

Revision Guide

for OCR GCSE Computer Science (J277)

Covering Component 1 and Component 2

R Lee

zigzageducation.co.uk

POD
11717

Publish your own work... Write to a brief...
Register at publishmenow.co.uk

Follow us on Twitter [@ZigZagComputing](https://twitter.com/ZigZagComputing)

Contents

Product Support from ZigZag Education	ii
Terms and Conditions of Use	iii
Teacher's Introduction.....	1
Revision Checklist.....	2
Component 1: Computer Systems.....	2
Component 2: Computational Thinking, Algorithms and Programming	4
Component 1: Computer Systems	5
1.1. Systems Architecture	5
1.1.1. Architecture of the CPU	5
1.1.2. CPU Performance	6
1.1.3. Embedded Systems	7
1.2. Memory and Storage	10
1.2.1. Primary Storage (Memory)	10
1.2.2. Secondary Storage	11
1.2.3. Units.....	12
1.2.4. Data Storage	13
1.2.5. Compression	18
1.3. Computer Networks, Connections and Protocols	25
1.3.1. Networks and Topologies	25
1.3.2. Wired and Wireless Networks, Protocols and Layers	30
1.4. Network Security	37
1.4.1. Threats to Computer Systems and Networks	37
1.4.2. Identifying and Preventing Vulnerabilities	38
1.5. Systems Software	41
1.5.1. Operating Systems.....	41
1.5.2. Utility Software.....	42
1.6. Ethical, Legal, Cultural and Environmental Impacts of Digital Technology	45
1.6.1. Ethical, Legal Cultural and Environmental Impact.....	45
Component 2: Computational Thinking, Algorithms and Programming.....	51
2.1. Algorithms.....	51
2.1.1. Computational Thinking	51
2.1.2. Designing, Creating and Refining Algorithms	53
2.1.3. Searching and Sorting Algorithms	57
2.2. Programming Fundamentals	62
2.2.1. Programming Fundamentals	62
2.2.2. Data Types	67
2.2.3. Additional Programming Techniques	68
2.3. Producing Robust Programs	78
2.3.1. Defensive Design	78
2.3.2. Testing	81
2.4. Boolean Logic.....	85
2.4.1. Boolean Logic.....	85
2.5. Programming Languages and Integrated Development Environments	90
2.5.1. Languages	90
2.5.2. The Integrated Development Environment (IDE)	91
Sample Answers	94
Glossary	99
Component 1	A5 booklet
Component 2.....	A5 booklet

Teacher's Introduction

This guide has been produced specifically to support learning of the OCR GCSE (9–1) Computer Science specification, with first examinations in summer 2022.

As the specification is split into 11 sections, this guide has been split into 11 chapters, with content precisely mirroring the order of topics in the specification. As such, it is quite straightforward for learners to keep track of where they are in the material, and what remains to be done. The checklists, which are intended as a working document, are also useful in this regard.

Each chapter contains, along with the theory material and illustrations, a series of exam-style questions with model answers and commentaries. Throughout the guide, a full range of question types is covered, from single-word answers and definitions to long-answer descriptions and discussions.

One chapter can be distributed to students each week and can be used to supplement taught material by aiding such homework/classwork tasks as providing written summaries of a chapter and/or completing the end-of-chapter questions.

More imaginative supplementary tasks that can use this guide as a starting point include the following (you may want to build some or all of these into a weekly routine, each week focusing on a different chapter):

- Providing students with lines from the appropriate section of the specification and asking them to treat each line as if it were a question. The structure of this guide can aid them in locating the answer.
- Asking students to produce five multiple-choice questions based on each chapter. Each question they produce needs to contain a correct answer, three realistic wrong answers and an indication of which answer they believe is correct. The better sets of questions can be archived to produce a half-term multiple-choice quiz, generated by students.
- Asking students to produce a mind map of each chapter as a means of aiding revision. Where applicable, Venn diagrams, flow charts and other graphic organisers can be used in this way.
- Dividing students into groups to deliver presentations on different areas of a chapter. If the group is fairly mature, these presentations can be peer-assessed.
- Using this guide as the basis for flipped learning.

I hope this guide proves useful to both teachers and students.

R Lee, July 2022

Remember!

Always check the exam board website for new information, including changes to the specification and sample assessment material.

Revision Checklist

Component 1: Computer Systems

1.1. Systems Architecture	<input type="checkbox"/> Describe the fetch–execute cycle <input type="checkbox"/> Describe the role and function of the following components: <div style="display: flex; justify-content: space-between;"> <div> <input type="checkbox"/> Arithmetic Logic Unit (ALU) <input type="checkbox"/> Control Unit (CU) <input type="checkbox"/> Cache <input type="checkbox"/> Memory Address Register (MAR) </div> <div> <input type="checkbox"/> Memory <input type="checkbox"/> Program Counter <input type="checkbox"/> Accumulator </div> </div> <input type="checkbox"/> Describe the nature of the von Neumann architecture <input type="checkbox"/> Describe how CPU characteristics affect CPU performance <input type="checkbox"/> Describe, with examples, the characteristics and purposes of emulators
1.2. Memory and Storage	<input type="checkbox"/> Describe the purposes of RAM, ROM and virtual memory <input type="checkbox"/> Describe the need for, and characteristics of, secondary storage <input type="checkbox"/> Select storage devices for a given situation, based on capacity, speed, reliability and cost <input type="checkbox"/> Explain why data must be stored in binary format <input type="checkbox"/> Calculate storage requirements of sound, image and text files <input type="checkbox"/> Convert between each of the following: <div style="display: flex; justify-content: space-between;"> <div> <input type="checkbox"/> Bit <input type="checkbox"/> Nibble <input type="checkbox"/> Byte <input type="checkbox"/> Kilobyte </div> <div> <input type="checkbox"/> Megabyte <input type="checkbox"/> Gigabyte <input type="checkbox"/> Terabyte <input type="checkbox"/> Petabyte </div> </div> <input type="checkbox"/> Convert between binary, denary and hexadecimal integers <input type="checkbox"/> Perform addition on two binary integers <input type="checkbox"/> Carry out left and right binary shifts <input type="checkbox"/> Define the terms 'character set', 'ASCII' and 'Unicode' <input type="checkbox"/> Describe the relationship between the number of bits to represent characters that can be represented <input type="checkbox"/> Describe the way in which images are stored as a sequence of bits <input type="checkbox"/> Describe the effect of colour depth and resolution on the quality of images <input type="checkbox"/> Explain the role of image metadata, with examples <input type="checkbox"/> Describe how sound is sampled and stored <input type="checkbox"/> Describe the effect of sample rate, duration and bit depth on audio quality and size <input type="checkbox"/> Explain the need for compression <input type="checkbox"/> Describe, with examples, lossy and lossless compression

INSPECTION COPY

**COPYRIGHT
PROTECTED**



<p>1.3. Computer Networks, Connections and Protocols</p>	<ul style="list-style-type: none"> <input type="checkbox"/> Compare LANs and WANs <input type="checkbox"/> Describe a range of factors that can affect network performance <input type="checkbox"/> Compare peer-to-peer and client-server systems <input type="checkbox"/> Describe the roles of the hardware required to connect to a network <ul style="list-style-type: none"> <input type="checkbox"/> Wireless access point <input type="checkbox"/> Router <input type="checkbox"/> Switch <input type="checkbox"/> Network interface card <input type="checkbox"/> Transmission medium <input type="checkbox"/> Describe the nature of the Internet, including DNS, hosting, the client-server model <input type="checkbox"/> Compare star and mesh network topologies <input type="checkbox"/> Compare Ethernet, Wi-Fi and Bluetooth as means of connecting devices <input type="checkbox"/> Outline the nature and purpose of encryption <input type="checkbox"/> Compare IP addressing and MAC addressing <input type="checkbox"/> Describe the role of static and dynamic IP <input type="checkbox"/> Describe the purpose and key features of the following protocols: <ul style="list-style-type: none"> <input type="checkbox"/> FTP (File Transfer Protocol) <input type="checkbox"/> HTTP (Hypertext Transfer Protocol) <input type="checkbox"/> HTTPS (Hypertext Transfer Protocol Secure) <input type="checkbox"/> IMAP (Internet Message Access Protocol) <input type="checkbox"/> IP (Internet Protocol) <input type="checkbox"/> POP (Post Office Protocol) <input type="checkbox"/> SMTP (Simple Mail Transfer Protocol) <input type="checkbox"/> TCP (Transfer Control Protocol) <input type="checkbox"/> Describe the four-layer TCP/IP model
<p>1.4. Network Security</p>	<ul style="list-style-type: none"> <input type="checkbox"/> Describe the nature of a range of threats to computer systems: <ul style="list-style-type: none"> <input type="checkbox"/> Brute-force attacks <input type="checkbox"/> Data interception and theft <input type="checkbox"/> Denial of service attacks <input type="checkbox"/> Malware <input type="checkbox"/> Social engineering <input type="checkbox"/> SQL injection <input type="checkbox"/> Describe the nature of a range of countermeasures: <ul style="list-style-type: none"> <input type="checkbox"/> Anti-malware software <input type="checkbox"/> Encryption <input type="checkbox"/> Firewalls <input type="checkbox"/> Passwords <input type="checkbox"/> Penetration testing <input type="checkbox"/> Physical security <input type="checkbox"/> User access control
<p>1.5. Systems Software</p>	<ul style="list-style-type: none"> <input type="checkbox"/> Describe the following roles of an operating system: <ul style="list-style-type: none"> <input type="checkbox"/> File management <input type="checkbox"/> Memory management <input type="checkbox"/> Peripheral management <input type="checkbox"/> User interface <input type="checkbox"/> User management <input type="checkbox"/> Describe the overall nature of utility software <input type="checkbox"/> Explain the need for the following utilities: <ul style="list-style-type: none"> <input type="checkbox"/> Data compression <input type="checkbox"/> Defragmentation <input type="checkbox"/> Encryption
<p>1.6. Ethical, Legal, Cultural and Environmental Impacts of Digital Technology</p>	<ul style="list-style-type: none"> <input type="checkbox"/> Discuss cultural impacts of digital technology upon society <input type="checkbox"/> Discuss environmental impacts of digital technology upon society <input type="checkbox"/> Discuss ethical impacts of digital technology upon society <input type="checkbox"/> Discuss the legal implications of digital technology <input type="checkbox"/> Discuss privacy issues relating to digital technology <input type="checkbox"/> Describe each of the following pieces of legislation: <ul style="list-style-type: none"> <input type="checkbox"/> Computer Misuse Act (1990) <input type="checkbox"/> Copyright, Designs and Patents Act (1988) <input type="checkbox"/> Data Protection Act (2018) <input type="checkbox"/> Compare open source and proprietary software licences

