Zig Zag Education

OCR GCSE Computing Revision Booklet

Unit A451: Computer Systems and Programming

1. Computer Systems....................1

2. Computing Hardware....................5

3. Software....................13

4. Data Representation....................18

5. Databases....................30

6. Communications and Networking....................36

7. Programming....................45

Importance of Computers

Application	Importance
Safety	Computers are at least partially responsible for critical operations such as monitoring and controlling nuclear reactions within nuclear power stations.
Travel	The ways in which computers aid travel vary from satellite navigation to online ticket purchases to autopilots that help to fly planes.
Business	Many people depend on computers for business transactions and increasingly use the Internet. This might involve moving funds around the world instantly or transmitting any electronic product, such as books or music.
Education	Computers can aid distance learning (where people access learning resources online) and learning in the classroom. Computers can help to communicate information as well as administer and mark tests. Computers are also important in maintaining information on students such as grades, attendance and punctuality.
Retail	Computers connected to barcode scanners and scales can make shopping faster and have contributed to self-service tills. They also keep track of stock, often re-ordering automatically when needed.
Entertainment	Hand-held and desktop devices allow users to play games, read books, watch movies and listen to music, all electronically.
Communication	As well as emailing, computers facilitate forums, instant messaging, social networks, teleconferencing and videoconferencing.

If your code contains a syntax error, it will either not compile (if you are using a compiler) or it will crash when the erroneous line is reached (if you are using an interpreter).

A **logic** error does not cause a program to crash, although it does cause the program to do something it shouldn't. It might, for example, produce an incorrect result to a calculation. Here is an example concerning calculating a score as a percentage:

```
Display (score/possible) * 1000
```

In calculating percentages, the last stage is multiplying by one *hundred*, not one *thousand*. The program will not crash, but it will display the wrong answer.

Logic errors, because they do not cause the program to crash, are often harder to find than syntax errors, and they are only usually identified by detailed testing.

Handling Data in Algorithms

Every piece of data that a program needs, no matter how small, must be stored somewhere. Within a program, three features exist for storing data – **variables, constants** and **arrays.**

A variable is a space for a single piece of data, and programs usually have lots of variables, each one of which has a **data type**:

Boolean Can be either true or false – requires only one bit of storage. Examples of Boolean data might include whether a student has passed an exam or not, or whether or not an iteration (loop) is going to run again.

Character A single letter, number, punctuation mark, space, etc. – requires one byte. Examples of character data might include 'M' or 'F' to represent gender, or 'P', 'D' or 'E' as a shorthand for the type of fuel a car uses.

String To store any number of characters – variable length. Examples of strings might be a person's name, address or postcode.

Integer Whole numbers – the more storage space is available, the higher the largest number that can be stored. A person's age is usually stored as an integer, so might the quantity of an item a shop has in stock.

Real Decimal numbers – more storage space would be required for either larger numbers or numbers stored to a higher degree of precision. Pi must be stored as a real number, as well as anything that might involve fractions, such as distances.

A constant is exactly the same as a variable, and can be of any of the same data types. The only difference between them is that variables change, while constants, once set, stay the same. Constants are used to represent data that doesn't change very often, such as:

- VAT rate (this does change, but not very often)
- The value of π (pi) (this will never change)

Ethical, Environmental and Legal Considerations

Ethical

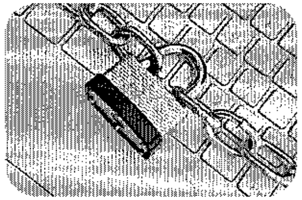
The word 'ethical' refers to something as being right or wrong, although not everyone agrees about what is right and what is wrong, even in something as specific as computing. Consider the following:

- Would it be ethical (right) for a government to store an electronic profile of the DNA of its entire population, of criminals, immigrants, etc.?
- Is it ethical to use cheap labour in foreign countries to produce software less expensively, or would it be more ethical to use local labour?
- What are the ethical challenges of living in a world where not everyone has access to computers, either for financial-, political- or disability-related reasons?

There are no simple answers to these questions; any one of them could start a debate.

Legal

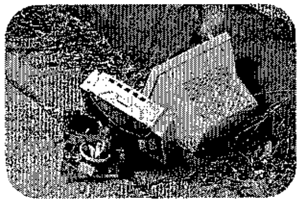
There are four laws that computer users need to be aware of:



- **Computer Misuse Act** – This makes **hacking** illegal. Hacking is gaining access to computer material that you are not allowed to have access to, possibly by stealing a password.
- **Data Protection Act** – If you (or the company you work for) store the personal data of anyone, you are required to keep it secure and up to date.
- **Copyrights, Designs and Patents Act** – It is illegal to make a copy of any publication (including written works, music and computer programs) without the permission of the copyright owner.
- **Health and Safety at Work Act** – This law makes employees responsible for the safety of their staff, as well as staff responsible for the safety of other staff. In terms of computers, the following are considerations:
    - Trailing cables that are a trip hazard
    - Too much time spent at a computer which can cause eye strain from looking at the screen or repetitive strain injury (RSI) from typing or using the mouse
    - A poorly positioned chair, which can cause back problems
    - A monitor not at eye level, which can cause neck problems

Environmental

In terms of the environmental impact of computers, there are some clear negatives:



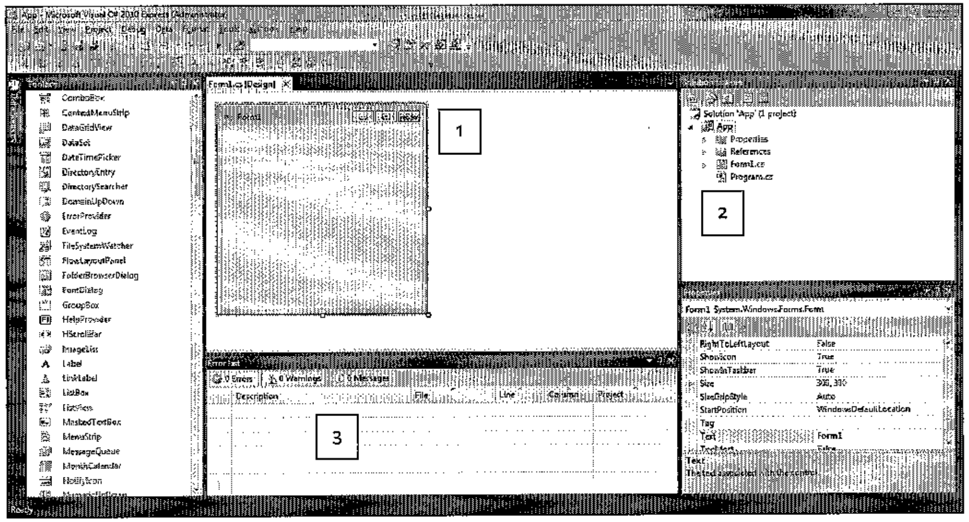
- The production of computers, mobile phones and other such devices requires raw materials as well as energy.
- Computers require energy to run, which may mean the burning of fossil fuels.
- Old computers are not easy to dispose of and are not always recycled as they could be, resulting in more waste going to landfill.

There are also benefits:

- Communication sent electronically does not waste paper or ink.
- People who work from home (which is made possible by computers) do not have to burn fossil fuels driving to work.
- Industrial processes that are computer-controlled, rather than overseen only by a human, are more efficient and, therefore, less polluting.

The Integrated Development Environment (IDE)

The IDE is a program that brings together a broad range of software development tools to allow the programmer to write programs without having to rely on lots of separate utilities. Below is an IDE that is used for developing C# (C 'sharp') programs:



Some features present within this IDE are labelled:

1. The **editor**. In this particular environment, this can switch between a text editor, where code is entered, and a GUI (graphical user interface), where you can drag and drop controls such as text boxes and buttons.
2. The **explorer**. A program may be made up of various different parts, perhaps including different screens that will appear and disappear as the program runs. These are organised here.
3. **Error diagnostics**. If any errors within the code are detected, they are brought to the attention of the user. Some IDEs even offer the user possible solutions to an error.

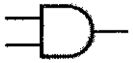
- You might also find **auto-documentation** features, which automatically create part of the maintenance documentation for your program as you type code.
- The **run-time environment** allows you to run your program as you are creating it, by pressing a 'play' button or equivalent.
- **Translators** form part of the IDE as well (see section above).

Binary Logic

Data in computers is represented in binary form (0s and 1s only – see the 'Data Representation' chapter). There are several reasons for this:

- Computer components often have two states that can easily model 0 and 1. A magnet, on a hard disk, can be aligned either north or south and a circuit can be either open or closed.
- Binary is **high tolerance**; it is unlikely that a computer could mistake a 1 for a 0, in the same way that 4.26 could be mistaken for 4.27 if a computer used, say, different voltages to represent more than two numbers.

To use binary numbers, the main memory of a computer will contain literally billions of circuits that can each be either open or closed. **Logic gates** allow calculations and other operations to take place by reading one or more inputs (each of which would be either 0 or 1) and providing outputs (also a 0 or 1). The three primary logic gates are **AND**, **NOT** and **OR**.

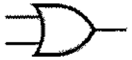


AND

This gate has two inputs and one output. The output will be a 1 only if both inputs are 1 (the first input AND the second input), otherwise the output will be a 0. The **truth table** for this gate is shown below. A truth table shows what the output of a logic circuit will be for every possible combination of inputs. The inputs will be referred to as A and B.

A	B	A · B
0	0	0
0	1	0
1	0	0
1	1	1

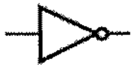
The first two columns are the inputs and the last is the output. The dot (·) simply means 'AND'.



OR

This gate also has two inputs and one output. If either or both of the inputs are 1, the output will be 1 (the first input OR the second input, or both), otherwise it will be 0. This logic gate has a slightly different truth table. Note the confusing symbol for 'OR' (+):

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1



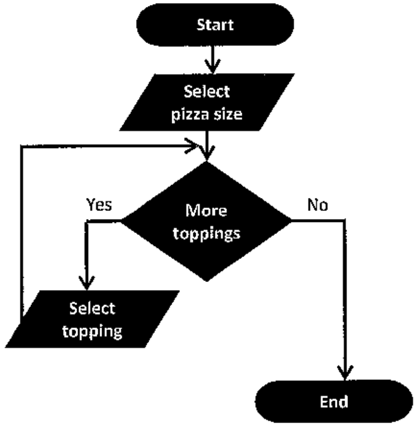
NOT

A 'NOT' gate has only one input and one output. Whatever the input is, the output will be the opposite. This means the truth table is smaller:

A	¬A
0	1
1	0

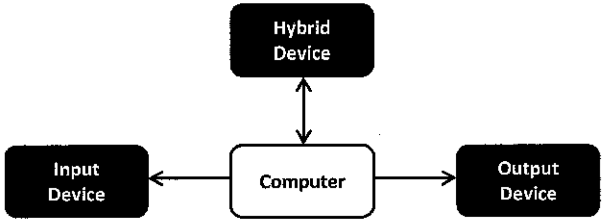
Iteration takes place when one or more instructions are executed more than once. **Iteration** is also known as repetition. Iteration is taking place when program code (or pseudocode) contains the words LOOP, FOR, WHILE or REPEAT. This program snippet might be for an online system for ordering pizza:

```
Select pizza type
WHILE more toppings required
      Select pizza topping
REPEAT
```



Input and Output

An input device is one that is used to introduce data into a computer. An output device is one that allows data to leave a computer, usually in a form intelligible to a human user. Some devices are hybrid devices, in which case they do both of these things.



