GCSE CS Revision Sheet

Eason's Toolbox

Yicheng Shao

Version 1. April 9, 2023

Contents

What is this and why this?

GCSE Computer Science is a knowledge-intense exam. Unlike what most people think, Computer Science is a subject that requires a lot of writing. Therefore, I made this document based on the CIE IGCSE Computer Science (9-1) Syllabus from 2023 onwards. I hope this could help with your IGCSE studies!

This is more of an extension of the syllabus and the structure is the same. However, it provides some sample answers for those questions in the syllabus and is a good way to refer to your self-assessment based on the syllabus.

I am also an IGCSE student so errors are inevitable in this document. Feel free to email eason.syc@icloud.com to point out any mistakes or submit an issue on the GitHub page!

Section 1 Data Representation

§1.1 Number Systems

Knowledge 1.1.1. **Analogue data** is continuous. **Digital data** is discrete.

Knowledge 1.1.2. **Denary** is a base-10 number system. **Binary** is a base-2 number system. **Hexadecimal** is a base-16 number system.

Knowledge 1.1.3. Binary is required as computers process data using logic gates and registers.

Knowledge 1.1.4. Digits in decimal, hexadecimal, and binary can be converted as

Denary	0	1	2	3
Hexadecimal	0	1	2	3
Binary	0000	0001	0010	0011
Denary	4	5	6	7
Hexadecimal	4	5	6	7
Binary	0100	0101	0110	0111
Denary	8	9	10	11
Hexadecimal	8	9	A	B
Binary	1000	1001	1010	1011
Denary	12	13	14	15
Hexadecimal	C	D	E	F
Binary	1100	1101	1110	1111

Knowledge 1.1.5. To convert binary into hexadecimal, we map four consecutive digits (divide from the right) to one digit in hexadecimal. Vice versa.

Knowledge 1.1.6. To convert binary or hexadecimal into denary, we write the digit it represents on top (e.g. 2, 4, 8 or 16, 256, 4096), times it with the number below and sum them together. Write from right to left.

Knowledge 1.1.7. To convert denary into binary or hexadecimal, we write down the result of integer division of the number and the base and use the remainder to continue. When we get a remainder that is less than the base, we write the result backwards.

Knowledge 1.1.8. Programmers use hexadecimal as it is easier for humans to read. Examples include MAC code.

Knowledge 1.1.9. Binary addition works similarly to denary addition.

Knowledge 1.1.10. **Overflow error** stands for when data is too big when stored in certain amounts of digits. It could happen in binary addition or binary shifts.

Knowledge 1.1.11. **Left shift** is defined by shifting all the digits to the left. **Right shift** is defined similarly. We fill in the empty slots with 0 and simply delete the slots that went out.

Knowledge 1.1.12. Left shift is timing the original number by the base. Right shift is (integer) dividing it.

Knowledge 1.1.13. **Negative binary is stored as follows.**

- Find the binary of the original number (i.e. non-negative).
- Invert all the 0s and 1s.
- Add this by 1 (treat it as a positive number).

This is called **two's complement**.

§1.2 Text, Sound and Image

Knowledge 1.2.1. We store text using **character sets**. ASCII code and Unicode are two examples. Unicode is more universal but is bigger in storing a single character.

Knowledge 1.2.2. An **image** is a series of **pixels** that are converted to binary. The **resolution** is the number of pixels in an image. The **colour depth** is the number of bits used to store each colour. There is **metadata** at the beginning of a file to state that information. The file size and quality of the image increase as the resolution and colour depth increase.

Knowledge 1.2.3. A **sound sampling** is done to convert analogue sound data into digital data. The **sample rate** is the number of samples taken in a second. The **sample resolution** is the number of bits per sample. The accuracy of the recording and the file size increases as the sample rate and resolution increase.

§1.3 Data Storage and Compression

Knowledge 1.3.1. The following are required data representation units:

- A **bit** is a binary digit.
- A **nibble** is four bits.
- A **byte** is eight bits.
- A **kibibyte** (KiB) is 1024 bytes.
- A **mebibyte** MiB is 1024 kibibytes.
- A **gibibyte** GiB is 1024 mebibytes.
- A **tebibyte** TiB is 1024 gibibytes.
- A **pebibyte** PiB is 1024 tebibytes.
- A **exbibyte** EiB is 1024 pebibytes.

Knowledge 1.3.2. Way to calculate the size of an image:

$$\text{size of image} = \text{width of image} \times \text{height of image} \times \text{colour depth of image} \times \text{number of images in file}.$$

Knowledge 1.3.3. Way to calculate the size of a soundtrack:

$$\text{size of soundtrack} = \text{sample rate} \times \text{sample resolution} \times \text{length of soundtrack}.$$

Knowledge 1.3.4. **Data compression** is the method used to reduce the size of a file. It is necessary as

- it will require smaller storage;
- it will take less time to transmit;
- it will be quicker to upload and download;
- it will require smaller bandwidth.

Knowledge 1.3.5. **Lossy** compression can not be reversed, it permanently removes unnecessary and redundant data in a file. Examples to compress include reducing resolution, reducing colour depth, and reducing sample rate. Examples of files include .jpg and .mp3.

Knowledge 1.3.6. **Lossless** compression reduces size without loss of information. Examples to compress include run-length encoding (RLE) which groups together repeating data. Examples of files include .midi.

Section 2 Data Transmission

§2.1 Types and Methods of Data Transmission

Knowledge 2.1.1. Data is transmitted in small units called **packets**. They are divided into **packet header**, **payload** and **trailer**.

Knowledge 2.1.2. The packet header includes the destination address, packet number and originator's address. The addresses are often IP addresses.

Knowledge 2.1.3. The payload is the actual data you are sending.

Knowledge 2.1.4. The trailer (a.k.a. footer) indicates 'end of packet' and the error detection systems used.

Knowledge 2.1.5. **Packet switching** is the process of transmitting packets over a **network** with **routers**. The following is the process:

- Data is broken down into packets;
- Each packet could take a different route;
- A router controls the route a packet takes;
- Packets may arrive out of order;
- Once the last packet has arrived, packets are reordered.

Knowledge 2.1.6. Data transmission can be divided into two types, **serial** and **parallel**.

Knowledge 2.1.7. Serial transmission is transmitted one bit at a time along one wire. Parallel transmission is transmitted multiple bits at a time along multiple wires.