**COPYRIGHT
PROTECTED**



Component 2: Computational Thinking, Algorithms and

2.1. Algorithms	<input type="checkbox"/> Describe the concept of abstraction <input type="checkbox"/> Describe and apply the principle of decomposition <input type="checkbox"/> Apply algorithmic thinking to a problem <input type="checkbox"/> Identify inputs, processes and outputs for a given problem <input type="checkbox"/> Create and interpret a structure diagram <input type="checkbox"/> Define an algorithm using pseudocode <input type="checkbox"/> Define an algorithm using a flow chart <input type="checkbox"/> Define an algorithm using a high-level language <input type="checkbox"/> Identify common errors in an algorithm <input type="checkbox"/> Determine an algorithm's purpose using a trace table <input type="checkbox"/> Outline the nature of a binary search <input type="checkbox"/> Outline the nature of a linear search <input type="checkbox"/> Outline the nature of a bubble sort <input type="checkbox"/> Outline the nature of a merge sort <input type="checkbox"/> Outline the nature of an insertion sort				
2.2. Programming Fundamentals	<input type="checkbox"/> Describe variables, constants, operators, inputs, outputs and assignment <input type="checkbox"/> Describe the role of sequence in a program <input type="checkbox"/> Describe the role of selection in a program <input type="checkbox"/> Describe the role of count-controlled iteration in a program <input type="checkbox"/> Describe the role of condition-controlled iteration in a program <input type="checkbox"/> Describe and use arithmetic operators <input type="checkbox"/> Describe and use comparison operators <input type="checkbox"/> Describe and use Boolean operators <input type="checkbox"/> Describe and use a range of data types (integer, real, Boolean, character) <input type="checkbox"/> Use casting to convert data to a specific type <input type="checkbox"/> Describe and conduct basic string manipulation <input type="checkbox"/> Write and understand code to interact with text files (opening, writing, reading) <input type="checkbox"/> Describe the nature of records in terms of data storage <input type="checkbox"/> Write SQL queries to interact with relational databases <input type="checkbox"/> Define and use arrays (one- and two-dimensional) <input type="checkbox"/> Describe and use subprograms, including functions and procedures <input type="checkbox"/> Write and understand code to generate random numbers				
2.3. Producing Robust Programs	<input type="checkbox"/> Describe the nature of defensive design in software development <input type="checkbox"/> Describe a range of data validation techniques <input type="checkbox"/> Explain how and why code should be made maintainable <input type="checkbox"/> Outline the purpose of testing <input type="checkbox"/> Distinguish between iterative and final/terminal testing <input type="checkbox"/> Identify syntax and logic errors in code <input type="checkbox"/> Select normal, boundary and error test data in a given situation <input type="checkbox"/> Explain how to refine algorithms in the light of testing				
2.4. Boolean Logic	<input type="checkbox"/> Create logic statements using AND, OR and NOT <input type="checkbox"/> Create and interpret truth tables using AND, OR and NOT <input type="checkbox"/> Use logic to solve problems				
2.5. Programming Languages and Integrated Development Environments	<input type="checkbox"/> Distinguish between high-level and low-level programming languages <input type="checkbox"/> Distinguish between compilers and interpreters <input type="checkbox"/> Outline the roles of a range of tools within an integrated development environment <table border="0" style="width: 100%;"> <tr> <td><input type="checkbox"/> Editors</td> <td><input type="checkbox"/> Runtime systems</td> </tr> <tr> <td><input type="checkbox"/> Error diagnostics</td> <td><input type="checkbox"/> Translators</td> </tr> </table>	<input type="checkbox"/> Editors	<input type="checkbox"/> Runtime systems	<input type="checkbox"/> Error diagnostics	<input type="checkbox"/> Translators
<input type="checkbox"/> Editors	<input type="checkbox"/> Runtime systems				
<input type="checkbox"/> Error diagnostics	<input type="checkbox"/> Translators				

INSPECTION COPY

**COPYRIGHT
PROTECTED**



1.1. Systems Architecture

1.1.1. Architecture of the CPU



CPU – the Central Processing Unit executes program instructions, performs comparisons, as well as coordinating the behaviour of other hardware components.

Component	Function
Arithmetic logic unit	Performs various operations: <ul style="list-style-type: none"> • Arithmetic operations (+, -, *, /) • Comparison operations (< > =) • Logical operations (AND, NOT, OR)
Control Unit	Manages the execution of instructions by coordinating the activities of the other components.
Registers	<p>MAR (Memory Address Register) – stores the memory location that either be read from this location or written to it.</p> <p>MDR (Memory Data Register) – stores the data itself that has been read from memory or is about to be written to it.</p> <p>PC (Program Counter) – contains the memory location of the next instruction. Between instructions, the contents of this register are incremented.</p> <p>Accumulator – stores the intermediate results of calculations. For example, if a calculation '5 + 2 - 4' would be partially complete at '5 + 2'. At the next step, the result, would be stored in the accumulator.</p>
Cache	Cache memory stores copies of data or instructions from RAM that are used regularly. This means that these data or instructions can be accessed more quickly.

The activities of the processor are governed by the fetch–execute cycle.



Fetch–execute cycle – instructions are fetched from memory, into the CPU (carried out). It's a cycle because it repeats. During the fetch–execute cycle, the CPU carries out its role as described in the table above.

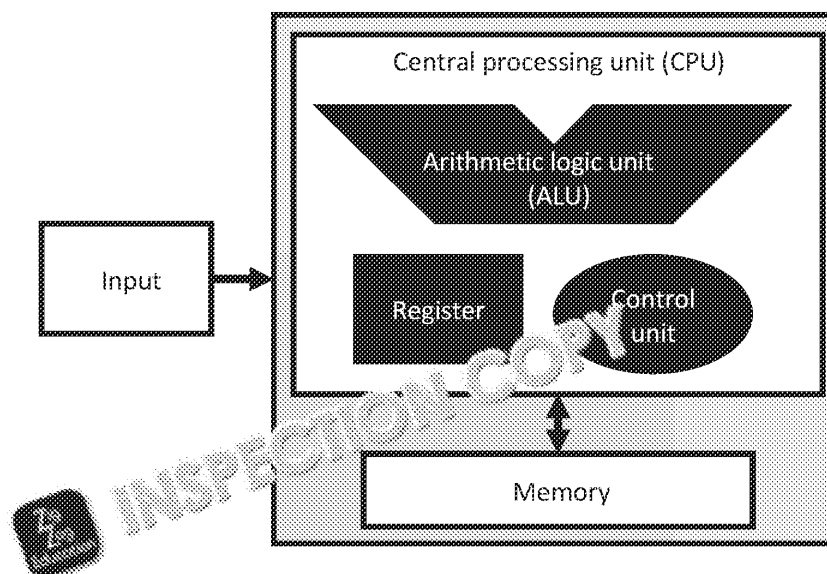
Many times per second, around three billion on a modern PC, a component called the clock generates a regular pulse. The frequency of the clock pulse is measured in hertz (Hz). For example, a clock pulse that is generated one billion times per second is generated with a frequency of 1 gigahertz (GHz). Any fetching or executing, begins on one of these clock pulses.

INSPECTION COPY

COPYRIGHT
PROTECTED



The CPU is connected to other components, and the **von Neumann architecture** shows how the CPU and other parts are connected:



Data is input, processed by the CPU, which has several components of its own, and then output, allowing data to be stored. The arrows in this diagram indicate **buses**, which are used to convey data.

1.1.2. CPU Performance



CPU performance – a CPU with a higher rate of performance can execute more instructions than a CPU with a lower rate of performance. Several factors can affect CPU performance:

- Clock speed
- Cache size
- Number of cores



Clock speed – the number of clock pulses per second, typically measured in Hertz (Hz). A processor has a clock that pulses three billion times per second, meaning there are three billion opportunities, each second, for an instruction to begin. Note that instructions typically require more than a single clock pulse to complete.

Cache memory is used to store data and instructions that will be accessed repeatedly. If cache memory is available, a larger amount of data and instructions can be stored there. Since cache is a different form of memory, this would improve CPU performance.



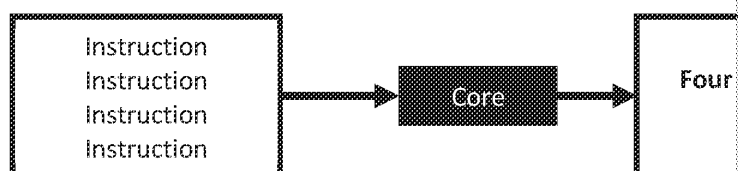
Core – a single unit, consisting of an ALU and a Control Unit, which can execute instructions. Multiple cores can execute instructions at the same time, so more cores mean that more instructions can be executed simultaneously.

INSPECTION COPY

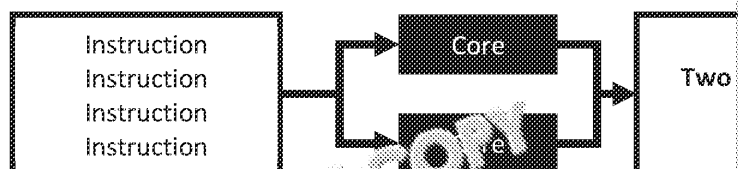
**COPYRIGHT
PROTECTED**



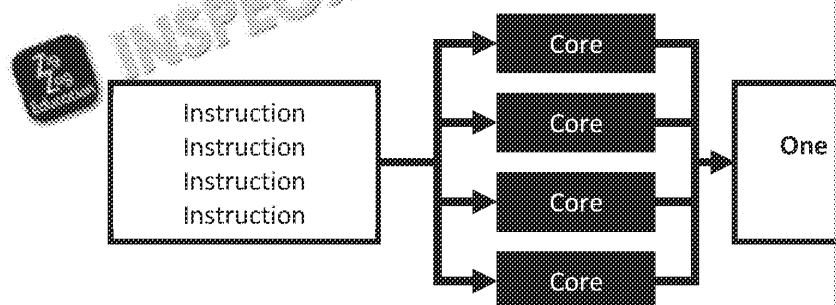
Four instructions with one core:



Four instructions with two cores (a **dual core** processor):



Four instructions with four cores (a **quad core** processor):



A dual core processor is not quite twice as fast as a single core processor because of organising which core will follow which instructions (this time is called the **overhead**). This overhead is usually small enough that a dual core processor can be considered twice as fast as a single core processor at the same clock speed. It should be noted, however, that not all applications are designed for multiprocessing.