Input Devices

- **Keyboard** – familiar to most users, keyboards are used in a large number of data-entry applications. Skilled users can enter data at a high speed, but errors can be quite common.
- **Mouse** – a pointing device central to WIMP interfaces (see GUI section of 'System Software').
- **Barcode reader** – the image of a barcode translates into numbers which are read into a system and searched for in a database to find data on the object displaying the barcode. Most commonly used in supermarkets, these systems are also used in manufacturing to keep track of components on a production line or in storage.
  Barcodes do not require high-quality printing techniques and can be put onto surfaces that are not flat. If the barcode is damaged, a human user can simply input the number itself into the system manually.

- **Magnetic stripe reader** – data is stored magnetically on an area of a flat surface. When the flat surface is read by a specific reader, the data is retrieved. This technology is widely used in credit/debit cards and tickets for trains or the London Underground.
- **Light pen** – used to interact directly with a screen (can be a television screen or a computer VDU). This technology can be used for drawing designs straight onto a screen or for quickly navigating through a menu structure. The pen recognises which pixel of the screen it is pointing at and sends a corresponding signal to the computer to change the display in some way.

- **OMR** (optical mark reader) – scans a piece of paper to look for marks in pre-determined places, usually indicated to the user by way of pre-printed boxes to fill in. This technology is used widely in the National Lottery, as well as in administering multiple-choice tests or questionnaires.
- **Scanner** – scanners can be black and white, greyscale or full colour. A scanner shines beams of light at a document and measures the colour and intensity of the light that bounces back. Scanners are often used in conjunction with optical character reader technology.
- **OCR** (optical character reader) – as well as scanning documents, an OCR device can recognise and interpret characters on the page. This technology is most reliable when dealing with printed characters, but handwritten characters can also be recognised. This is highly useful in scanning postcodes at mail sorting offices.

7. Programming

Algorithms and Flow Control

Computer programs are usually written to solve a problem, and an **algorithm** is a set of step-by-step rules, written for a computer, that explain how the problem is to be solved. There are two methods that are commonly used to describe algorithms – pseudocode and flow charts.

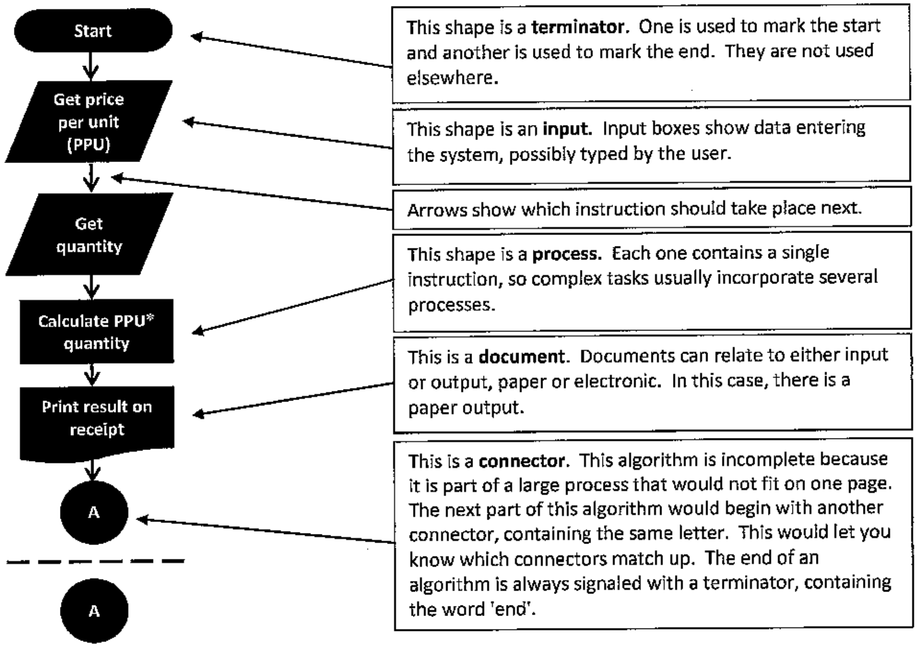
Pseudocode

Pseudocode looks like a programming language, but it isn't. It's a cross between English and a generic-looking programming language. It is useful for some, because it is a halfway point between describing the program in English and having a complete program written in a language such as Visual Basic, Java or Python. A pseudocode algorithm for calculating the cost of a purchase at a till might be as follows:

```
Get price per unit
Get quantity
Multiply price per unit by quantity
Display result
```

Flow Chart

A flow chart does the same job as pseudocode in defining an algorithm, but it is a diagrammatical representation instead. The flow chart segment below defines the same part of a program as the pseudocode above, but it looks very different:



This shape is a **terminator**. One is used to mark the start and another is used to mark the end. They are not used elsewhere.

This shape is an **input**. Input boxes show data entering the system, possibly typed by the user.

Arrows show which instruction should take place next.

This shape is a **process**. Each one contains a single instruction, so complex tasks usually incorporate several processes.

This is a **document**. Documents can relate to either input or output, paper or electronic. In this case, there is a paper output.

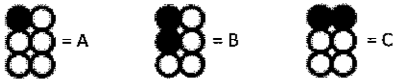
This is a **connector**. This algorithm is incomplete because it is part of a large process that would not fit on one page. The next part of this algorithm would begin with another connector, containing the same letter. This would let you know which connectors match up. The end of an algorithm is always signaled with a terminator, containing the word 'end'.