Advantages of Serial Transmission: (which relates to a disadvantage of parallel transmission)

- The data is transmitted in sequence, so there is less chance of data being skewed.
- The data is transmitted along a single wire hence less chance of **interference**, with less chance of error.
- Only one wire is necessary so it is cheaper.

Advantages of Parallel Transmission: (which relates to a disadvantage of series transmission) Data transmission is faster since multiple bits are sent at a time.

Disadvantage for serial: A start bit and an end bit is necessary.

Advantage for parallel: No requirement to convert data across the network. (Since within a device data is transmitted parallel.)

Knowledge 2.1.8. Data transmission can be divided into three types, **simplex**, **half-duplex** and **duplex**.

Knowledge 2.1.9. **Simplex** transmission is when data transmit in only one direction. **Half-duplex** is when data can be transmitted bi-directionally but only one direction at a time. **Full-duplex** is when data can be transmitted bi-directionally simultaneously.

Knowledge 2.1.10. The **USB**, or universal series bus, is a standard (protocol) for data transmission.

Knowledge 2.1.11. **Advantages for USB**:

- It is a very simple interface, and a very low rate of error (it is probably idiot-proof).
- The speed is relatively high.
- It is a very universal standard and widely used.
- It is automatic in terms of detection while inserted, including downloading of drivers.
- It can be used as a power source.

Disadvantages for USB:

- USB is length-limited.
- USB is not as fast as certain interfaces such as **ethernet** (and thunderbolt, PCIe, SATA).

§2.2 Methods of Error Detection

Knowledge 2.2.1. **Error detection** is necessary since during transmission data can have interference, such as data loss, data gain and data change.

Knowledge 2.2.2. **Parity check** can be odd or even (which is determined previously). The odd or even stands for the number of 1s in the data. **Parity block check** and **parity byte** is also used (vertical and horizontal parity checks) to detect and potentially recover data.

Knowledge 2.2.3. **Checksum** is using a calculated value to check for errors. This value is calculated from the data before transmission and compared with the result of the value after transmission. Examples include modulus 11 (md5, sha1, sha256).

Knowledge 2.2.4. **Echo check** is when data is transmitted back to the sender to check.

Knowledge 2.2.5. A **check digit** is like a checksum and parity combined. The checksum data is included in the data itself. Examples include ISBN and bar codes.

Knowledge 2.2.6. **ARQ** is automatic repeat query when data can be repeatedly transmitted when an error occurs. It involves **acknowledgement** and **timeout**.

A positive acknowledgement involves:

- The sending device transmits the first data packet;
- The receiving device receives the data and checks it for errors;
- If it does not have an error: it sends a positive acknowledgement and the sender continues sending the next packet.
- If the sender does not receive the positive acknowledgement within the set timeframe, this is a timeout, and it will continue sending the same packet until a positive acknowledgement occurs or a limit is reached.

A negative ackowledgement involves:

- The sending device transmits the first data packet;
- The receiving device receives the data and checks it for errors;
- If it does not have an error: no further action. The sender begins to send the next packet after the set time.
- If it does have an error: A negative acknowledgement is sent. The sender will resend the data.

§2.3 Encryption

Knowledge 2.3.1. **Encryption** is a method of data protection while data is transmitted since hackers may try and intercept the data while transmitting.

Knowledge 2.3.2. The **plain text** is the original text, and the **encryption key** is the method used to encrypt it into the **cipher text** which seems meaningless.

Knowledge 2.3.3. **Symmetric** encryption is when the encryption and decryption use the same key.

- The plain text is encrypted into cipher text using an encryption key;
- The cipher text and the encryption key are sent separately to the receiving device;
- The same key is then used to decrypt the cipher text back into its plain text form.

Asymmetric encryption is when a public key (encryption) and a private key (decryption) are used separately.

- The plain text is encrypted into cipher text using a public key;
- The cipher text is transmitted to the receiving device;
- The cipher text is decrypted with a private key.

The reverse (private for encrypting and public for decrypting) can be worked for digital signatures, e.g. SSL.

Section 3 Hardware

§3.1 Computer Architecture

Knowledge 3.1.1. **CPU**, the central processing unit processes instructions and data that are input into the computer so that the result can be output. **Microprocessor** is a type of integrated circuit on a single chip.

Knowledge 3.1.2. Components in a von-Neumann CPU include:

Control Unit (CU)	A component that sends signals to control the interactions of all other components during the fetch-decode-execute cycle.
Arithmetic Logic Unit (ALU)	A component that performs all calculations and logical operations required during the fetch-decode-execute cycle.
Address Bus	A bus that is used to transmit addresses within the CPU and to, and from RAM.
Data Bus	A bus that is used to transmit data or instructions within the CPU, and to and from RAM.
Control Bus	A bus that is used to transmit control signals that are sent by the control unit.
Memory Address Register (MAR)	A register that stores the address of where data or instruction is located in RAM.
Memory Data Register (MDR)	A register that stores data or an instruction when it is fetched from RAM.
Program Counter (PC)	A register that holds the address of the next instruction to be processed.
Current Instruction Register (CIR)	A register that holds the instruction that is currently being processed.
Accumulator (ACC)	A register that is built into the ALU that is used to store the interim results of calculations.

Knowledge 3.1.3. **The fetch-decode-execute** cycle is the process in which the CPU processes instructions (which usually adjusts data).

- The data within the PC is sent to the MAR.
- PC adds itself by one instruction.
- The MAR data is sent to the RAM via the Address Bus.
- The data within that address is sent to the MDR via the Data Bus.
- This data (the instruction) is sent to the CIR.
- The CPU decodes the instruction using an instruction set. (A set of all commands)
- The CPU fetches all data similarly but processes them with ALU (ACC is also involved in loops).

The control bus is involved in the fetch stage while data is transmitted via the two other buses.

Knowledge 3.1.4. The **core** (which is a unit of the FDE cycle execution), the **clock speed** (which is a speed where instructions are processed), and the **cache** (which is a type of RAM-like volatile storage but smaller and quicker) will affect the performance of a CPU.

Knowledge 3.1.5. An **instruction set** is a list of all the commands that can be processed by a CPU and the commands are machine code

Knowledge 3.1.6. An **embedded system** is used to perform a dedicated function, e.g. domestic appliances, cars, security systems, lighting systems or vending machines. This is different to a **general purpose computer** that is used to perform many different functions, e.g. a personal computer (PC) or a laptop

§3.2 Input and output devices

Knowledge 3.2.1. An **input device** is a device that allows data to be entered into a computer system.