1.1.3. Embedded Systems



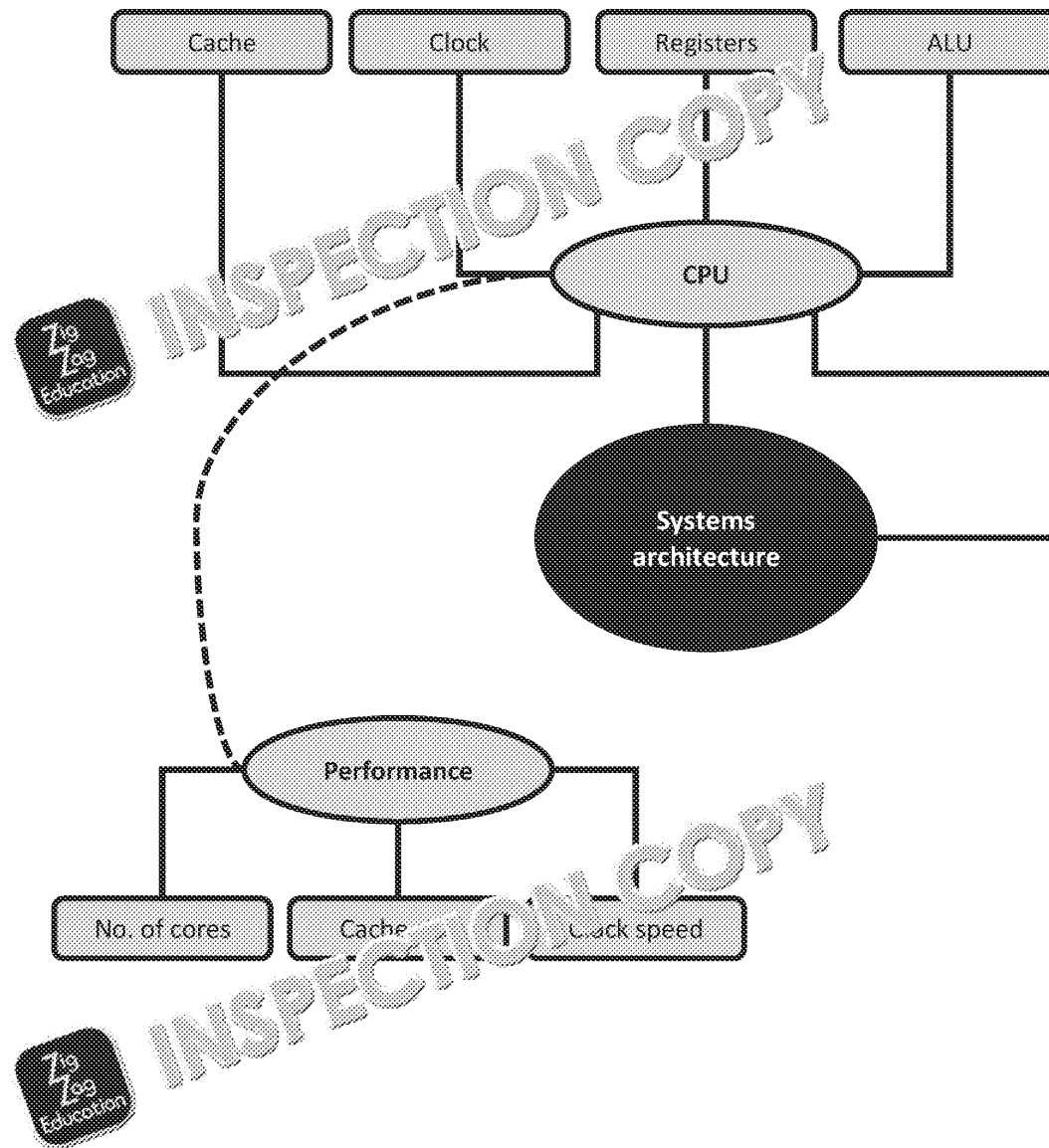
Embedded system – a computer that exists within a larger mechanical system, such as a microwave oven or a guided missile. Embedded systems are used when a system is required to be reliable (as in a microwave) or when they have only a single, specific purpose.

Examples of embedded systems are kitchen appliances (microwaves, washing machines), consumer electronics (watches, fitness trackers) and subsystems of vehicles (entertainment systems, climate control systems).

COPYRIGHT
PROTECTED



Systems Architecture Mind Map



INSPECTION COPY

COPYRIGHT
PROTECTED



Sample Examination-style Questions

1. Describe, with an example, what is meant by the term **embedded system**.


.....

.....

.....

.....

2. Describe the role of each of the following components in the fetch–execute cycle.

Component	Description
 Program counter	
Accumulator	
Control unit	

INSPECTION COPY

COPYRIGHT
PROTECTED




1.2. Memory and Storage

1.2.1. Primary Storage (Memory)

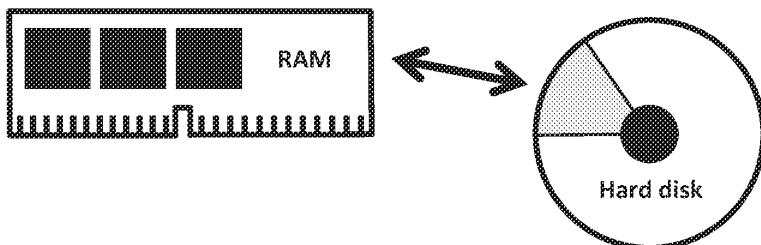


Primary storage – one of several terms (including main memory and memory) that describes the component that stores programs that are currently in use. It can be distinguished from secondary storage, which also stores programs and data not currently in use.

Term	Definition
RAM 	Random Access Memory. When a program is loaded from its data and instructions are copied into RAM, which is generally faster than a hard disk. When a computer is turned off or switched on, the content of RAM is lost. This means that RAM is volatile . If a computer has more RAM, it will be able to run more applications.
ROM	Read Only Memory. 'Read only' means that the content cannot be updated. (meaning it is not volatile). As such, ROM stores data or instructions that are not updated. ROM will typically store bootstrapping instructions, the initial steps in finding and initialising the operating system when the computer is turned on.



Virtual memory – part of the computer's secondary storage is treated as if it were primary storage. Data and instructions currently in use. Data is transferred between the user being made aware of each individual transfer. Virtual memory, in a computer system, is used when the computer needs to free up space in primary storage.



INSPECTION COPY

INSPECTION COPY

COPYRIGHT
PROTECTED



1.2.2. Secondary Storage



Secondary storage – long-term storage in a computer system, necessary for data that is not volatile and will also run out of storage space. Three categories of secondary storage are **magnetic** and **solid state**. All of these devices are required to store vast amounts of data, but they do so in different ways.

Device	How it works
Optical	<p>Ones and zeros are written and read using lasers. At many locations on a disk, a particular location might be smooth, or there might be a bump. Different optical media have different capacities, and some are rewritable.</p> <p>CD: Approximately 700 MB, although there is some variation.</p> <p>DVD: 4.7 GB to 17 GB</p> <p>Blu-ray: 25 GB to 50 GB</p>
Magnetic	<p>The surface of a magnetic disk comprises billions or trillions of tiny regions which are each either magnetised or not. A read-write head can also change them, but magnetic disks need to spin to the correct location to be read, which takes time.</p>
Solid state	<p>Solid state storage devices (of which flash drives are one type) have no moving parts, to store data electronically. With no moving parts, they are sturdier and quieter than magnetic or optical devices. With no moving parts, they are stored using two transistors. One either holds a charge or it doesn't. The second transistor can read the state of the first.</p>

There is no such thing as the *best* storage medium; if there was, no other media would exist. There are several ways in which one medium might be a better fit than another in any given situation.

Criterion	Meaning	Example
Capacity	How much data can be stored on this medium?	An internal hard disk has a much higher capacity than a USB flash drive.
Speed	How quickly can data be retrieved?	Solid state drives are faster than magnetic drives since there are no parts that need to move.
Portability	How easily can data be moved from one location to another?	A USB flash drive is easily portable, while an internal hard drive is not.
Durability	How sturdy is it? Can it stand up to the rigour of daily use?	A solid state hard drive is more durable than a magnetic hard drive, as it contains no moving parts.
Reliability	Can data always be retrieved from the device?	Optical disks, such as CDs, are more reliable than solid state media, as they are not subject to accidental overwrites.
Cost	Not just how much it costs, but how much it costs per gigabyte.	Magnetic disks, in particular, are cheaper than solid state drives.

**COPYRIGHT
PROTECTED**



1.2.3. Units

There are many units for measuring the capacity of a storage medium, including

Unit name	Size	
Bit	A single <u>b</u> inary digit	Either a 1 or 0
Nibble	A sequence of four bits	A whole number
Byte	A sequence of eight bits	An individual character as '#' or 'a'
Kilobyte	1,000 bytes	A paragraph of 200 words
Megabyte	1,000 kilobytes or 1,000,000 (one million) bytes	Around 3 minutes of MP3 music
Gigabyte	1,000 megabytes or 1,000,000,000 (one billion) bytes	About 90 minutes of video.
Terabyte	1,000 gigabytes or 1,000,000,000,000 (one trillion) bytes	Depending on use, up to a hundred years of data
Petabyte	1,000 terabytes or 1,000,000,000,000,000 (one quadrillion) bytes	A one-petabyte hard drive

You may need to convert between these units of measurement:

Conversion required	Calculation	Conversion required
Bit → Nibble	Divide by 4	Nibble → Bit
Nibble → Byte	Divide by 2	Byte → Nibble
Byte → Kilobyte	Divide by 1,000	Kilobyte → Byte
Kilobyte → Megabyte	Divide by 1,000	Megabyte → Kilobyte
Megabyte → Gigabyte	Divide by 1,000	Gigabyte → Megabyte
Gigabyte → Terabyte	Divide by 1,000	Terabyte → Gigabyte
Terabyte → Petabyte	Divide by 1,000	Petabyte → Terabyte

So, if you're asked how many bits to store an 8 kilobyte file:

- $8 \times 1,000 = 8,000$ bytes
- $8,000 \times 2 = 16,000$ nibbles
- $16,000 \times 4 = 64,000$ bits

**COPYRIGHT
PROTECTED**



1.2.4. Data Storage

Different numbering systems exist for whole numbers, fractions, positive numbers. At GCSE level, you need only to work with binary integers that have a denary equivalent.



Denary – the numbering system you're already familiar with, which uses digits from '0' to '9'.

Binary → Denary

This conversion will use the example binary number 11001010. The first step is to identify the value of each binary digit. The placeholder above the rightmost bit is '1', and the value of each bit doubles to the left:

128	64	32	16	8	4	2	1
1	1	0	0	1	0	1	0

Add together the denary values that contain a '1':

$$128 + 64 + 8 + 2 = 202$$

Denary → Binary

The following steps show you how the number 85 is converted, with no need for a calculator.