Users with Specific Needs

Computer systems can be customised to meet the specific needs of users who are visually impaired, unable to use their hands or subject to any other form of disability. This customisation usually involves adding one or more input or output device, as well as the installation of software.

Input devices include:

- **Eye-typer** – This allows people to communicate with a computer using only their eyes. A camera focused on the eye can determine where the user is looking on the screen, and move the cursor to that position. Patterns of blinking can then be translated to different types of mouse click (left, right, double, etc.).
- **Braille keyboard** – Braille is a system of writing used by blind and partially sighted people. Each number or letter is represented by a pattern of dots, raised up from the page (typically by punching the page), that can be read by a person's fingertips. For example:

  The *black* dots represent dots that can be felt by the reader. The *white* dots are just spaces, not yet raised, that are needed to represent other letters. All six are used, in combination, making Braille not dissimilar to binary.

  A Braille keyboard is either a device used to produce these characters by punching a page, or a standard keyboard with Braille symbols overlaid on the keys to make them accessible to blind or partially sighted people.
- **Puff-suck switch** – This allows the user to communicate with a computer by blowing into or sucking on a tube. What the computer does with this input depends on the software, but it is ideal for allowing limited mobility users to answer yes/no questions..

Output devices include:

- **Braille printer** – Using the Braille system described above, a Braille printer produces output that can be read by blind people. This printer does not print, but punches the page at specific points to raise dots.
- **Text-to-speech** – A combination of speech synthesis software and speakers to read documents or web pages to visually impaired people.
- **Screen magnifiers** – These devices, aimed to help partially-sighted users, clip into the front of a monitor to enlarge the visible display.

HTML

Most websites you access using the Internet are written in a language called **Hypertext Markup Language (HTML)**. This language is used to tell a browser how to display each web page, such as which images are to be used, where they will be positioned. HTML is the **industry standard** (nearly all web pages use it), which means that different browsers display web pages in the same way.

Much of HTML is made up of **tags**, which surround text to describe how it will be displayed. Here are some examples, with the tags highlighted:

```
<strong>Hello</strong>
```

With this line of HTML, the word 'Hello' would be displayed in bold.

```
<h1>Hello</h1>
```

With this line of HTML, the word 'Hello' would be displayed in a large font. The 'h' means heading, and '1' is the largest heading available ('h2' being slightly smaller, then 'h3' and so on).

```
<img src="http://www.images.com/12345.jpg">
```

This tag would display an image that would be displayed on a web page.

Different tags are used to dictate font size, text position, bullet points, numbered lists, tables, text boxes, buttons and anything else that can be displayed on a web page. The purpose is to ensure that the page is described, in HTML, so precisely that all web browsers would display a page in exactly the same way.

Like other technologies, the capabilities of HTML increase with each **version**. The present version is **HTML4** (as at October 2012). The next version, **HTML5**, will introduce new tags, such as `<audio>` and `<video>`, to enable web pages to offer more sophisticated content.

When you access a web page, you might view or download files of different **formats**. A file's format shows us what type of file it is (such as an image or a text file) and what program could be used to open it. Here are some common file formats found across the Internet:

JPG – This is a file format for images that allows photographs to be stored using less space than with other file formats.

GIF – This is another image file format that is better suited to graphics rather than photographs. Animated (moving) pictures can also be stored as a GIF.

PDF – This format is well suited to documents to be read. It is very popular online, as the software to access PDFs is usually free. Additionally, PDFs are **platform independent**; this means that any computer, running any browser or PDF reader, will display a document in the same way.

MP3 – This is the most popular file format for storing music online. The quality is not as good as the WAV format (another music format), but the file sizes of MP3s are smaller, thus quicker to download.

MPEG – There are many formats available for creating videos to share online, but this is one of the more popular ones. The file will contain compressed (see below) video and audio.

Different storage devices are selected for different reasons:

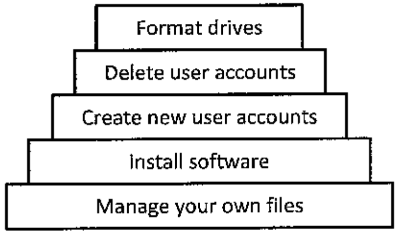
Device	Details	Example Use
USB flash drive (solid state)	**Capacity** – typically range from 1 GB to 64 GB, but higher capacities are always being developed. **Speed** – USB 2.0 technology transmits data at a rate of 5 million bits per second. **Portability** – very small and light; designed to be moved between computers. **Durability** – as solid-state devices, USB flash drives are generally very sturdy.	Commonly used by students who do not always work at the same computer but always need access to their work.
Optical disks	**Capacity** – CDs: 700 MB; DVDs: 4.7 GB; Blu-Ray disks: 25 GB. **Speed** – highly variable, although Blu-Ray drives are much quicker than DVDs or CDs. **Portability** – small and light, but not as easy to carry as a USB flash drive. **Durability** – quite sturdy, although if the surface is scratched, it can make the data irretrievable.	Useful for posting data from one physical location to another, especially as most computers have an optical drive. The content of many optical drives cannot be edited once saved.
Internal hard disk drive (magnetic)	**Capacity** – variable, although new computers can have disks with capacities measured in terabytes. **Speed** – data transfer from an internal drive is usually far quicker than any external device. **Portability** – it is difficult and time-consuming to move an internal drive from one machine to another. **Durability** – very well protected when inside the computer, and protected by a sturdy metal case when removed.	Most programs are stored on a computer's internal hard drive, as it is sturdy, quick and often has a huge capacity. The only real disadvantage is a lack of portability.