Knowledge 3.2.2. Input devices include:

Barcode Scanner	It scans a barcode so that the data stored in the barcode can be obtained.	It is used in a supermarket to get the price of a product and as part of a tock control system.
Digital Camera	It captures light through a lens and converts it into binary.	It is built into a mobile phone to allow the user to photograph items or people.
Keyboard	It allows the user to press keys that have a designated ASCII/Unicode value that is converted to binary.	It is one of the main methods of input that allows a user to type data into a personal computer.
Microphone	It captures soundwaves and converts them to binary.	It is built into a mobile phone to capture the user's voice so that it can be heard by other users.
Optical Mouse	It captures the light that is bounced back from a laser that is shone from the mouse to the surface underneath, to track the mouse's movements.	It is one of the main methods of input that allows a user to select icons and menu options whilst using a personal computer.
QR Code Scanner	It uses a sensor or a camera to capture light reflected from a QR code and converts it to binary.	it can be an application that is downloaded onto a mobile phone and used to scan QR codes that store information, e.g. a website link.
Scanner (2D and 3D)	They use sensors to capture light that is reflected from a 2D or 3D object and convert it to binary.	It can be used to scan 3D objects to create a digital copy of them.
Touch Screen (Resistive, Capacitive and Infra-Red)	They use pressure, conductivity or light to register the touch of a user on a screen. The coordinates of the touch can be calculated.	It is built into a ticket machine to allow a user to select which ticket they would like to buy.

Knowledge 3.2.3. An **output device** is a device that allows the result of the data processing to be seen or heard.

Knowledge 3.2.4. Output devices include:

Actuator	It is a component that outputs an action, often a type of movement, that causes another device to operate.	It can be used in an automated system to move or turn on/off another device, e.g. a light.
Digital Light Processing (DLP) Projector	It is a device that uses light reflected from millions of little mirrors to output an image.	It can be used in a classroom to project an image onto an interactive whiteboard.
Inkjet Printer	This is a device that squirts liquid ink from nozzles to output a document or image.	It can be used in a house to print photographs.
Laser Printer	This is a device that uses a rotating drum and powdered toner to output a document.	It can be used in an office to print letters.
Light Emitting Diode (LED) Screen	This is a screen that uses LEDs as a backlight to output an image.	the screen can be built into a mobile phone.
Liquid Crystal Display (LCD) Projector	This is a device that shines light through crystals and then through a lens to project an image onto a blank wall or screen.	This can be used to project an image in a home cinema system.
Liquid Crystal Display (LCD) Screen	This is a screen that shines light through crystals to output an image.	This can be built into a television screen.
Speaker	This is a device that outputs sound.	This can be built into a mobile phone so one user can hear another user's voice.
3D printer	This is a device that builds layers of material to output a 3D object.	This can be used in medicine to create prosthetic limbs.

Knowledge 3.2.5. A **sensor** is also an input device, and it is used in an **automated system**.

Knowledge 3.2.6. Sensors include:

Acoustic	This type of sensor measures the level of sound in an environment.	These sensors are used in many applications that involve sound. An acoustic sensor can be used in a security system. It can be placed near a window and constantly measure the level of sound. If it captures a reading that shows a sudden increase in sound, this could mean that the window has been broken and a building may be at risk.
Accelerometer	This type of sensor measures acceleration forces. These may be static forces, such as the continual force of gravity. They can also be dynamic forces, such as those created by movement and vibrations.	These sensors are used in a wide variety of devices. Mobile phones use an accelerometer to know which way up it is faced, to automatically turn the screen on and off. They can be used to monitor earthquakes, as they can capture the initial vibrations created. They can also be used in cars to sense when a crash has occurred so that airbags can be inflated.
Flow	This type of sensor measures the amount of liquid, gas or steam that is flowing through or around a certain environment.	These sensors are often used in factories and sites such as nuclear power plants. They make sure that the liquid, gas or steam flows at a constant temperature through an environment, such as a pipe. This makes sure that the pipes don't rupture and break due to too much flowing through them.
Gas	This type of sensor measures the presence and concentration of gas within the immediate atmosphere.	These sensors can be used in people's homes. They can be set to measure a certain gas, such as carbon monoxide. They constantly capture the data in the immediate atmosphere to see if too much carbon monoxide is present, which could endanger the health of anyone living in the home.

Humidity	This type of sensor measures the level of moisture in the immediate atmosphere.	These sensors can be used in farming and agriculture to make sure that the air in areas such as greenhouses has the correct level of moisture to provide the best growing conditions for fruits and vegetables. They can also be used in places such as art galleries, to make sure that the humidity level is constant. Too much or too little can ruin paintings.
Infra-red	This type of sensor measures infrared radiation. This type of radiation can be emitted in different amounts by both objects and people.	These sensors can also be used in security systems. This can be done in two different ways. The device containing the sensor can emit infrared radiation and when this bounces back to the device, the readings can show from the distance it has travelled whether an intruder is present. It can also operate by capturing the infrared radiation emitted by the intruder.
Level	This type of sensor measures whether a substance, such as a liquid, is at a certain level or amount.	These sensors can be used in a car to make sure that essential liquids, such as oil and fuel, do not get too low.
Light	This type of sensor measures the ambient light in a certain environment. It can also measure the presence of direct light, such as a laser beam.	These sensors can be used in automatic lighting systems. Streetlights can be fitted with a light sensor that will allow the light to turn on in the evening when it becomes darker.
Magnetic field	This type of sensor measures the presence of a magnetic field that may be emitted by an object.	These sensors can be used to count how many cars pass through a certain area, for example into a car park. The car will disrupt the Earth's natural magnetic field as it passes over the sensor and the data can be captured by the sensor.

Moisture	This type of sensor measures the amount of water that is present in a substance, such as soil.	These sensors can also be used by farming and agriculture, to make sure that the fruits and vegetables have the best level of water in the soil to help them grow.
pH	This type of sensor measures the pH level of a substance.	These sensors can be used by environmental agencies to make sure that local lakes and rivers are not being polluted.
Pressure	This type of sensor measures the force of pressure that is applied to the sensor or device. This could be the pressure created by a solid object, or it could be created by liquid or gas.	These sensors could also be used in a security system. They can be placed at the base or sides of an opening, such as a window or a door. When that window or door is opened, the pressure will decrease, and the system will recognise that an intruder has entered.
Proximity	This type of sensor measures how close an object is in comparison to the sensor. It does this by emitting electromagnetic radiation or an electromagnetic field and measures the radiation as it returns to see if there are any changes.	These types of sensors can be used in robots in manufacturing. They allow the robots to measure how close they are to different objects when moving around a factory.
Temperature	This type of sensor measures the temperature of an object or substance by either directly touching it or capturing data from the surrounding environment.	These types of sensors are used in air conditioning systems. They allow the temperature of a room to be kept at a certain level.