Instruction	Answer so far																
We know that there will be eight bits in our answer, so we create a space for eight digits.	<table><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>																
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.	<table><tr><td>128</td><td>64</td><td>32</td><td>16</td><td>8</td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>	128	64	32	16	8											
128	64	32	16	8													
Now, we start with the left-most bit. 128 is higher than the number we're trying to convert, so we enter a '0'.	<table><tr><td>128</td><td>64</td><td>32</td><td>16</td><td>8</td><td></td><td></td><td></td></tr><tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>	128	64	32	16	8				0							
128	64	32	16	8													
0																	
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.	<table><tr><td>128</td><td>64</td><td>32</td><td>16</td><td>8</td><td></td><td></td><td></td></tr><tr><td>0</td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>	128	64	32	16	8				0	1						
128	64	32	16	8													
0	1																
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.	<table><tr><td>128</td><td>64</td><td>32</td><td>16</td><td>8</td><td></td><td></td><td></td></tr><tr><td>0</td><td></td><td>0</td><td></td><td></td><td></td><td></td><td></td></tr></table>	128	64	32	16	8				0		0					
128	64	32	16	8													
0		0															
Our number (21) is larger than the next digit (16), so we enter a '1' and subtract 16.	<table><tr><td>128</td><td>64</td><td>32</td><td>16</td><td>8</td><td></td><td></td><td></td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td><td></td><td></td><td></td><td></td></tr></table>	128	64	32	16	8				0	1	0	1				
128	64	32	16	8													
0	1	0	1														
With only 5 left to convert, which will clearly be made up of a '1' and a '4', we place '1's into each of these columns and '0's into the others.	<table><tr><td>128</td><td>64</td><td>32</td><td>16</td><td>8</td><td></td><td></td><td></td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td></td><td></td><td></td></tr></table>	128	64	32	16	8				0	1	0	1	0			
128	64	32	16	8													
0	1	0	1	0													

The binary equivalent of '85' is '01010101'.

The right-most binary digit is always '1', then '2', '4', '8', etc., doubling each time. As the binary number varies from eight bits, the left-most bit will change, but the right-most bit will always be '1'.

COPYRIGHT
PROTECTED



Binary Addition

Addition of numbers in binary is similar to addition of numbers in denary. The numbers are added one pair at a time, and each pair is added, going from right to left. In binary, when adding two numbers, there are only five possible combinations of numbers, because we're only dealing with '1's and '0's.

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 10 \quad (\text{seems strange, but '10' is binary for '2'})$$

$$1 + 1 + 1 = 11 \quad ('11' \text{ is binary for '3', and you will only need this one for carrying})$$

1. The two numbers to be added are scaled to the same base on top of the other.

$$\begin{array}{r} 10 \\ + 00 \\ \hline \end{array}$$

2. Starting with the right-most digits, the first pair is added together. $1 + 0 = 1$.

$$\begin{array}{r} 10 \\ + 00 \\ \hline \end{array}$$

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

$$\begin{array}{r} 10 \\ + 00 \\ \hline \end{array}$$

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

$$\begin{array}{r} 10 \\ + 00 \\ \hline \end{array}$$

5. Next, we add $0 + 1 + 1$ (the carried '1'). In binary, $0 + 1 + 1 = 10$, so another '0' and another carried '1'.

$$\begin{array}{r} 10 \\ + 00 \\ \hline \end{array}$$

6. Here, it's $1 + 1 + 1$ including the carried digit. In binary, $1 + 1 + 1 = 11$.

$$\begin{array}{r} 10 \\ + 00 \\ \hline \end{array}$$

7. Again, $1 + 1 + 1 = 11$.

$$\begin{array}{r} 10 \\ + 00 \\ \hline \end{array}$$

8. Now, $0 + 0 + 1$ (the carried '1') gives us '1'.

$$\begin{array}{r} 10 \\ + 00 \\ \hline \end{array}$$

9. The final pair on the left is $1 + 0 = 1$.

$$\begin{array}{r} 10 \\ + 00 \\ \hline \end{array}$$

**COPYRIGHT
PROTECTED**



Overflow – occurs when the value that results from a calculation requires more bits than are available. An example of this would be 11111111 (255) plus 00000010 (2), requires nine bits, so overflow would have occurred if only eight bits were available.

Hexadecimal



Hexadecimal – a numbering system with 16 possible values for an individual digit. Denary has 10 possible values (0, 1, 2, 3, 4, 5, 6, 7, 8, 9); hexadecimal has 16 possible values (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F).

One hexadecimal digit can always be translated to four binary digits. Hexadecimal is quicker to write and less prone to being misread.

Binary representation	Decimal representation
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

To convert from binary to decimal:

1. This is a binary number we will convert to hexadecimal.
2. If the binary number has a number of digits divisible by four (four-digit, eight-digit, 12-digit, etc.), it can be left alone. Otherwise, add '0's to the left until you have such a number. Since our number has six digits, we will add two '0's to the left of it.
3. Next, split the number into 'nibbles' of four bytes each.
4. Finally, convert each nibble separately, using the table above. This table contains every possible value for a binary nibble.

So '011110' in binary is equivalent to '1E' in hexadecimal.

To convert from hexadecimal to binary:

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

So 'A6' in hexadecimal is equivalent to '10100110' in binary.

**COPYRIGHT
PROTECTED**



If you need to convert between decimal and hexadecimal numbers, the best way is to use a binary number, so either **decimal → binary → hexadecimal** or **hexadecimal → binary → decimal** depending on the conversion you are asked to make.

Dealing with binary and hexadecimal conversions, as well as binary arithmetic, is more about practising than memorising. Try the following:

- Adding together two random binary numbers, then converting each of the results back to denary to check that the addition was correct.
- Write multiple-choice questions for a revision partner; you'll get plenty of practice writing out the answers yourselves, and trip them up with convincing (yet incorrect) answers.

Binary Shifts

Binary shift – moving the bits of a binary number left or right:

a. 00011001 Shift right by one place 00001100
 0001100 Shift left by two places 001100

For each place you shift left ←, you multiply a binary value by two. For each place you shift right →, you divide a binary value by two.

Overflow – occurs when the number you want to store is too big for the storage space available. It can happen during addition, and it can also happen during shifting:

00110000 ← This is the original number, which is 48
 01100000 ← A shift left by one place doubles it to 96
 11000000 ← A second shift left doubles it again to 192
 10000000 ← The leftmost '1' is lost – this value is actually 128

Overflow has occurred here because the number that *should* have resulted from doubling 192 (384) cannot be stored using eight bits (which can store a maximum of 255).

Characters

Character – a single symbol, such as a letter, number, symbol or space character. A single character can cause one character to appear on the screen.

Character set – a list of all characters recognised by a computer system, each with a corresponding code, and the same codes are used by all computers that use the same character set. Examples include **ASCII** (American Standard Code for Information Interchange) and **Unicode**.

Character	ASCII value
A	0100 0001
a	0110 0001
#	0010 0011
3	0011 0011

ASCII uses eight bits, allowing 256 different characters to be represented (2⁸). Unicode uses 16 bits, allowing 65,536 characters to be represented. Using Unicode instead of ASCII gives more storage space for more alphabets, including Chinese, Japanese, Arabic and Russian, but more storage space is needed.

**COPYRIGHT
PROTECTED**

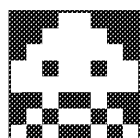


Images

One way to store images is to divide them into pixels, each of which is a tiny dot of one colour, and when a picture is saved, the colour of each individual pixel is stored. The more pixels that are used to store each pixel, the more colours are potentially available.



Pixel – short for *picture element*, this term refers to the smallest possible unit of a screen. A pixel cannot be divided up into smaller units, and a pixel can only have one colour at a time.



```
1 1 1 0 0 1 1 1
1 1 0 0 0 0 1 1
1 0 0 0 0 0 0 1
0 0 1 0 0 1 0 0
0 0 0 0 0 0 0 0
1 1 0 1 1 0 1 1
1 0 1 0 0 1 0 1
0 1 0 1 1 0 1 0
```

When working with **monochrome** images, each pixel is either on or off, so one bit can represent one pixel.

The image above is eight pixels by eight, so 64 pixels in total. Only black and white are enough to represent each pixel, set to '0' for black or '1' for white. This means 64 bits are enough to store the data for the pixels of this image.

If more colours are needed, more bits are needed. Many images store 24 bits, like this way:

First byte	Second byte	
11111111	10001011	
Red	Green	

The 'red' value in the first byte is as high as it can be, so there will be lots of red in the image. There will be some green, but not as much as red, and there will be no blue at all as the third byte is all zeros. Over 16 million colours are available, but a 64-pixel image saved in this format would take up a lot of space compared with eight for the image above.

The amount of storage required for an image depends on a number of factors, including:

- **Colour depth** – a measure of how many colours are available; the more colours, the more bits that must be assigned to store each pixel.
- **Resolution** – the number of pixels in *height* and *width* for an image. A higher-resolution image (more pixels) requires more storage space than a lower-resolution image.



Metadata – as well as storing information about each pixel, an image file can also store information about the image file as a whole. The term 'metadata' describes data'. This might include:

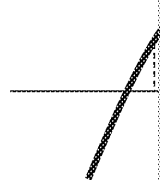
- The colour depth
- The resolution
- The date/time the file was created/edited
- The name of the image's creator

**COPYRIGHT
PROTECTED**



Sound

Sound is **analogue data**, meaning it is not digital. Analogue data needs to be converted to digital in order to be stored and processed. This is done by an **analogue to digital convertor (ADC)** by taking regular samples of the analogue data. With sound, thousands of samples are taken per second.

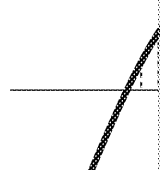


(each dot



Sampling frequency – a measure of how often a sample is taken, measured in samples per second. 1 MHz (megahertz) means one million times per second. A higher sampling frequency represents a higher sampling frequency.

A higher sampling frequency results in a better-quality audio file but requires more storage space.



Sample size – refers to how many bits are required to store each sample. A larger sample size means a higher **bit rate**, which is how many bits are required per second of sound.

Larger sample sizes mean more bits per sample. This allows more accurate recreation of the original sound but increases the file size needed.

1.2.5. Compression



Compression – techniques to reduce the size of a file, so that it takes less time to be transmitted across a network more quickly. There are different types of compression: lossy and lossless.