Network Security Measures

Passwords are the most obvious means of security. If you don't know the password, you cannot have access to the computer. In order to keep a computer truly secure, **strong passwords** are needed:

- `james` – This is a weak password (especially if the user's name is James or has a child named James). Other people could guess this quite easily, so whatever this password protects is not really secure.
- `fencepost186` – This is a better password. Combining words with other words or words with numbers makes passwords harder to guess.
- `fN%^601t` – This is a strong password. With a combination of upper case, lower case, numbers and symbols, as well as the fact that this password contains no words, it would be nearly impossible to guess. Even if someone were looking over your shoulder as you typed, it would be very difficult for them to remember.

User access levels – Passwords are often tied to **usernames**, so the computer knows who you are when you log in. Different users have different user access levels. Depending on who you are, you will have certain **privileges** within the network, i.e. you will be allowed to do certain things. Consider this:



These are some of the tasks that could be performed on a network, and not everyone can do all of the tasks, for example:

- If this were a school or college network, a *student* could only manage their own files.
- A *trainee network technician* might be able to do the bottom two. They would not be able to create user accounts, because their access level does not allow them to do this.
- A *network manager* would be able to do any of these things. They would also be able to change the access level of any other user.

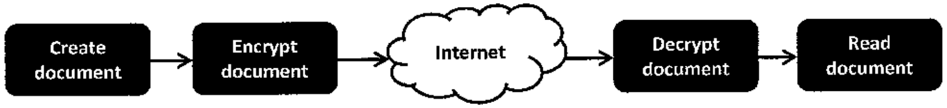
Encryption processes turn readable data into what seems to a human being to be complete nonsense. Let's take the following sentence and encrypt it:

```
I like mopping
```

We will now take each character as it was typed, on a standard keyboard, and move it two keys to the right:

```
P 'p;t .[]]p,j
```

Unless the person who reads this knows how it was derived, it will make no sense to them whatsoever. This data could be sent across the Internet safely. In reality, data that needs to be encrypted would be much more sensitive, and the encryption method would be far more sophisticated, but that's how it works.



Command-Line Interface (CLI)

Command-line interfaces typically require only typed input from the user, who must learn a set of commands before they are able to use the operating system. One such interface is MS-DOS 6.0, which is operated by typing in commands such as those below:

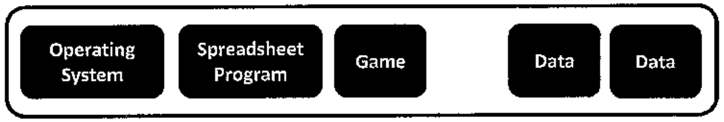
```
DELETE C:/FILE.TXT                  Deletes a file from one location
COPY C:/FILE2.TXT A:/FILE2.TXT      Copies a file from one location to another
```

As computers have become more powerful and more and more novices have become regular users, GUIs are far more in demand than CLIs.

Advantages	Disadvantages
• Can be installed on less advanced machines, as there is less need for an interface requiring advanced graphics or an abundance of memory and backup storage. • Run very quickly, particularly when used by an expert user.	• Users are required to learn an instruction set before they can use the interface.

Memory Management

A computer's main memory can only contain so much. Programs that are open and data that is being used take up space in the main memory. Among the roles of the operating system are loading programs or data into memory and removing them when they are no longer needed.

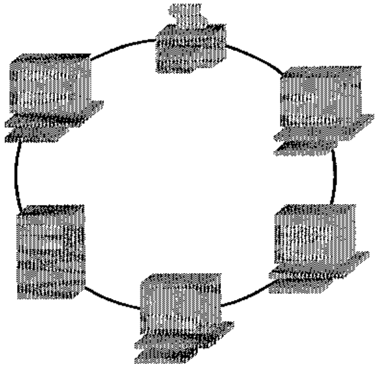


Peripheral Management

A **peripheral** is a hardware device, not part of the CPU or main memory, that is plugged into the computer. Common peripherals include the keyboard, mouse, monitor, printer and speakers, with less obvious ones including barcode scanners, sensors and servomechanisms.

The role of the operating system is to manage these devices and their **drivers** (a driver is a small piece of software that tells a computer how to operate a particular peripheral). The operating system may sometimes need to coordinate peripherals that run at different speeds (for example, a digital camera can transmit pictures to print much faster than a printer could print them). The operating system will also ask the user for a driver if the correct one cannot be found.

Ring Topology



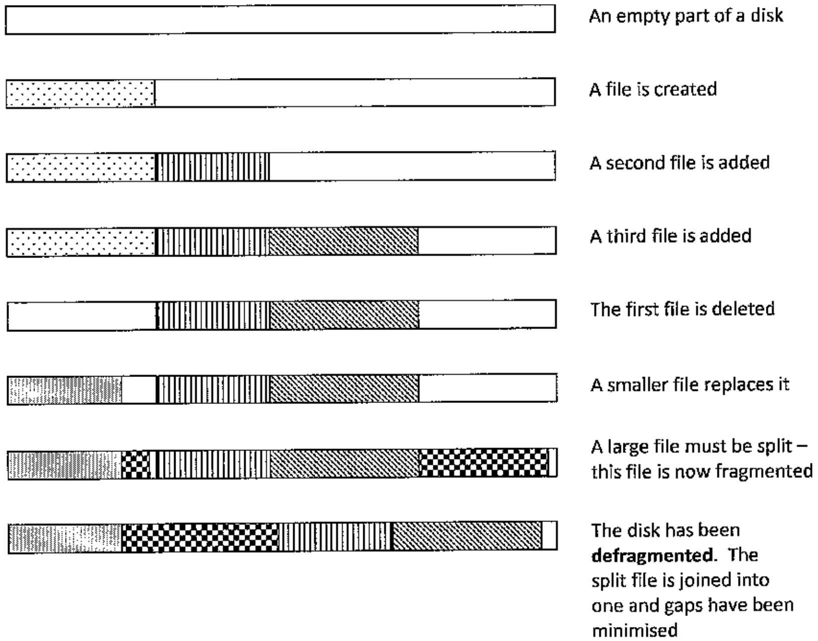
Each device is connected to two other devices, so that the devices and connection media make up the shape of a ring (in practice, it is rarely round – it may pass from desk to desk in an office). To communicate with a particular device, a signal often has to pass through several other devices en route. For an Internet connection, one of these devices must either be a router or connected to a router.