§3.3 Data Storage

Knowledge 3.3.1. Storage can be divided into two types, **primary storage** and **secondary storage**. The primary is the one that is directly accessed by the CPU, secondary is the one that cannot be directly accessed by the CPU (usually by an I/O device).

Knowledge 3.3.2. **RAM**, random access memory, and **ROM**, read-only memory, are two types of primary storage.

RAM is volatile (temporary). When powered off, the contents are lost.	ROM is non-volatile (permanant). When powered off, the contents remain.
RAM stores data and programs currently in use.	ROM stores the bootstrap and the BIOS which boots the computer.
RAM contents are constantly being changed.	ROM data are fixed. (It is possible to change, but it is difficult and we usually do not do it.)
RAM can be increased by adding components.	ROM is fixed on the motherboard (usually).

Knowledge 3.3.3. Secondary storage is not directly accessed by the CPU (and buses hence slower) but is necessary for larger storage capacity. There are three main types, **magnetic**, **optical** and **solid-state storage**.

Knowledge 3.3.4. Magnetic storage uses platters which are divided into tracks and sectors. Data is read and written using electromagnets.

Optical storage uses lasers to create and read pits and lands.

Solid-state (flash memory) uses NAND or NOR technology. Transistors are used as control gates and floating gates.

Knowledge 3.3.5. **Virtual memory** is the type of memory created for temporary use and is an extension to RAM. **Pages** of data are exchanged between secondary storage and the RAM. When the CPU needs them they will be transferred to the RAM, and when not they will be swapped back to the secondary storage.

Knowledge 3.3.6. **Cloud storage** can be accessed remotely in comparison to storing data locally. Physical servers and storage are needed to store data in cloud storage.

Knowledge 3.3.7. Cloud or local:

You do not need to pay for cloud storage hardware.	You need to pay for local storage hardware.
You are not responsible for the security, but you should make sure the third-party service provider is safe.	You need to make sure the storage is regularly checked and updated for security issues.
You can access data and resources using different devices from anywhere as long as the internet is available. However, if the connection fails, you cannot access it. If the connection is slow, access is slow.	You do not need an internet connection.
You can increase/decrease storage capacity as you need.	You might have redundant hardware, more than you need.

§3.4 Network hardware

Knowledge 3.4.1. **NIC**, or network interface card, is required to connect to the internet. **MAC** address, or media access control address, is paired with a NIC to identify a device on a network. It is assigned by the manufacturer and never changed. It includes a 14-digit hexadecimal with 7 parts using colons (:) to separate them. It is created using the manufacturer code and the serial code.

Knowledge 3.4.2. A **router** sends data to a specific destination on a network, assigns IP addresses, and can connect a local network to the internet.

Knowledge 3.4.3. An **IP address** is allocated by the network and it can be static or dynamic. **Dynamic** IP addresses are the most common and they are assigned every time a device connects to a network. **Static** IP address is usually provided by the Internet Service Provider and will be the same every time you connect. They are unique and can also be used to identify a device on a network. **IPv4** consists of a 32-bit IP address, with 4 numbers (decimal) and full stops. separating them. **IPv6** consists of 128 bits (32-digit hexadecimal) with colons separating them and is capable of creating more IP addresses.

Section 4 Software

§4.1 Types of Software and Interrupts

Knowledge 4.1.1. **Softwares** can be divided into two categories, **System Software** and **Application Software**.

Knowledge 4.1.2. **System Software** provides the services that the computer requires, including the operating system and utility software. **Application Software**provides the services that the user requires.

Knowledge 4.1.3. Functions of an **Operating System** include:

- managing files,
- handling interrupts,
- providing an interface (e.g. graphical GUI, command line, natural language),
- managing peripherals (I/O devices) and drivers,
- managing memory,
- managing multitasking (by interrupts),
- providing a platform for running applications,
- providing system security, and
- managing user accounts.

Knowledge 4.1.4. Examples of **Application Softwares** include:

- word processor,
- spreadsheet,
- database, and
- web browser.

Knowledge 4.1.5. **Application software** runs on **operating system**, which runs on **firmware**, which is loaded by the **bootstrap**, which directly runs on the hardware.

Knowledge 4.1.6. An **interrupt** is a signal to tell the processor to tell it that something needs attention. Two types are **Software Interrupt** and **Hardware Interrupt**.

Knowledge 4.1.7. Examples of software interrupts include:

- Division by 0;
- Two processes attempting to access the same memory location;
- Request for input;
- Output required;
- Data required from memory.

Examples of hardware interrupts include:

- Data input (e.g. keyboard input/mouse click);
- Error from hardware (e.g. printer out of paper);
- Hardware failure;
- Hard drive signal that it has read data;
- New hardware device connected.

Knowledge 4.1.8. Interrupts are handled by an **Interrupt Handler (IH)** with an **Interrupt Service Routine (ISR)**. The process is as follows:

- When the CPU finishes an FDE cycle it checks the interrupt queue.
- It checks whether there is an interrupt with a higher priority than the current task.
- If yes,
  - It stores the current process and fetches the interrupt.
  - It checks the source of the interrupt.
  - It calls (executes) the relative ISR which handles the interrupt.
  - The stored process is returned to the memory or a higher-priority interrupt is fetched.
- If not, it runs another FDE cycle.

§4.2 Types of Programming Language, Translators and Integrated Development Environments (IDEs)

Knowledge 4.2.1. Programming languages include two types, **high-level** and **low-level**.

Knowledge 4.2.2. A high-level programming language uses human-style words for instructions. High-level programming languages are **portable** – you can write it on one device and run it on another.

Knowledge 4.2.3. Low-level programming languages can be further divided into **machine code** and **assembly language**. Machine code, as its name suggests, is machine-specific i.e. it is **non-portable**. Assembly uses **mnemonics** to represent code which is in the middle.