Type	Description	Examples
Lossy	An individual file is made smaller, although not all data is stored. As such, some data is permanently lost.	JPEG image files (not all pixels were originally, but with pixels so close together it's not obvious to the human eye) MP3 audio files (not all frequencies are stored) H.264 video files
Lossless	An individual file is made smaller, but all data is stored. When uncompressed, the file will be exactly the same as the original. Lossless files must be stored in a lossless format as loss of data would not be acceptable.	PNG image files GIF image files (although they only work with 256 colours)

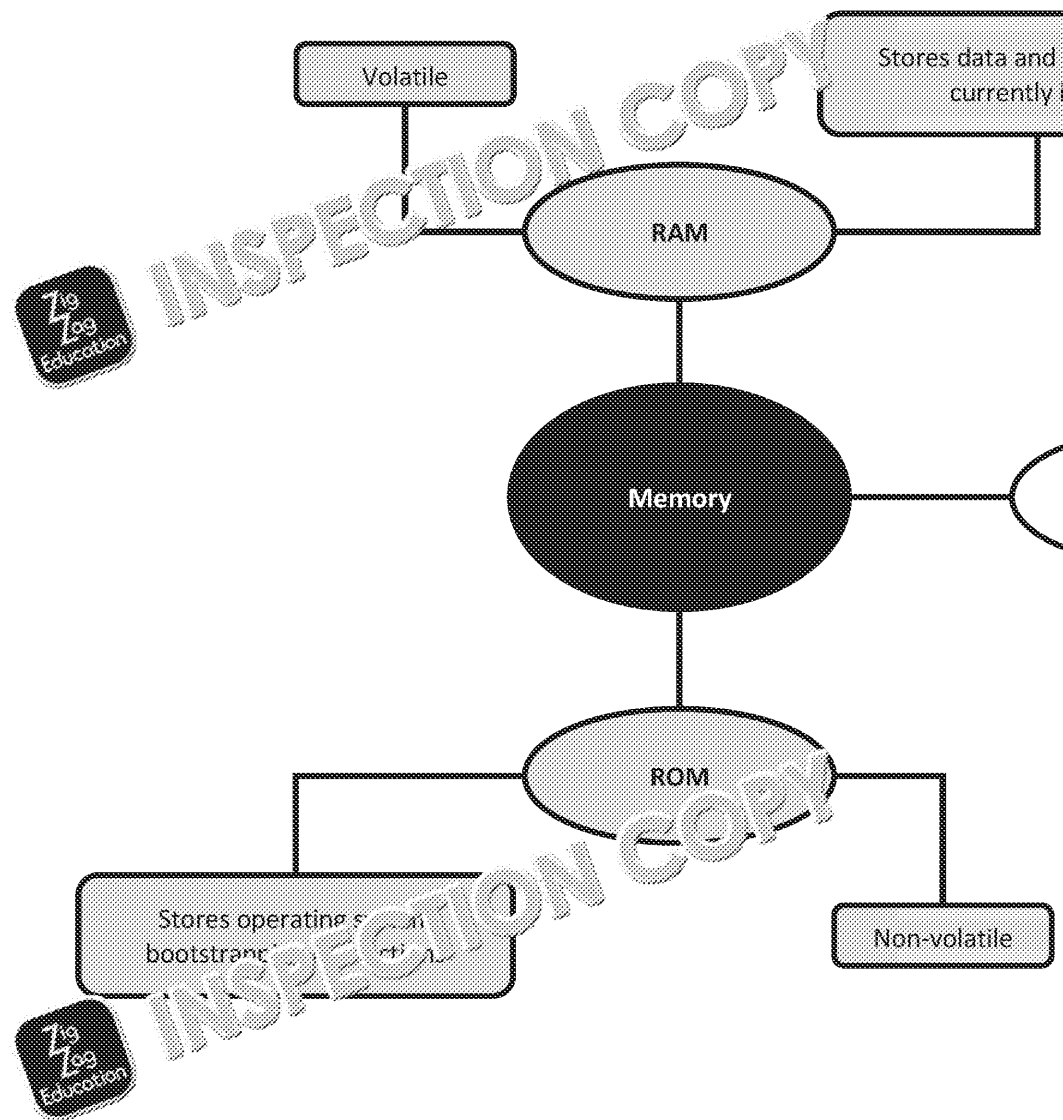
Choice of compression type might depend on a number of factors:

- If data needs to be precise, such as in money transactions, lossless compression is preferred.
- If data does not need to be precise, typically as in photographs or music, lossy compression is considered to save disk space or transmission time.

COPYRIGHT
PROTECTED



Memory Mind Map

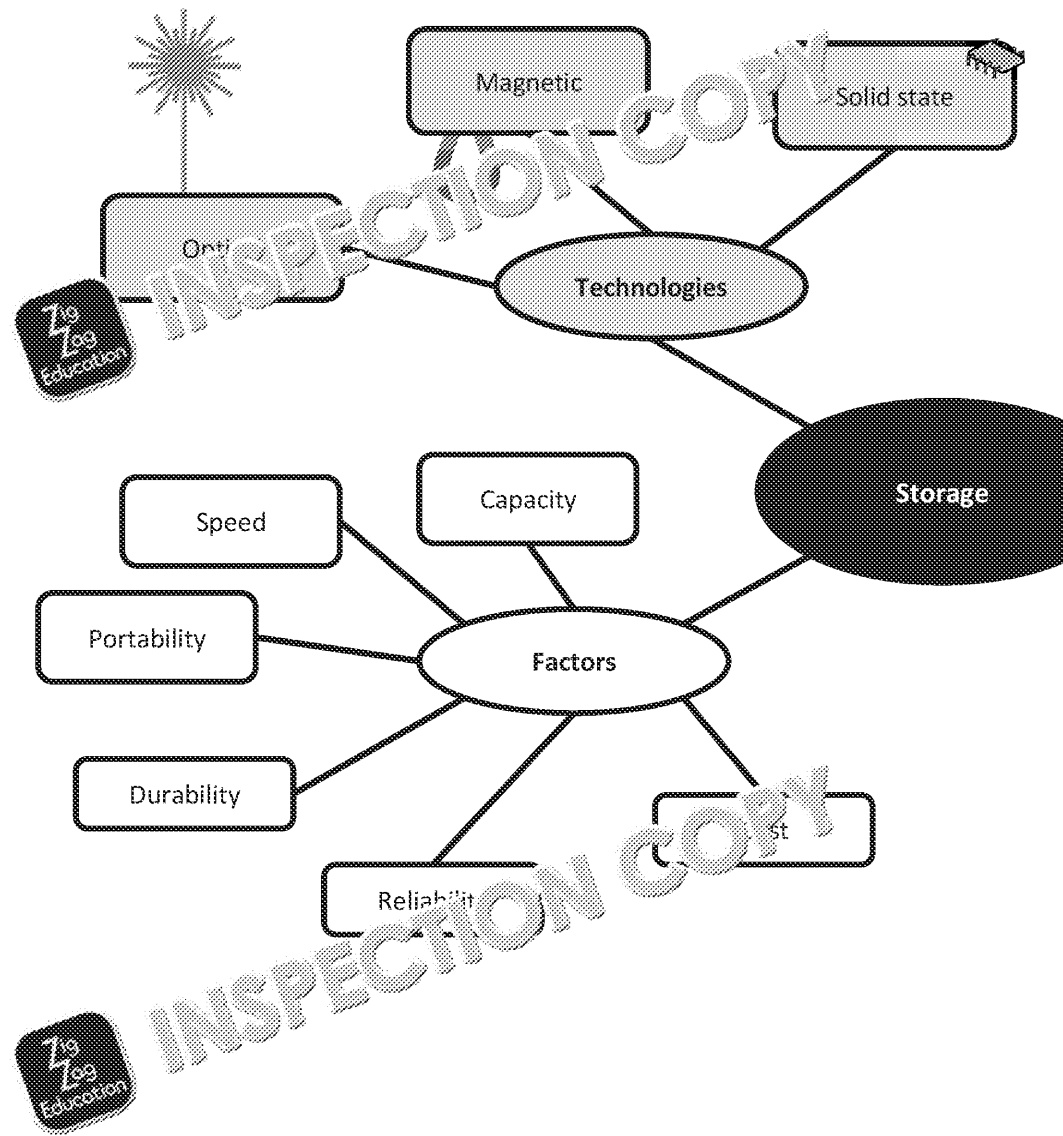


INSPECTION COPY

COPYRIGHT
PROTECTED



Storage Mind Map

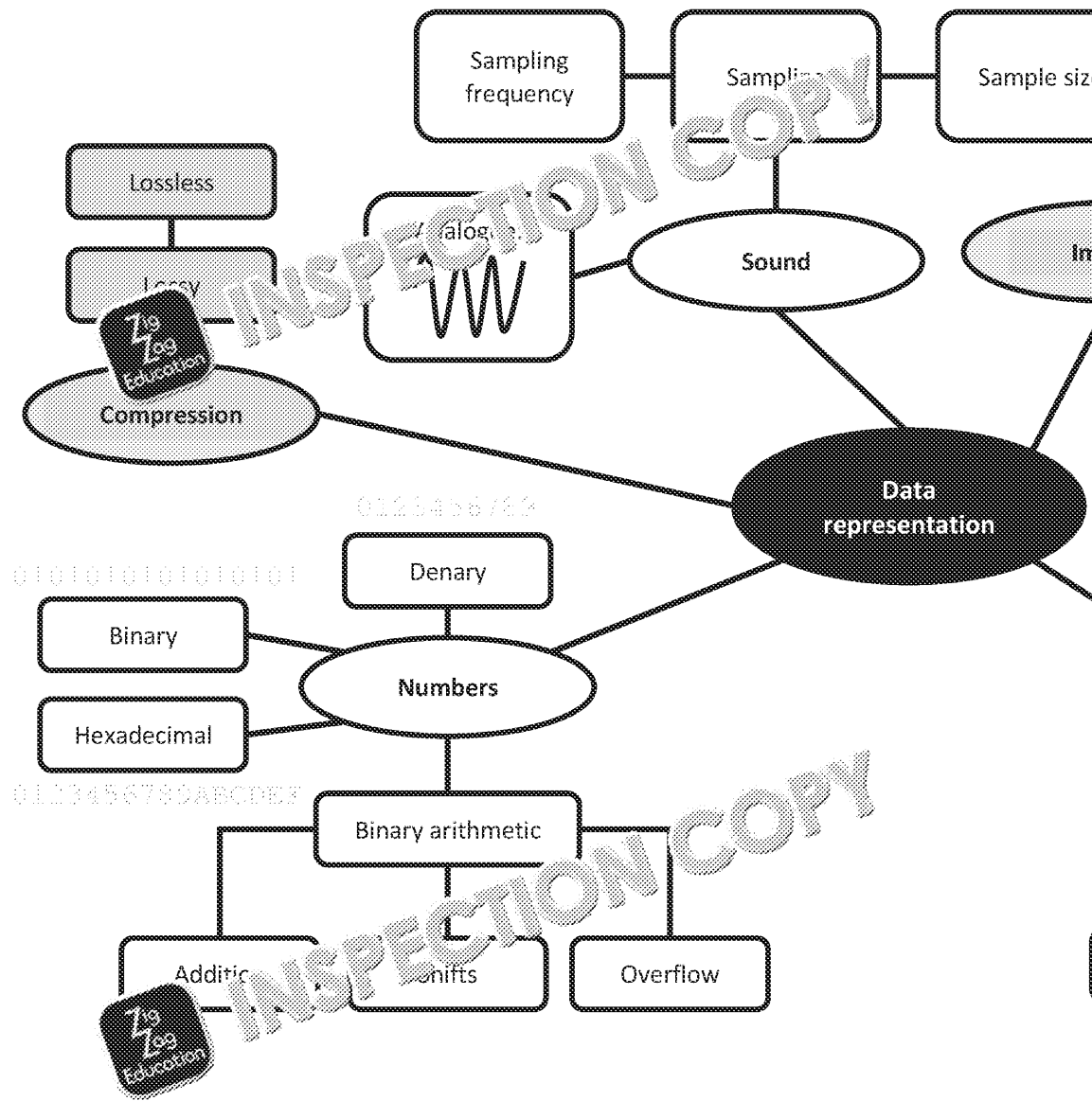


INSPECTION COPY

COPYRIGHT
PROTECTED



Data Representation Mind Map



INSPECTION COPY

COPYRIGHT
PROTECTED



Sample Examination-style Questions

3. Describe the difference between RAM and ROM, providing one example of each type of memory.

.....