Network Topology	Advantages	Disadvantages
Bus	• Very cheap, as it requires a minimal amount of cabling • Easy to add or remove computers	• Failure in the main cable will cause failure of the network • The network can be significantly slowed by high network traffic
Star	• If one cable fails, only the device attached to it is isolated from the network • Faults are easy to find • It is easy to add extra devices • The system is secure, as data is only sent to devices that need that data	• Lots of cable, which means more expensive to install than other topologies
Ring	• No collisions, as there is typically only one batch of data allowed on the network at any one time • Very high rates of transmission are possible	• Failure of a single cable can bring down the entire network • Failure of a single device can bring down the entire network • Security is not ensured, as data passes through other devices en route to its final destination

Some networks can be a combination of several topologies.

Defragmentation

When a disk has been written to and re-written to several times, files may not be stored as efficiently as they might be, and files can be split and spread across the disk, increasing disk access time. Defragmentation moves file segments, physically, from one disk location to another, in an attempt to minimise these gaps and to ensure that files are stored, as much as possible, in a single location. This reduces disk access time, making disk access speed faster.


An empty part of a disk
A file is created
A second file is added
A third file is added
The first file is deleted
A smaller file replaces it
A large file must be split – this file is now fragmented
The disk has been **defragmented.** The split file is joined into one and gaps have been minimised

File Transfer

The operating system is also responsible for moving files, both physically on the disk and conceptually within a directory structure. An operating system will also be capable of moving a file from one device (such as an internal hard drive) to another (such as a USB flash drive).

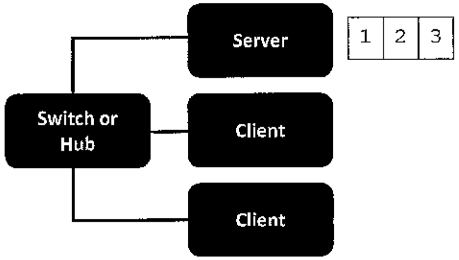
Disk Formatting

Formatting a disk means permanently deleting everything from it and making it ready to have new files saved to it. This process is managed by the operating system, although the operating system itself is deleted if it is on the disk being formatted. There are several reasons for formatting a disk:

- Although it is extreme, it guarantees that viruses and other malware are removed
- If a disk is going to be disposed of, it ensures all personal data is deleted
- Sometimes a disk is formatted as part of the process of installing a new operating system.

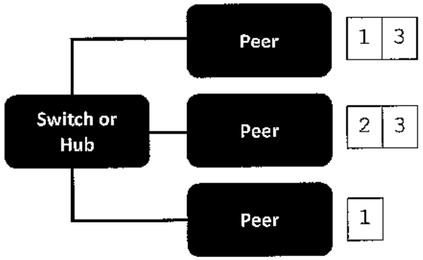
Client-Server or Peer-to-Peer Model

When creating a network, two common models are the **client-server** model and the **peer-to-peer** model. Each one essentially describes how files are stored and shared within a network. This is the client-server model:



A server (which is a computer) holds all the files. In this case, for simplicity, there are only three. If either of the clients (which are also computers; they just play a different role in this model) wants access to any of the files, they would need to request those files from the server.

This is the alternative – the peer-to-peer model:



In this model, all computers have equal standing. All of them can store files, and any peer can transmit a file to any other peer. This model is generally less secure than a client-server model, where security can be concentrated on the machine that holds all the files. There is also an issue of duplication, as you can see, but this can be a good thing. If one peer were to go offline, there is a good possibility that any files that are needed also exist on another peer.

4. Data Representation

As we have seen, computers can only store 1s and 0s. This means that everything that can be saved on a computer, including text, images and sound, is stored using only 1s and 0s. This chapter covers how such storage takes place.

Units of Data

All data needs to be physically stored somewhere and all storage devices (such as CDs, pen drives and hard disks) have a **capacity** – they can only store so much. The following table shows the terms you need to know when trying to understand how much a device can store.

Unit Name	Size	Example of Storage
Bit	A single binary digit	Either a single 1 or a single 0 – nothing else
Nibble	A sequence of four bits	A whole number between 0 and 15
Byte	A sequence of eight bits	An individual keyboard character, such as '#' or 'k'
Kilobyte	Approximately 1,000 bytes	A paragraph of text, containing around 200 words
Megabyte	Approximately 1,000 kilobytes or 1,000,000 (one million) bytes	Around one minute of average quality MP3 music
Gigabyte	Approximately 1,000 megabytes or 1,000,000,000 (one billion) bytes	About 90 minutes of standard definition video
Terabyte	Approximately 1,000 gigabytes or 1,000,000,000,000 (one trillion) bytes	Depending on the quality, several hundred hours of video

Validation

When data is entered into a database, **validation checks** can be conducted on that data to check that it is reasonable and sensible (a validation check does not necessarily make sure that data being entered is correct).