Knowledge 4.2.4. High-level language v.s. low-level language:

High-level languages	Low-level languages
Easier for users to understand, read, write and ament.	More difficult to do so.
Easier to debug.	More difficult to do so.
Machine independent (portable).	Machine dependent (non-portable).
It must be converted to a low-level language to run.	Machine code does not require converting, assembly needs to be assembled but is significantly faster than the execution of a high-level language.
One statement can represent many low-level instructions.	Multiple statements are required to represent just one high-level statement.
Cannot directly manipulate the hardware.	Can directly manipulate hardware, e.g. writing to specific memory locations, which makes it more time and space efficient.

Knowledge 4.2.5. A **translator** is required to translate programs to machine code to execute.

Knowledge 4.2.6. In a high-level language, you can use **interpreter** or a **compiler** to translate the instructions.

An **interpreter** translates and executes the code line-by-line. It stops where there is an error. It is more useful for program writing, but not for whole-system testing since it needs to interpret everything every time code is executed.

A **compiler** translates the whole file all at once and produces an **executable file** which can be directly executed. It produces an error report of the whole program. It is suitable for whole-program testing.

Knowledge 4.2.7. Interpreters v.s. compilers:

Interpreter	Compiler
Translates one line of code into machine code and then executes it.	Translates all lines of code into machine code, before executing the program.
Reports a syntax error as soon as it is picked up and stops the program until it is corrected.	Reports all syntax errors at the same time, the program is not run until all errors are corrected.
Useful when writing a program.	Useful when a program has been finished and is ready for testing or distribution.
Code needs to be re-translated each time the program is run.	Code does not need re-translating.
Does not produce an executable file.	Produces an executable file.
Source code is required to run.	Source code is not required.
Interpreter software is required to run.	No other software is required.
Partially testing is available.	Whole section of code must be completed to test.

Knowledge 4.2.8. An **Integrated Development Environment (IDE)** is an application software for you to write and test the code, including most/all software necessary. It includes an **editor**, the **translator** and also the **run-time environment** where the interface is shown while running.

Knowledge 4.2.9. IDE functions include:

- code editors;
- run-time environment;
- translators;
- error diagnostics;
- auto-completion;
- auto-correction; and
- prettyprint.

Section 5 The Internet and its uses

§5.1 The Internet and the world wide web

Knowledge 5.1.1. The **internet** is the infrastructure, especially the cable. It is just a type of WAN which is very special that covers the whole world. The **world wide web** is the collection of **websites** and **web pages** accessed using the internet

Knowledge 5.1.2. A **URL** (uniform resource locator) is a text-based address for a web page. It contains the protocol, the domain name and the web page/file name.

Knowledge 5.1.3. **HTTP** (hypertext transmission protocol) is a protocol used to transmit requests and the results of the requests between the web browser and the web server.

HTTPS (hypertext transfer protocol secure) adds a layer of security by encrypting data using **digital certificates** where owners will need to apply from a **certificate authority**.

Knowledge 5.1.4. The **domain name server (DNS)** is a special kind of server that stores the domain names with their equivalent IP address as a form of a database.

Knowledge 5.1.5. The following is the process involved in retrieving web pages (for a standard HTTP protocol):

- The user opens the web browser and enters the URL into the **web browser**.
- The web browser sends the URL to the **DNS**.
- The DNS server returns the IP back to the web browser.
- The web browser receives the IP address and uses the HTTP protocol to request the website to the web server.
- The web server returns **Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript** back to the browser using **HTTP**.
- The web browser then renders the website using those data.

HTTPS works similarly.

- Before the web browser send a request to ask for the web pages, it asks for a digital certificate.
- The web server sends a certificate to the web browser.
- The web browser checks whether the certificate is authentic.
- If yes, the web browser will continue transmission just as before but encrypted.
- If no, the web browser will report that the website is not secure.

Knowledge 5.1.6. HTTPS works with a layer of security which may be **SSL (Secure Sockets Layer)** or **TLS (Transport Layer Security)**. These are the protocols used to encrypt.

Knowledge 5.1.7. As described before, the web browser **renders** the data to let you see the webpage. Furthermore, it provides the following functions:

- storing bookmarks and favourites,
- recording user history,
- allowing the use of multiple tabs,
- storing cookies,
- providing navigation tools, and
- providing an address bar.

Knowledge 5.1.8. Browsers use **cookies** to store certain data. Data stored includes

- saving personal details,
- tracking user preferences,
- holding items in an online shopping cart, and
- storing login details.

Knowledge 5.1.9. Cookies can be divided into two types, **session cookies** and **persistent cookies**. The former refers to cookies that will be deleted when the web page closes, while the latter will be stored until an expiration date.

§5.2 Digital currency

Knowledge 5.2.1. A **digital currency** is one that only exists electronically and is exchanged by computers. Examples of use include **credit cards**, **mobile phone**, and **smart watches**.

Knowledge 5.2.2. One type of digital currency is called **cryptocurrency**, which is a digital currency that is managed by a delocalised system (no central authority) – bitcoin is a typical example of it. It is encrypted to prove that the transaction exists.

Knowledge 5.2.3. **Blockchain** is used to keep track of the payments, which is a chain of blocks/records that shows all transactions for a specific currency.

It uses a **digital ledger**, which is a public record of all payments made with cryptocurrency. The records cannot be altered since the whole ledger is encrypted, but each record has a unique digital signature with time and date to prove that it exists.

§5.3 Cyber security

Knowledge 5.3.1. **Brute force attack** is an attempt to guess a password by trying all possibilities (all combinations). It is an automated process.

Solutions include

- Strong passwords,
- Limiting the number of login attempts,
- Biometric 'passwords', and
- 2 Factor Authentication (2FA).

Knowledge 5.3.2. **Data interception** is a process using pocket sniffer software to intercept data packets as they move through a network.

Solutions include

- Use encryption SSL/TLS (HTTPS).

Knowledge 5.3.3. **Distributed denial of service (DDoS) attack** is a process of sending too many requests to a server in an attempt to make it crash. Distributed stands for requests being sent by a network of computers (botnet) infected by malware and such computers are called bots/zombies.

Solutions include

- Setting up a **proxy server** which acts like a filter/firewall of requests to prevent the real web server from crashing.
- Using **anti-malware** software to prevent your computer from becoming a bot.

Knowledge 5.3.4. **Hacking** stands for the act of trying to gain unauthorised access to data by exploiting a vulnerability, and the person acting is called a hacker.

Solutions include

- Firewall, which manages outgoing and ingoing connections;
- Automatic OS updates;
- Strong passwords; and
- 2FA.

Knowledge 5.3.5. **Malware** stands for malicious software designed to disrupt a computer or its data.