.....

.....

.....

Example of data in RAM

.....

.....

.....

Example of data in ROM

.....

.....

4. A student needs to store their data in a way that means it can be accessed from a college.

- a. Identify **three** characteristics of secondary storage devices and state them for storing the student's data.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....



.....

.....

.....

- b. Recommend an appropriate secondary storage device.

.....

INSPECTION COPY

COPYRIGHT
PROTECTED



5. The binary number 10011011 can have several values when translated into

- a. Convert 10011011 to denary, assuming it is an unsigned binary integer.

.....

.....

- b. Convert 10011011 to hexadecimal.

.....

.....

6. a. Carry out an arithmetic shift left by two places on the binary number 00011000.

.....

.....

- b. State the effect of an arithmetic shift left by two places.

.....

.....

- c. Describe the effect of carrying out an arithmetic shift left by four places on 00011000. The result is stored in an eight-bit binary register.

.....

.....

.....

.....

7. a. State what is meant by the term **character set**.

.....

.....

- b. The ASCII code for 'P', in denary, is 80. What is the corresponding code

.....

.....

**COPYRIGHT
PROTECTED**



8. Each pixel in an image uses eight bits of storage, and the resolution of the image is 1024 by 768 pixels.

a. i. What is meant by the term **colour depth**?

.....

.....

ii. State the maximum number of different colours the image can represent.

.....

b. Calculate the minimum amount of storage space that will be required for the image. Give your answer in **kilobytes**.

.....

.....

.....

9. Describe what is meant by the term **virtual memory**.

.....

.....

.....

.....

**COPYRIGHT
PROTECTED**



1.3. Computer Networks, Connections

1.3.1. Networks and Topologies



Local Area Network (LAN) – connects devices together within a building or buildings, such as a school or college campus. Usually, in a LAN, all computers are privately owned.



Wide Area Network (WAN) – connects devices together across a large geographical area, which can consist of the entire planet. The Internet is an example of a wide area network.

Network Performance

There are several ways in which network performance might be defined, but usually it refers to data to be sent and received both quickly and accurately, across the office or across the world.

Measure	Definition	Contributing Factors
Bandwidth	The maximum amount of data that can be transferred across a connection in a given time, e.g. bits per second.	<ul style="list-style-type: none">• Connection type – fibre-optic is faster than copper connection.• Number of users – the more users, the slower each individual user's connection.
Latency	The length of delay between data being transmitted and subsequently received, measured in seconds or milliseconds.	<ul style="list-style-type: none">• Network traffic – more traffic can result in greater latency.• Number of nodes – if a signal has to pass through many devices on the way to its destination, latency will increase.
Error rate	Measured as a percentage, the amount of data that is not received exactly as it was transmitted.	<ul style="list-style-type: none">• Interference – other signals can corrupt data; a '1' might be received as a '0'.• Attenuation – a signal that is not amplified can fade.

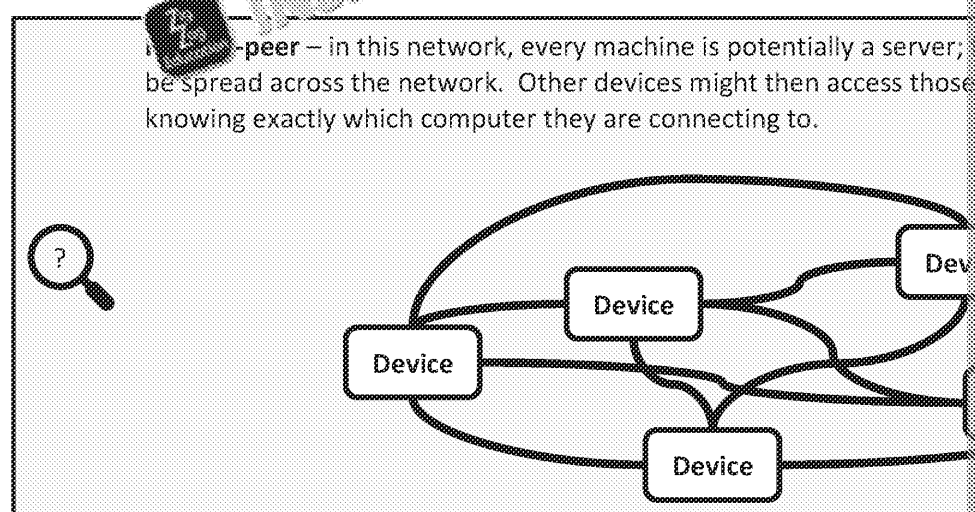
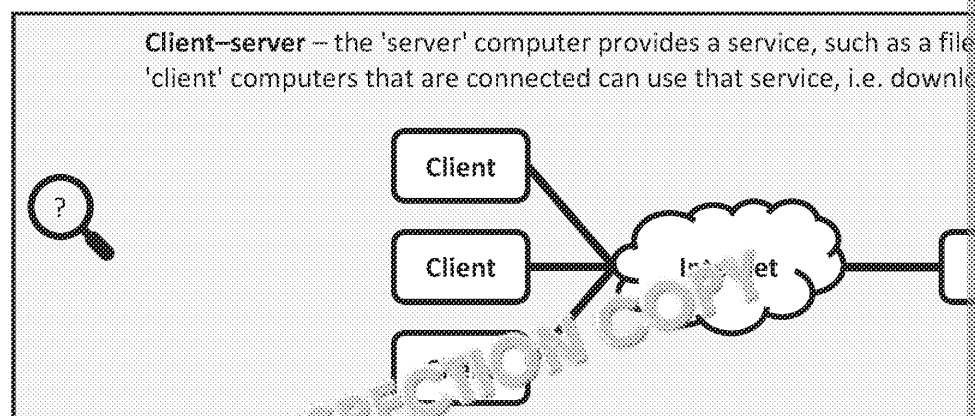
INSPECTION COPY

COPYRIGHT
PROTECTED



Client-Server and Peer-to-Peer

'LAN' and 'WAN' refer to the scale of a network, but with either of these, there are two network models that might operate. There are two **usage models** you need to be familiar with.



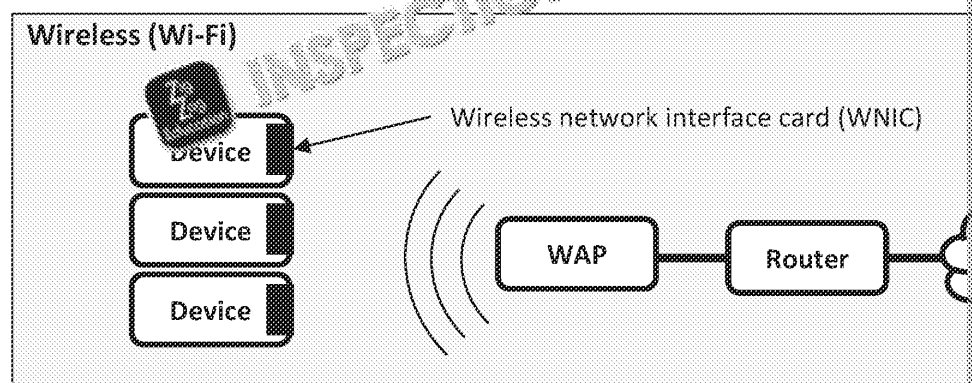
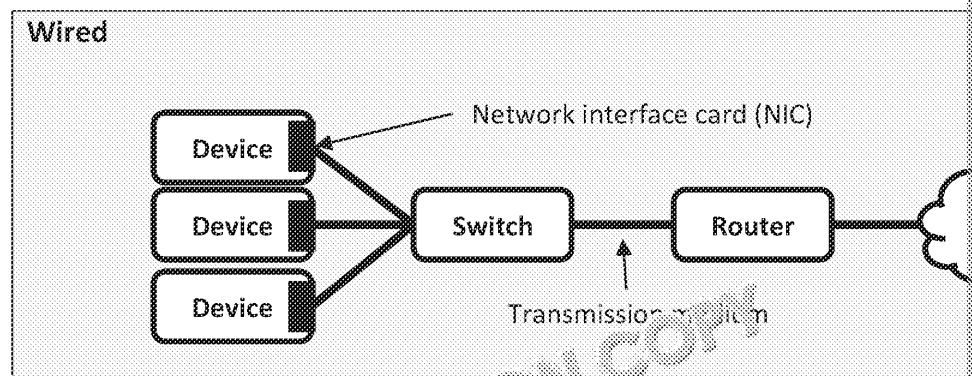
Client-server	Peer-to-peer
<ul style="list-style-type: none"> + Files are stored centrally, so are easier to manage (edit, back up, etc.) + For other resources, such as a printer, these are also easier to manage centrally + Licences for software are managed centrally, so less risk of a user working from an unauthorised copy 	<ul style="list-style-type: none"> + Can be set up with less need for specialist hardware + The hardware is typically less expensive + The failure of one device has less consequences + A specialist network manager is not required
<ul style="list-style-type: none"> - Failure of a server can potentially take down all file access or all printing - A single server may struggle to deal with the needs of a large number of clients 	<ul style="list-style-type: none"> - Security is not centralised, so the network may be more vulnerable - The computer of a user who is slow at accessing files can slow down the network

**COPYRIGHT
PROTECTED**



Connecting to a Network

Networks can be wired or wireless. Each of these arrangements requires a slight



Component	Purpose
Device	Any device that can connect to a network, such as a laptop or smartphone.
Switch	Allows connection to multiple devices on the same network. It can communicate with one another.
Router	Provides a connection to the Internet, allowing different networks to communicate. In the home, routers and switches are usually combined. A combined router acts as both a router and a wireless access point.
Transmission medium	Any means by which two or more devices can be connected. Examples include Ethernet cable. The plural of 'medium' is 'media'.
(Wireless) Network interface card	A computer component that handles network communication. In a wired network, this is where the Ethernet cable would plug in. In a wireless network, this card can receive and transmit data via radio waves.
Wireless Access Point (WAP)	Transmits and receives radio waves to/from devices. It can be connected to a router, but in a large office or public space, it would be connected to a switch, which would in turn be connected to a router.