Different types of validation check exist for different types of data. The table below describes some of the more commonly used validation checks:

Validation Check	Description	Example Data
Range check	Numerical data has to be between one number and another, e.g. between 1 and 10.	5 (valid) 11 (invalid)
Presence check	Checking that the user has entered something (anything at all). This is often used for long text fields for which other validation checks do not really apply.	Anything at all (valid) Leaving it blank (invalid)
Type check	Not allowing, for example, letters into a field that should only contain numbers.	5 (valid) five (invalid)
Lookup	Providing the user with a drop-down list from which to select the desired option.	Anything from the list (valid) Anything else (invalid)
Length check	Whatever is entered into a field has to be a certain number of characters long. For example, a National Insurance number (excluding spaces) always contains nine characters.	JD553323C (valid) JD456B (invalid)
Format check	Some pieces of data are formed of characters in a very specific order, such as postcodes.	B12 8GD (valid) B1 AAAA (invalid)

Each field will have its own set of validation checks, and a field can have more than one validation check applied to it.

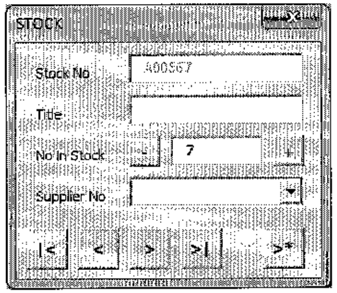
You also need to know how to convert a decimal number to an eight-binary number. The following steps show you how the number **85** is converted, with no need for a calculator:

Instruction	Answer so far	Remaining
We know that there will be eight bits in our answer, so we create a space for eight digits.		**85**
We can then write in the value of each digit immediately above. Start with 1 on the right-hand side, then double each time you add a new number to the left.	**128 64 32 16 8 4 2 1**	**85**
Now, we start with the leftmost bit. 128 is higher than the number we're trying to convert, so we enter a 0.	**128 64 32 16 8 4 2 1** 0	**85**
Next, we look at 64, which is lower than the number we're trying to convert, so we enter a 1 and subtract 64 from our number.	**128 64 32 16 8 4 2 1** 0 1	~~**85**~~ **21**
The next number is 32, which is bigger than the number we're trying to convert (21 at this point, as we've subtracted 64 in our last step). We enter 0 and leave our number unchanged.	**128 64 32 16 8 4 2 1** 0 1 0	**21**
Our number (21) is larger than the next digit (16), so we enter a 1 and subtract 16.	**128 64 32 16 8 4 2 1** 0 1 0 1	~~**21**~~ **5**
With only 5 left to convert, which will clearly be made up of a 1 and a 4, we place 1s into each of these columns and 0s into the others.	**128 64 32 16 8 4 2 1** 0 1 0 1 0 1 0 1	~~**5**~~ **0**

The binary equivalent of 85 is 01010101.

Form

A form is a user-friendly way to enter data into a table or view data already in a table. The content and layout of a form can be customised, using text boxes, drop-down boxes, list boxes, etc. as needed. A form to enter data into the STOCK table might look like this:



Stock No	A00567
Title	
No in Stock	7
Supplier No	

Query

A query is a way of **interrogating** a database to find out specific information, without having to read through whole database tables yourself. If we start with the STOCK table again, we will see some examples of queries:

Stock No	Title	No in Stock	Supplier No
A00123	Halo 4	17	X01
A00124	Sleeping Dogs	12	X02
A00125	Fifa 13	0	X03
A00126	Dragon Age II	7	X03

If we run the query `'No in Stock > 0'` (greater than zero), we will see the following result:

Stock No	Title	No in Stock	Supplier No
A00123	Halo 4	17	X01
A00124	Sleeping Dogs	12	X02
A00126	Dragon Age II	7	X03

We can only see the records that match our query (because zero is not greater than zero). If we run a query with two parts (starting with the original table), we can narrow the data down even more. This is the result of the query `'No in stock > 5 AND Supplier No = "X03"'`:

Stock No	Title	No in Stock	Supplier No
A00126	Dragon Age II	7	X03

Because our query used the AND operator, we will only see a record if it has both a 'No in Stock' of greater than five AND the 'Supplier No' is X03. Only one record matches both of these criteria, so only this record is displayed.

If we use OR, instead of AND, there is usually a larger number of records that appear as the result of a query. Consider the query 'Supplier No = "X01" OR Supplier No = "X02"':

Stock No	Title	No in Stock	Supplier No
A00123	Halo 4	17	X01
A00124	Sleeping Dogs	12	X02

In this case, a record needs to have a 'Supplier No' of either 'X01' OR 'X02' to be displayed in the results of the query.

3. The next pair is just as straightforward: 0 + 0 = 0

```
  10110101
+ 00111100
        01
```

4. As for the next pair, 1 + 1 = 2, which is 10 in binary, just as we did in adding decimal numbers, we must carry the 1 and place the 0 in the answer.

```
  10110101
+ 00111100
       001
      1
```

5. In the next column, we're adding 0 + 1, but we also need to add the carried 1. In binary, 0 + 1 + 1 = 10. This means another 0 in our answer and another carried 1.

```
  10110101
+ 00111100
      0001
     11
```

6. This time, we're adding three 1s – two from the calculation and another that was carried. In binary, 1 + 1 + 1 = 11.

```
  10110101
+ 00111100
     10001
    111
```

7. The next calculation is identical to the previous one. We're carrying out 1 + 1 + 1 = 11.

```
  10110101
+ 00111100
    110001
   1111
```

8. Next, 0 + 0 = 0, but adding the carried 1 gives us a total of 1.

```
  10110101
+ 00111100
   1110001
   1111
```

9. The final pair of numbers, 1 + 0, is straightforward.

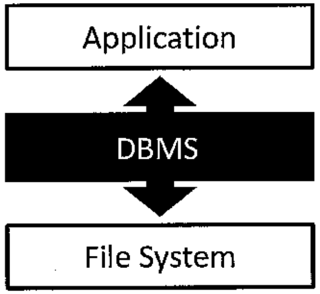
```
  10110101
+ 00111100
  11110001
   1111
```

In binary addition, 10110101 + 01111100 = 11110001.