Examples include

Virus	Downloaded onto your hard drive which replicates itself and corrupts stored data or uses up all available memory, causing it to slow down or crash.
Worm	Similar to a virus, but looks for vulnerability holes in a network to use to replicate itself, and will clog up the bandwidth of a network and slow it down.
Spyware	Downloaded onto your hard drive and designed to record actions on the computer, such as keyloggers which records all key presses. Data is sent to the perpetrator where it is analysed to identify patterns in data, which could reveal passwords. This could allow access to online accounts for fraud and identity theft.
Trojan Horse	A computer software that is used to disguise malware. It is designed to look harmless (e.g. application or game) but contains a virus/worm.
Adware	A computer program designed to create pop-up and banner adverts when online, which can be irritating and frustrating. When you click on the links the creaters are given money.
Ransomware	It is designed to encrypt your data and stop you from gaining access to it, and the creators will demand a ransom for data to be decrypted. They will threaten to release and leak your stored data if the ransom is not paid.

Solutions include

- encrypting data,
- data trackup,
- firewall, and
- anti-malware software.

This is an example of an effective cybersecurity system consisting of multiple layers.

Knowledge 5.3.6. **Pharming** is malware that directs you to a fake website when you enter a genuine URL.

Solutions include

- anti-malware software, and
- users visually check websites.

Knowledge 5.3.7. **Phishing** is the process of sending an email that encourages you to click on a link to a fake website.

Solutions include

- Delete.

Knowledge 5.3.8. **Social engineering** is an attempt to manipulate or deceive people to release security information.

Solutions include

- access control,
- high privacy level on social media, and

- awareness.

Knowledge 5.3.9. You are expected to know the following solutions and come up with a solution to a given situation:

- access levels
- anti-malware including anti-virus and anti-spyware
- authentication (username and password, biometrics, two-step verification)
- automating software updates
- checking the spelling and tone of communications
- checking the URL attached to a link
- firewalls
- privacy settings
- proxy-servers
- secure socket layer (SSL) security protocol.

Section 6 Automated and emerging technologies

§6.1 Automated Systems

Knowledge 6.1.1. An **automated system** performs actions without interference with humans and is widely used. It usually involves sensors, microprocessors and actuators (which are introduced previously).

Knowledge 6.1.2. An automated system works as follows:

- Sensors retrieve data from the environment and output analogue signal to the ADC.
- The ADC (analogue-to-digital) converter converts this signal to digital and outputs it to the microprocessor.
- The microprocessor reads the data, compares it with stored values (or a simple algorithm) and decides whether and how to use the actuator to affect the environment.
- It outputs a signal to the DAC (digital-to-analogue) converter and is converted to an analogue signal.
- The actuators receive the analogue signal and cause something to happen, e.g. opening a door, or switching on a fan/a heater.

The above-described process is repeated.

Knowledge 6.1.3. This is an example of a **closed feedback loop**:

- The use of an actuator changes the environment;
- The change in environment is measured by sensors; and
- The sensor readings are used to influence future decisions about the use of the actuator.

Knowledge 6.1.4. Do consider the following while being asked questions related with automated systems:

- Initial cost (the cost for the infrastructure),
- running cost (e.g. cost of electricity and maintenance),
- safety (usually safer since people may be distracted),
- replacing people's jobs (usually both replaced and created),
- continous work all day every day (the result of it), and
- precision (usually higher since no human errors are made).

The given scenarios will be some that you are familiar with, i.e. limited to

- industry,
- transport,
- agriculture,
- weather,
- gaming,
- lighting, and
- science.

§6.2 Robotics

Knowledge 6.2.1. **robotics** is the science of the design, construction and operation of robots.

A **robot** is:

- programmable,
- semi-autonomous (need human supervision only),
- mechanical device that performs an action (often human action),
- using sensors to measure the environment,
- using actuators to move themselves/other objects, and
- using microprocessors to control what they do.

Knowledge 6.2.2. Robots may be used in multiple fields, including (with examples):

- industry e.g. print spraying,

- transport e.g. driverless trains,
- agriculture e.g. moving lawns,
- medicine e.g. surgery,
- domestic e.g. washing machines and drones,
- entertainment e.g. dance, music and light.

§6.3 Artificial Intelligence

Knowledge 6.3.1. **Artificial Intelligence (AI)** is the development of programs to simulate intelligent human behaviour, with certain functions such as:

- image recognition,
- speech recognition,
- natural language,
- computer games, and
- diagnostic systems.

Knowledge 6.3.2. Four key features of AI are:

- collection of data,
- a set of programmed rules,
- the ability to reason, and
- the ability to learn and adapt. This may not be true for all AI systems but ML (machine learning) will do so as we will explain in the next point.

Knowledge 6.3.3. **Machine learning** (ML) is the process of adaptive behaviour as a result of experience (i.e. data).

It can be divided into two types: **supervised** and **unsupervised**. Supervised stands for the user telling the program what the data means (labelled), while unsupervised for the opposite, which requires the program to identify patterns.

Knowledge 6.3.4. **Expert systems** try to replicate the knowledge of a human expert to diagnose a problem.

The architecture includes

- UI (user interface): where they ask questions and you answer;
- Inference engine: deciding which question to ask next;
- Rule base: linking facts; and
- Knowledge base: a list of facts.

Section 7 Algorithm design and problem-solving

Knowledge 7.0.1. The **program development life cycle** includes four stages,

- analysis: abstracting, decomposition, and identification of problem and requirements;
- design: further decomposition, creating structure diagrams, flowcharts and pseudocode;
- coding: writing program code and iterative testing;
- testing: testing program code with test data.

Knowledge 7.0.2. **Sub-systems** make up computer systems and they are even made up of further sub-systems.

We could decompose a problem into four main parts: inputs, processes, outputs and storage.

We could use a structure diagram (which looks like a tree), a flowchart or pseudocode to design a solution.

Knowledge 7.0.3. **Flowchart** is a diagram representing an algorithm, and its syntax involves the following:

- Flow line, which is an arrow representing passing between states;
- Process, a rectangle representing some operations being done;
- Subroutine, a rectangle with two vertical lines close to the sides, representing calling a separate flowchart;
- I/O, a parallelogram representing input/output;
- Decision, a rhombus used to represent a Yes/No a.k.a. True/False decision being made based on a condition; and
- Terminator, an oval used to fill in Start or Stop.

Knowledge 7.0.4. Note: time complexity is not required, just for reference how quick each algorithm is.

