might cause the program to crash.” |
| 209. | **2.3.2** | 2.3 Producing robust programs | Testing | Iterative testing | “Each module of a program is tested as it is developed.” |
| 210. | **2.3.2** | 2.3 Producing robust programs | Testing | Final/terminal testing | “Testing that all the modules of a program work together as expected. Checking the program meets the expectations of the user with real data.” |

| | | | | | |
|---|---|---|---|---|---|
| 211. | **2.3.2** | 2.3 Producing robust programs | Testing | Syntax error | “Rules of the language have been broken. The program will not run. Variables not being declared before use. Incompatibility of variable types. E.g. sum = A Using assignments incorrectly. E.g. 2 + 2 = x Keywords misspelt. E.g. PRNT(“Hello”)” |
| 212. | **2.3.2** | 2.3 Producing robust programs | Testing | Logical error | “The program runs but does not give the expected output. Division by zero. Infinite loop. Memory full. File not found.” |
| 213. | **2.3.2** | 2.3 Producing robust programs | Testing | Test data | “Values used to test a program, includes normal test data, boundary test data and erroneous test data.” |
| 214. | **2.3.2** | 2.3 Producing robust programs | Testing | Test data: Normal | “Data supplied to a program which you would expect.<br><br>e.g. A program has been written to average out test scores from students, the scores allowed are from 0-100. Normal test data could be: 32, 40, 82 etc.” |
| 215. | **2.3.2** | 2.3 Producing robust programs | Testing | Test data: Boundary | “Data supplied to a program which is designed to test the boundaries of a problem.<br><br>e.g. A program has been written to average out test scores from students, the scores allowed are from 0-100. Boundary test data could be: -1,0,1 or 99,100,101” |
| 216. | **2.3.2** | 2.3 Producing robust programs | Testing | Test data: Invalid | “Data of the correct type but outside accepted validation limits.<br><br>e.g. a program asks for the user to input a whole number from 0-100 then examples of invalid data could be -5, 150 etc.” |
| 217. | **2.3.2** | 2.3 Producing robust programs | Testing | Test data: Erroneous | “Data of the incorrect type which should be rejected by a computer system.<br><br>e.g. a program asks for the user to input a whole number from 0-100 then examples of erroneous data could be the string ‘hello’ or the real 3.725 etc.” |
| 218. | **2.4.1** | 2.4 Boolean logic | Boolean logic | Logic diagram | “A method of expression Boolean Logic in a diagrammatic form using a set of standard symbols representing the various Logic Gates such as AND NOT OR NAND etc.” |
| 219. | **2.4.1** | 2.4 Boolean logic | Boolean logic | Logic gate | “An individual symbol used in a logic diagram which represents a single gate e.g. AND, OR, NOT.” |
| 220. | **2.4.1** | 2.4 Boolean logic | Boolean logic | Logic gate: AND |  “A logic gate which accepts two inputs and produces one output. Both inputs must be TRUE (1) for the output to the TRUE (1), otherwise the output is FALSE (0).” |
| 221. | **2.4.1** | 2.4 Boolean logic | Boolean logic | Logic gate: OR |  “A logic gate which accepts two inputs and produces one output. At least one input must be TRUE (1) for the output to the TRUE (1), otherwise the output is FALSE (0).” |

| | | | | | |
|---|---|---|---|---|---|
| 222. | **2.4.1** | 2.4 Boolean logic | Boolean logic | Logic gate: NOT |  "A logic gate which accepts one input and produces one output. If the input is TRUE (1) then the output will be FALSE (0). If the input is FALSE (0) then the output will be TRUE (1)." |
| 223. | **2.4.1** | 2.4 Boolean logic | Boolean logic | Truth table | "A notation used in Boolean algebra for defining the output of a logic gate or logic circuit for all possible combinations of inputs." |
| 224. | **2.5.1** | 2.5 Programming languages and IDEs | Languages | High-level language | "A language designed to help a programmer express a computer program in a way that reflects the problem that is being solved, rather than the details of how the computer will produce the solution. One-to-many language." |
| 225. | **2.5.1** | 2.5 Programming languages and IDEs | Languages | Low-level language | "A language which is close to machine code. Related closely to the design of the machine. A one-to-one language." |
| 226. | **2.5.1** | 2.5 Programming languages and IDEs | Languages | Translator | "A program that translates a program written in assembly language into machine code." |
| 227. | **2.5.1** | 2.5 Programming languages and IDEs | Languages | Compiler | "A program that translates a high-level language program, source code, into a computer's machine code." |
| 228. | **2.5.1** | 2.5 Programming languages and IDEs | Languages | Interpreter | "Translates and executes a program one statement at a time." |
| 229. | **2.5.2** | 2.5 Programming languages and IDEs | The Integrated Development Environment | IDE | **Integrated Develop Environment:** "A software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger." |
| 230. | **2.5.2** | 2.5 Programming languages and IDEs | The Integrated Development Environment | IDE: Error diagnostics | "These are tools provided by IDE's which give detailed feedback on errors in your code. " |
| 231. | **2.5.2** | 2.5 Programming languages and IDEs | The Integrated Development Environment | IDE: Run-time environment | "A configuration of hardware and software. It includes the CPU type, operating system and any runtime engines or system software required by a particular category of applications." |