Database Management Systems (DBMS)

A database is a collection of related data items stored systematically in a file system. A DBMS is a piece of software that sits between the application (the program the user works with) and this file system (the data itself). The purpose of a DBMS is to allow databases to be created, used and maintained. The main features of a DBMS are:

- An ability to check data for inconsistencies
- An ability to minimise duplicated data (a major role of a database is to ensure each piece of data is stored only once, thereby reducing potential errors)
- An ability to allow multiple users access to the data at the same time
- An ability to obtain related data from different tables in the file system by linking these tables to one another
- The provision of a customisable user interface to make the data understandable
- The capability to implement security to keep data secure



Keeping the data and the program separate is a good thing to do, because the program could be updated, or even replaced with a completely new program, and the data wouldn't need to change. Similarly, if the data needs to change at any time, the program does not need to be altered in any way.

Overflow

Whenever a computer adds together two binary numbers, it reserves a space in memory for the result. The size of this space is usually determined in advance and it is usually a number of bits divisible by eight (8 bits, 16 bits, 32 bits, 64 bits or 128 bits). Overflow takes place when the result is too big for the number of bits reserved to hold the result.

Let's look at an example:

11100111 in binary is 231 in decimal
10100011 in binary is 163 in decimal.

Both of these numbers are small enough to fit within eight bits, but once they are added together...

```
   11100111
 + 10100011
 ----------
  110001010
  111  111
```

...we are left with a nine-digit result. If 16 digits had been reserved for this result, then we don't have a problem, but if only eight digits have been reserved, this result will not fit, and one of the following things will happen:

- The user will see an error message saying that 'overflow' has occurred, and whatever program was running may crash or close down.
- The computer will ignore the ninth bit (the 1 at the leftmost bit of our result) and instead go with the result as 10001010 (the last eight bits of the result). This would essentially mean the result of 231 + 163 being given as 138, which is obviously incorrect.

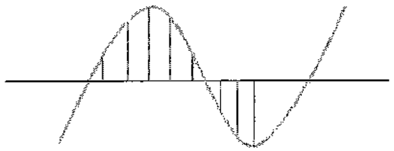
No matter how many bits are reserved for the result of a calculation, there is always the possibility that the result may be too big to fit into these bits, and 'overflow' will occur.

Sound

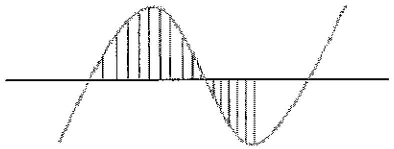
When converting sound to a digital format (which means converting it to binary numbers), two things influence how much storage space is needed:

Sampling Rate

The term **sampling rate** or **sampling interval** refers to how often the computer takes a measure of the sound, including the volume and frequency of the sound. It will do this multiple times per second.
This is a lower sampling rate. The vertical lines each represent a sample:



This is a higher sampling rate:



A higher sampling rate results in better sound quality, but requires more storage space.

Bit Rate

The **bit rate** is how many bits are used for each sample (measured in **bits per second** – bps), and this can vary widely. Again, higher bit rates result in higher quality of sound but also higher storage requirements.

Instructions

Every computer program is made up of instructions and data. The instructions tell the computer what to do, and the data might be text, number, image or sound that the instructions apply to. For example:

- ADD 60

If this were in a program, a number might need to be added to a running total. 'ADD' is the instruction – this is what the computer must do. '60' is the data – this is the number that must, in this case, be added to the running total.

- DELETE picture.bmp

Here, 'DELETE' is the instruction, and the data, 'picture.bmp' is the name of the file that must be deleted.

Because both instructions and data will consist of a sequence of 0s and 1s, the computer needs a way to distinguish between them. Each instruction and each piece of data will be stored in a specific location on the disk with a specific **memory address**. The operating system keeps track of which memory addresses contain instructions and which contain data.

Hexadecimal → Binary

1. This is the hexadecimal number we will convert to binary. `A6`
2. Each hexadecimal digit will translate to a binary nibble, according to the table above. Translate each digit separately. `A = 1010` `6 = 0110`
3. Attach the nibbles together. If you choose to, you may leave a space between them for readability, but you do not have to. `1010 0110`

So, A6 in hexadecimal is equivalent to 10100110.
If you need to convert between decimal and hexadecimal numbers, the best way is to go *through* a binary number, so either **decimal → binary → hexadecimal** or **hexadecimal → binary → decimal**, depending on the conversion you are asked to make.

Characters

Binary numbers are also used to represent **characters** (including letters, numbers, punctuation and symbols). A computer will have at least one **character set** installed, which will store the binary values of all characters that computer is capable of using. Two common character sets are ASCII (American Standard Code for Information Interchange) and Unicode:

ASCII

ASCII Code	Character Value
1000001	A
1100001	a
0100011	#
0110011	3

Unicode

Unicode	Character Value
0000 0000 0100 0001	A
0000 0000 0110 0001	a
0000 0000 0010 0011	#
0000 0000 0011 0011	3

ASCII characters, requiring only seven bits, needs less storage space but can only store 128 different characters. Unicode, requiring 16 bits, need more storage space but can store 65,536 different characters – enough for foreign alphabets and thousands of symbols.