- A **Linear Search** algorithm searches for a value (and its position) in a given list (usually unsorted, if sorted we usually use binary search which has a time complexity of $\mathcal{O}(\log n)$), with time complexity of $\mathcal{O}(n)$ (not required but good to know);
- A **Bubble Sort** algorithm sorts elements by comparing neighbouring elements and swapping them into the correct sequence, which has a time complexity of $\mathcal{O}(n^2)$. (Faster algorithms include merge sort and quicksort which has a time complexity of $\mathcal{O}(n \log n)$.)
- A **totalling** algorithm finds the sum of elements in a list by iterating. It has a time complexity of $\mathcal{O}(n)$ but with pre-processing the time complexity could be reduced to $\mathcal{O}(1)$.
- A **counting** algorithm finds the number of elements (with a certain value) in a list. This should have a time complexity of $\mathcal{O}(n)$, but pre-processing data (during input) gives us a time complexity of $\mathcal{O}(1)$.
- Finding **maximum, minimum** is similar to the totalling (we do this by iterating) and has a time complexity of $\mathcal{O}(n)$ and $\mathcal{O}(1)$ without pre-processing.
- Finding **average** is just dividing the total by the number of elements.

Knowledge 7.0.5. **Validation check** is the check on data to make sure is reasonable. Checks inlude:

- Range check,
- Length check,
- Type check (e.g. str or int),
- Presence check (e.g. NULL),
- Format check (e.g. 1 or one),
- Check digit.

Knowledge 7.0.6. **Verification** is the check to make sure the input data is correct. Checks include visual checks and double-entry checks.

Knowledge 7.0.7. **Testing data** is the data you input into a program to test whether it works properly or not. Types include:

- Normal: data that the program should accept;
- Abnormal: data that the program should not accept;
- Extreme: data at the edge of what's allowed;
- Boundry: data at two edges of what's allowed (just allowed and just disallowed).

Section 8 Programming

§8.1 Programming concepts

Knowledge 8.1.1. Data types include integer, real (a.k.a. floating point/float/double), char (character), string and boolean (T/F). They are `INTEGER`, `REAL`, `CHAR`, `STRING`, `BOOLEAN` respectively.

The way to declare a variable is as follows in pseudocode:

```
DECLARE <identifier>:<datatype>
```

and the way to declare a constant is:

```
CONSTANT <identifier> ← <value>.
```

Knowledge 8.1.2. The sign ← is called **assignment**, which is used as

```
<identifier> ← <value>.
```

Knowledge 8.1.3. **Input and output** are two very important things for a program. Their syntax is

```
INPUT <identifier>
```

and

```
OUTPUT <value(s)>.
```

Knowledge 8.1.4. **Sequence** is the idea of executing code in a certain order, it does not have a specific syntax - just don't write code the other way around.

Knowledge 8.1.5. **Selection** is when logical operators are used to determine which branch a program goes to. Two selection syntaxes are if and case, with syntax as follows:

```
IF < condition > THEN
    < statements >
ELSE IF < condition > THEN
    < statements >
...
ELSE
    < statements >
ENDIF
```

and

```
CASE OF < identifier >
    < value 1 >:< statement >
    < value 2 >:< statement >
    ...
    OTHERWISE :< statement >
ENDCASE
```

The first is useful when a small number of selections is necessary while the second is useful when there are a lot of cases and each case only requires one statement (important! or it will look weird).

Notice that the otherwise is optional but I strongly recommend writing it.

Knowledge 8.1.6. **Iteration** or **loop** is a structure when a statement (or multiple statements) is run multiple times. We have **for (count-controlled)**, **while (pre-condition)** and **repeat until (post-condition)** loops.

```
FOR < identifier > ← < value1 > TO < value2 > STEP < increment >
    < statements >
NEXT < identifier >
```

Note that the step increment in for loop is optional and usually does not appear (unless you want to do a loop reversed).

```
WHILE < condition > DO
    < statements >
ENDWHILE
```

```
REPEAT
    < statements >
UNTIL < condition >
```

Knowledge 8.1.7. **String manipulation** includes length, substring, upper case, and lower case. We use

$$\texttt{LENGTH}(<\texttt{identifier}>)$$

to find a string,

$$\texttt{LCASE}(<\texttt{identifier}>)$$

to convert everything to lowercase,

$$\texttt{UCASE}(<\texttt{identifier}>)$$

to convert everything to uppercase, and

$$\texttt{SUBSTRING}(<\texttt{identifier}>, <\texttt{start}>, <\texttt{length}>)$$

to find the substring from the start position with a certain length.

Notice that generally string starts at position 1 (by syllabus pseudocode syntax section) but the syllabus also states that you can treat it as start position 0.

Knowledge 8.1.8. $+, -, *, /, \wedge$(raised to the power of) are required mathematical arithmetic operations. Furthermore,

$$\texttt{DIV}(<\texttt{identifier 1}>, <\texttt{identifier 2}>)$$

gives the integer division result (with the fractional part discarded), and

$$\texttt{MOD}(<\texttt{identifier 1}>, <\texttt{identifier 2}>)$$

gives the remainder of the division result.

Knowledge 8.1.9. $=, <, >, <=, >=, <>$(not equal to) are required logic operators (which are used in conditions).

Knowledge 8.1.10. `AND`, `OR`, `NOT` are required boolean operators. They can be used to combine conditions together to give a result.

Knowledge 8.1.11. **Nested statements** are when you put one layer of iteration/selection around another (for example, in a bubble sort).

Knowledge 8.1.12. **Subroutine** is a self-contained sub-module of a code (i.e. a sub-system) which can be **called** in the main program.

Procedures operates on something while **functions** produces a return value.

Parameters are inputs to subroutines.

To define a procedure,

```
PROCEDURE < identifier > (< param1 >:< datatype >, < param2 >:< datatype >, ...)
    < statements >
ENDPROCEDURE
```

and to call it,

$$\texttt{CALL } <\texttt{identifier}> (\texttt{Value1}, \texttt{Value2}, \ldots).$$

To define a function,

```
FUNCTION <identifier> (<param1>:<datatype>, <param2>:<datatype>, ...)
                RETURNS <datatype>
    <statements>
    RETURN <value>
ENDFUNCTION
```

and to call it,

```
<identifier> (Value1, Value2, ...).
```

Knowledge 8.1.13. **Local** variables are declared and used in a loop/branch/subroutine (and are invalid outside it), and **global** variables are the ones which can be used anywhere.

Knowledge 8.1.14. The function

```
ROUND(<identifier>, <places>)
```

is used to round a certain variable to a certain decimal place.

The function

```
RANDOM()
```

returns a random value between 0 and 1 inclusive.

Knowledge 8.1.15. It is a good habit to make variable/constant/array/procedure/function names (identifiers) understandable and to use procedures and functions where necessary and suitable.

§8.2 Arrays

Knowledge 8.2.1. **Arrays** are just simply a list of elements. **2-D Arrays** is just simply a list of lists.

To declare a 1-D array we use

```
DECLARE <identifier>: ARRAY[<1>:<u>] OF <data type>
```

and use

```
<identifier> [<index>]
```

to call it. To declare a 2-D array we use

```
DECLARE <identifier>: ARRAY[<l1>:<u1>, <l2>:<u2>] OF <data type>
```

and use

```
<identifier> [<index1>, <index2>]
```

to call it.

At a GCSE level, a 2-D array is a table, but at a higher level, it is not necessary.

Notice that in the syllabus it said that the beginning index can be either 0 or 1.

§8.3 File Handling

Knowledge 8.3.1. **File handling** is important, to save and read data easier externally into an external file. There are two types of file modes, read (READ) and write (WRITE).

To open a file we write

$$\texttt{OPENFILE} < \texttt{File identifier} > \texttt{FOR} < \texttt{File mode} >,$$

and to read/write from a file we do

$$\texttt{READFILE} < \texttt{File identifier} >, < \texttt{Variable} >$$

and

$$\texttt{WRITEFILE} < \texttt{File identifier} >, < \texttt{Variable} >$$

respectively.

To close a file we write (it is a good habit to close after using)

$$\texttt{CLOSEFILE} < \texttt{File identifier} >.$$

§8.4 Sample Pseudocode for Required Algorithms

Knowledge 8.4.1. An example code for bubble sort is:

```
FUNCTION BubbleSort (length : INTEGER, list : ARRAY[1 : len] OF REAL)
            RETURNS ARRAY[1 : len] OF REAL
   DECLARE swapped : BOOLEAN
   swapped ← TRUE
   WHILE swapped
      swapped ← FALSE
      FOR i ← 1 TO len − 1 DO
         IF list[i] > list[i + 1] THEN
            DECLARE tmp : REAL
            tmp ← list[i + 1]
            list[i + 1] ← list[i]
            list[i] ← tmp
            swapped ← TRUE
         ENDIF
      NEXT i
   ENDWHILE
   RETURN list
ENDFUNCTION
```

Knowledge 8.4.2. An example code for linear search and counting is:

```
FUNCTION LinearSearch (length : INTEGER, list : ARRAY[1 : len] OF REAL, val : REAL)
        RETURNS INTEGER, BOOL, INTEGER
    // This program will return the LAST index found just for counting.
    // To return the first one just RETURN after found.
    DECLARE count : INTEGER
    DECLARE found : BOOLEAN
    DECLARE indexFound : INTEGER
    count ← 0
    found ← FALSE
    indexFound ← −1
    FOR i ← 1 TO len DO
        IF list[i] = val THEN
            count ← count + 1
            indexFound ← i
            found ← TRUE
        ENDIF
    NEXT i
    RETURN count, found, indexFound
ENDFUNTION
```

Knowledge 8.4.3. An example code for finding the total, maximum, minimum, and average of a list:

```
FUNCTION TotMaxMinAvg (length : INTEGER, list : ARRAY[1 : len] OF REAL)
                RETURNS REAL, REAL, REAL, REAL
    DECLARE tot : REAL
    DECLARE max : REAL
    DECLARE min : REAL
    DECLARE avg : REAL
    tot ← 0
    max ← −INF
    min ← INF
    FOR i ← 1 TO length DO
        tot ← tot + list[i]
        IF list[i] > max THEN
            max ← list[i]
        ENDIF
        IF list[i] < min THEN
            min ← list[i]
        ENDIF
    NEXT i
    avg ← tot/len
    RETURN tot, max, min, avg
ENDFUNCTION
```

Section 9 Databases

Knowledge 9.0.1. A **database** is a collection of logically organised data, often arranged in **tables**.

Knowledge 9.0.2. The **rows** in a database is called a **record**. The **columns** in a database is called a **field** or an **attribute**.

Knowledge 9.0.3. A **primary key** is a field that uniquely identifies a record.

Knowledge 9.0.4. Similar to pseudocode, **real, integer, text (alphanumeric), character, date, time, boolean** data can be stored in databases.

Knowledge 9.0.5. We search for data by **query**, using **structural query language (SQL)**.

Knowledge 9.0.6. A basic SQL script consists of the following:

```
SELECT list of fields
FROM name of table
WHERE condition
ORDER BY field ASC/DESC
```

Knowledge 9.0.7. In the `SELECT` field, we could also use `SUM(field)` to find the sum of the field and use `COUNT(field)` to find the number of records which meet the requirements. (In fact, it finds the number of records with a non-empty field but we do not need to consider that at this stage.)

Section 10 Boolean logic

Knowledge 10.0.1. We are expected to know the following logic symbols:

- NOT. A verticle line at the input, a tip with a circle at the output.
- AND. A vertical line at the input and a curve at the output.
- OR. A curve at the input and a curve at the output.
- NAND. AND gate with a circle on the output.
- NOR. OR gate with a circle on the output.
- XOR (EOR). (E stands for Exclusive) OR gate with an extra curve at the input.

Knowledge 10.0.2. Only NOT gate has a single input; all the rest has two inputs.

Knowledge 10.0.3. The logic gates create the following outputs:

- NOT. 1 iff 0.
- AND. 1 iff (1, 1).
- OR. 0 iff (0, 0).
- NAND. 0 iff (1, 1).
- NOR. 1 iff (0, 0).
- XOR. 1 iff different.

Note: iff stands for if and only if.

Afterwords

I hate typing Pseudocode like this in LaTeX!

This sheet took me some time to populate, but I would genuinely like to share my understanding of GCSE CS (and beyond) with all of you.

My deepest thanks to all Computer Science teachers I have met and my friends who helped and supported me with producing this.

I would like to give special thanks to Dr Mulvey, my current Computer Science teacher, and Dr Zhang, my tutor for providing me with all this knowledge.

Finally, I really hope this helped you to gain a better understanding of Physics. Feel free to email eason.syc@icloud.com to send me a feedback.