**COPYRIGHT
PROTECTED**



The Internet

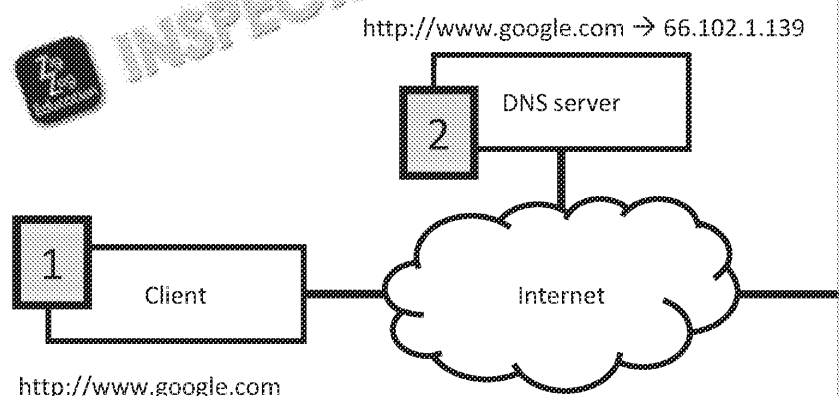


Internet – an interconnected collection of networks spanning the globe. Every piece of hardware that is part of this connection, including computers and routers. This is not to be confused with the **World Wide Web**, which uses the Internet.

When you want to access a web page, you type the URL of that page into a browser. For example, <http://www.google.com>. This triggers a series of events that ends with the page displayed in your browser.



URL – Uniform Resource Locator. Resources on the Internet, be they web pages, are identifiable by their unique 'IP' address.



1. Your computer ('client') does not know the **Internet Protocol (IP) address** of <http://www.google.com> so it asks the DNS server. In order for this process to work, the client needs to know the IP address of the DNS server.



Internet Protocol (IP) address – a unique number used to identify every device on the Internet. No two IP addresses are the same. If your computer requests a web page, the IP address specifies where that web page should ultimately be delivered.

2. The DNS server contains a list of URLs and corresponding IP addresses. The client looks up the URL, and an IP address is found and returned to the client machine (66.102.1.139). If no matching link is found, the URL may be forwarded to an alternative server.
3. Now that the client machine has the IP address, it can access the web page. The client sends a request to the server, and a copy of their main page is transmitted to, and displayed upon, the client.



Hosting – provision of storage space online on a web server. If you want to store images and other associated files would be stored on a **server**, which is a computer that provides a service (access to a website being an example of a service). **Clients** access those services, perhaps by requesting content such as videos or images.



Cloud computing – this involves storage on remote computers, called **clouds**, or other organisations. When a file is saved or loaded, it is transmitted to the allocated server, and multiple backups of files often exist around the world. Files can also be processed remotely. When something is stored 'in the cloud', it can be accessed from another device, or perhaps across several devices.

**COPYRIGHT
PROTECTED**



Topologies



Network topology – the pattern in which the hardware on a network connections. Common topologies include star and mesh.

Topology	Explanation
<p>R = router S = switch</p>	<p>Star Every device is connected to a switch or router. Communication travels from the device to the switch/router and then to the destination device.</p> <ul style="list-style-type: none"> + Very few data collisions, since all communication goes through the switch/router + Strong, centralised security - Lots of cabling needed - If the switch/router has no spare ports, adding more devices can be difficult
<p>Full mesh:</p> <p>Partial mesh:</p>	<p>Mesh All devices are connected to all other devices in the network. If not all potential devices are connected, it is called a partial mesh topology.</p> <ul style="list-style-type: none"> + High resilience – multiple cables mean that if one cable fails, communication can still get through + Perfect for implementation of load balancing - If cables are used instead of wireless, a large amount of cabling is required - Adding a single device can be difficult as many connections needed - If connections do start to fail, it can be difficult to troubleshoot meaning it won't be repaired quickly

INSPECTION COPY

COPYRIGHT
PROTECTED



1.3.2. Wired and Wireless Networks, Protocols and Layers

Connecting to a network can take place using a **wired** or **wireless** mode of connection. A wired connection might make use of Ethernet cabling, and a wireless connection might use Wi-Fi or Bluetooth. Here are some advantages and disadvantages:

Wired	
<ul style="list-style-type: none"> + Transmission is usually quicker, though that can depend on other factors as well + Wired transmission is less prone to interference than wireless transmission 	<ul style="list-style-type: none"> + Easy to add a device as no cabling is required + A user can move around more easily as a teacher taking a lesson to a different classroom
<ul style="list-style-type: none"> - If enough devices are already on the network, there may be nowhere to connect a new device - Installing or repairing a wired network can be more expensive 	<ul style="list-style-type: none"> - Radio signals that are transmitted have a limited range and a shorter range than wired connections - With signals transmitted, they are more prone to interference - Signals are more likely to be intercepted

It's easy to use the table above to create additional advantages and disadvantages. For example, 'a user can move around more easily' is an advantage of a wireless connection, so 'cannot move around easily' would be a corresponding disadvantage of a wired connection.



Encryption – the process of converting **plaintext** into **ciphertext**. Any signals sent over a network should receive only the ciphertext, which is scrambled, with no one being able to (unscramble) it.

In order for computers and other devices to communicate with one another, they need to be identifiable. Effectively, each device needs an address at which it can be found, known as an **IP address** and a **MAC address**.



Internet Protocol (IP) address – a unique number that is used to identify a device connected to the Internet. Since this number is assigned by software, if you reboot your computer or rejoin it, you might have a different IP address.



Media Access Control (MAC) address – another unique number that is assigned to your network interface card. This can only be changed by replacing your network interface card.

COPYRIGHT
PROTECTED



Standards



Standard – an agreed way of working, developed in order to make it easier for software, including from different developers, to intercommunicate.

Examples of standards cover both hardware and software, including:

- The ASCII character set, which is interpreted identically across billions of devices
- HTML – a markup language that is interpreted by multiple browsers
- File formats, such as JPEGs, enabling cameras and computers to process the data
- USB connections, to maximise the compatibility between hardware devices

Adhering to standards when creating a new device or application makes it more likely that the device or application will be able to communicate with other devices or applications (being more popular).

One category of standard is **communication protocols**.



Protocol – a set of rules governing how a computer communicates on a network. There are many protocols, each necessary for a different purpose (email, access to the Internet, etc.). Without protocols, communication between computers would be impossible.

- **TCP/IP** – *Transfer Control Protocol and Internet Protocol*. These are two protocols that govern how data is sent across the Internet. Their collective role is to break up data into **packets**, each of which is a chunk of data that has been sent from and where it is to be delivered to.
- **HTTP** – *Hypertext Transfer Protocol*. This is the set of rules governing how hypertext (the World Wide Web) is moved around the Internet, from device to device.
- **HTTPS** – *HTTP Secure*. This protocol encrypts data that is sent across the Internet so that it cannot be read if intercepted, so is favoured when sending passwords or credit card numbers.
- **FTP** – *File Transfer Protocol*. This is how files are moved from one computer to another. This protocol is heavily relied upon in building websites, moving files from the web server, from where they can be accessed publicly.
- **POP3** – *Post Office Protocol Version 3*. This is a protocol for accessing email. It allows Microsoft Outlook to take the emails from an online location, opening them up for use.
- **IMAP** – *Internet Message Access Protocol*. This email protocol is used to allow multiple devices (tablets, phones, etc.) to access the same email account.
- **SMTP** – *Simple Mail Transfer Protocol*. While POP3 might be used to retrieve emails, SMTP is used to send them from one server to another.

INSPECTION COPY

COPYRIGHT
PROTECTED



INSPECTION COPY

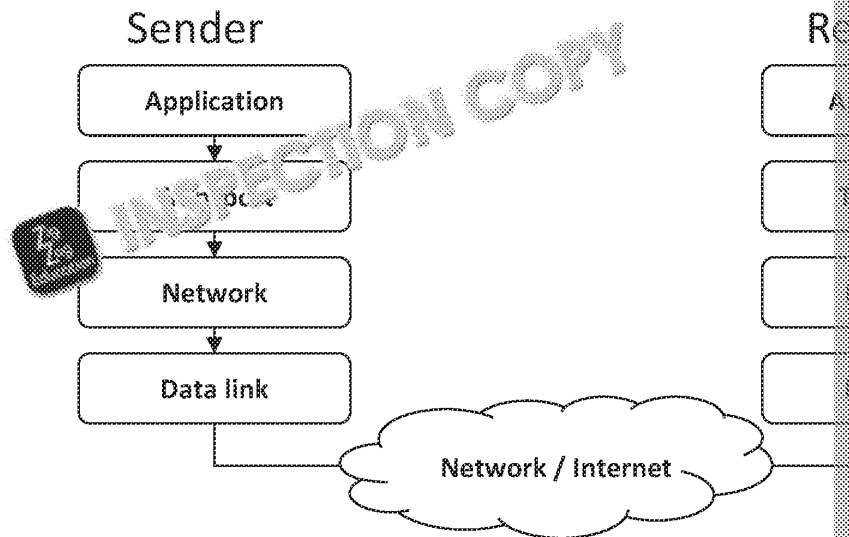


Layers



TCP/IP stack – a series of protocols. When they work together, they move data from one computer, through any number of pieces of network hardware, to another computer. The stack is a *concept*, not a physical thing.

This stack has four layers, each containing a number of protocols. When data is sent down the stack and repackaged the data into smaller units, before passing the data to the network. When data is received, those units are reassembled as they move up the stack.



Layer	Protocols at this layer
Application layer	HTTP, HTTPS, FTP, SMTP, IMAP, POP3
Transport layer	TCP
Internet layer	IP
Network interface layer	Ethernet, Wi-Fi

There are benefits to developing systems using this model:

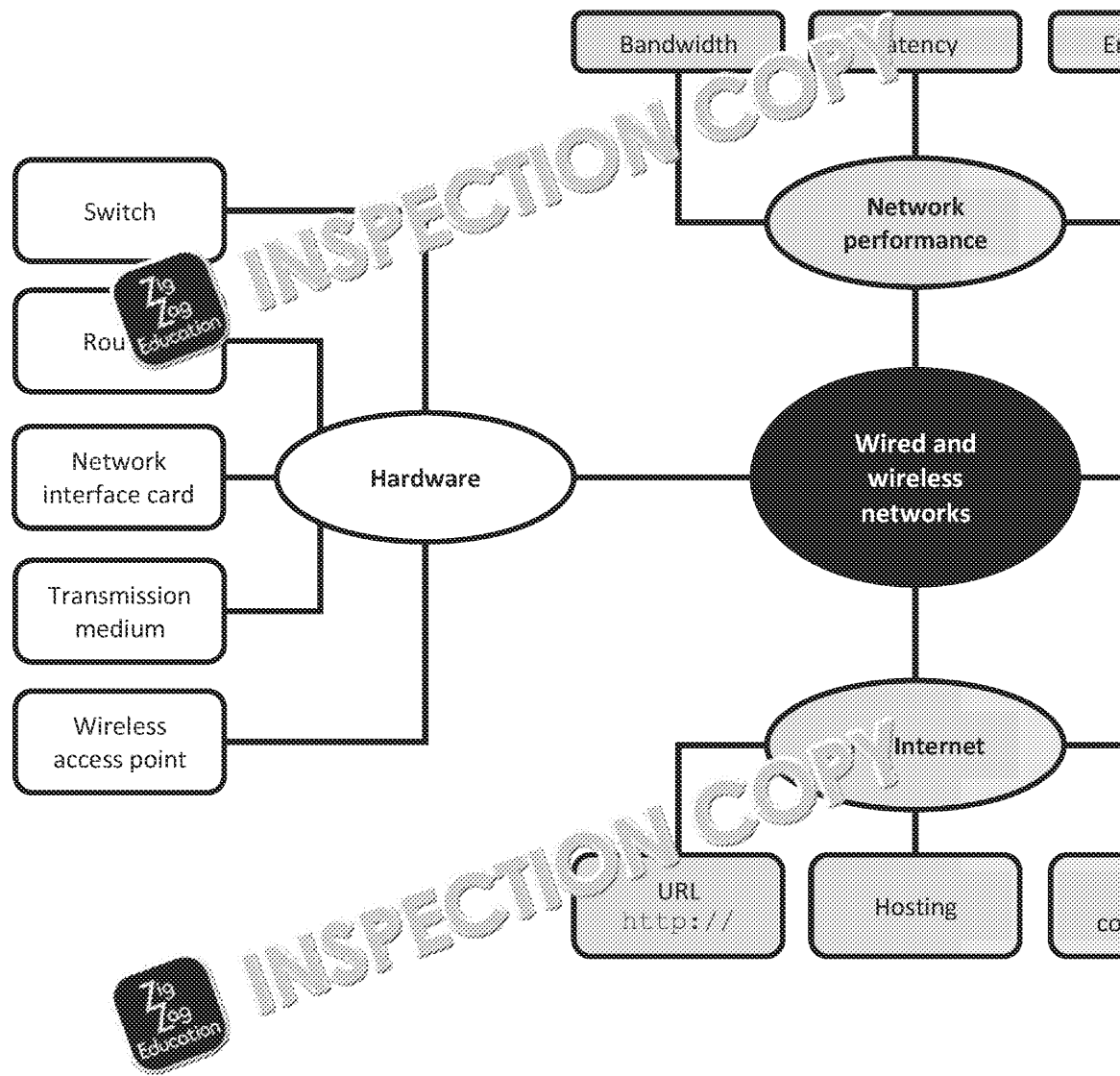
- Different developers can be assigned to different aspects of a system
- Part of a system can be removed and altered without affecting the rest of the system

INSPECTION COPY

COPYRIGHT
PROTECTED



Wired and Wireless Networks Mind Map

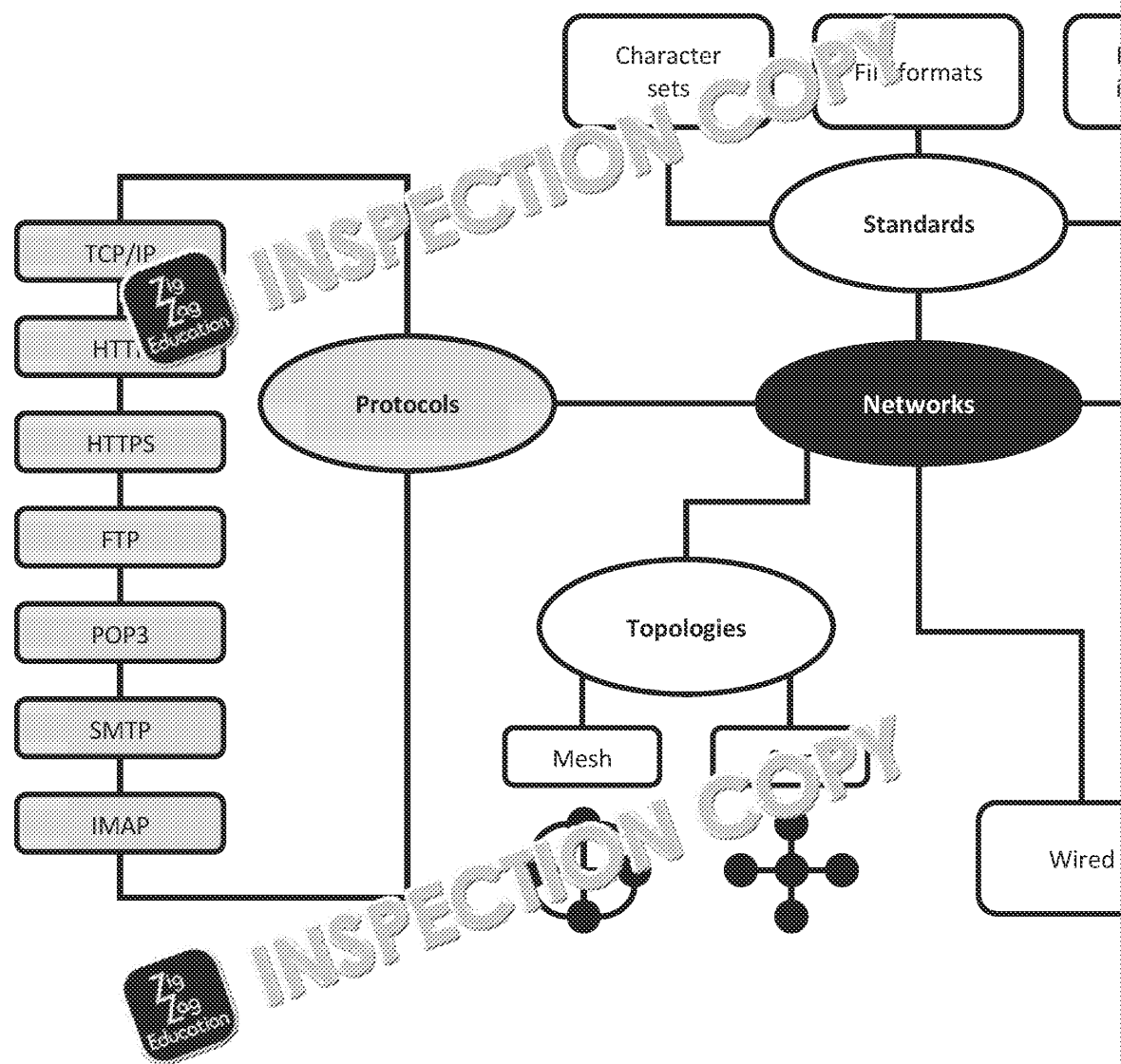


INSPECTION COPY

COPYRIGHT
PROTECTED



Network Topologies, Protocols and Layers Mind Map



COPYRIGHT
PROTECTED



Sample Examination-style Questions

10. Computers can connect to the Internet using cables. Name and describe two that can be used to connect a desktop computer to the Internet.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

11. Outline the difference between a switch and a router.

.....

.....

.....

.....

.....

12. Outline the purpose of each of the following in accessing a web page:

Uniform Resource Locator (URL)

.....

.....

Web server

.....

.....

INSPECTION COPY

COPYRIGHT
PROTECTED



13. a. Define the term **protocol** when used in the transmission of data in a network.

.....

.....

b. Describe **two** different protocols that are used with email.

.....

.....

.....

.....

.....

.....

.....



INSPECTION COPY



INSPECTION COPY

INSPECTION COPY

COPYRIGHT
PROTECTED



1.4. Network Security

1.4.1. Threats to Computer Systems and Networks

Keeping your data safe, whether on your own device or in the cloud, is vital. Failing to protect your passwords, financial data, or even money. For businesses, inadequate security can result in data being issued, as well as a loss of business.

Malware



Malware – any program that works against the interests of you or your organisation. Viruses, Trojans, adware and spyware are types of malware, although there are many others.

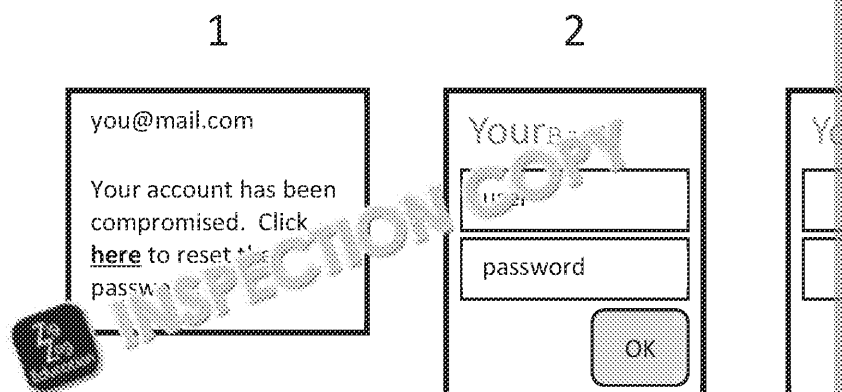
- **Viruses** attach themselves to a file (typically executable files) in an attempt to spread to other files. They can damage your data, but only if you open the file they are attached to.
- **Worms** are very much like viruses, but they do not need to attach themselves to a file and can damage a system without a person opening a file.
- **Trojans** or **Trojan horses** are legitimate programs developed with the intent to be useful. Since they are largely legitimate, they are often not recognised as malware.
- **Adware** downloads unwanted Internet adverts, often observing your online activity to serve specific adverts.
- **Spyware** covertly obtains sensitive data, such as credit card numbers and passwords.

Social Engineering: 'People as the Weak Point'



Social engineering – forms of cyberattack that focus on people, rather than technology. It exploits the weak point in any system.

- **Pretexting** involves fabricating a scenario in order to gain unauthorised access. For example, a hacker might pretend to be from IT support in order to persuade an employee of a company to provide login details.
- **Shoulder surfing** is simply watching someone, over their shoulder, as they type in sensitive information.
- **Phishing** uses emails to lure people to convincing but fake web pages. They look legitimate, but they're really transmitting their login details to an unknown person. Here's how it works:



1. The victim receives an email with a hyperlink. The email tells the user that their account has been compromised and provides a hyperlink, often saying that their security has been compromised in order to make them feel urgent.
2. They will be taken to a screen that asks them to enter personal information. This screen is designed to look identical to a screen with which they are familiar.
3. When they have entered the information, they are usually forwarded to a confirmation page. In the meantime, the information they entered has been transmitted to a hacker.

INSPECTION COPY

COPYRIGHT
PROTECTED



Other Attacks



Brute-force attack – using a program to crack a password by trying every possible combination of characters. Longer passwords, and those that use symbols, numbers and letters, are harder to crack in this way.



Denial of service attack – requesting access to a website (or other online service) repeatedly. The web server is unable to keep up, meaning no one can access the website. Who does this usually controls multiple computers, without their users knowing, and makes repeated requests.



Data interception – when data is transmitted between two devices, it can be intercepted by other devices. Confidential information, credit card details and passwords are of great interest to people using digital technology.



SQL injection – SQL is a language used to manipulate the contents of a database. An attacker smuggles a command into a system either to provide a copy of the data or corrupt data in-place. This is often done by typing an SQL command, for example, a date of birth.

1.4.2. Identifying and Preventing Vulnerabilities

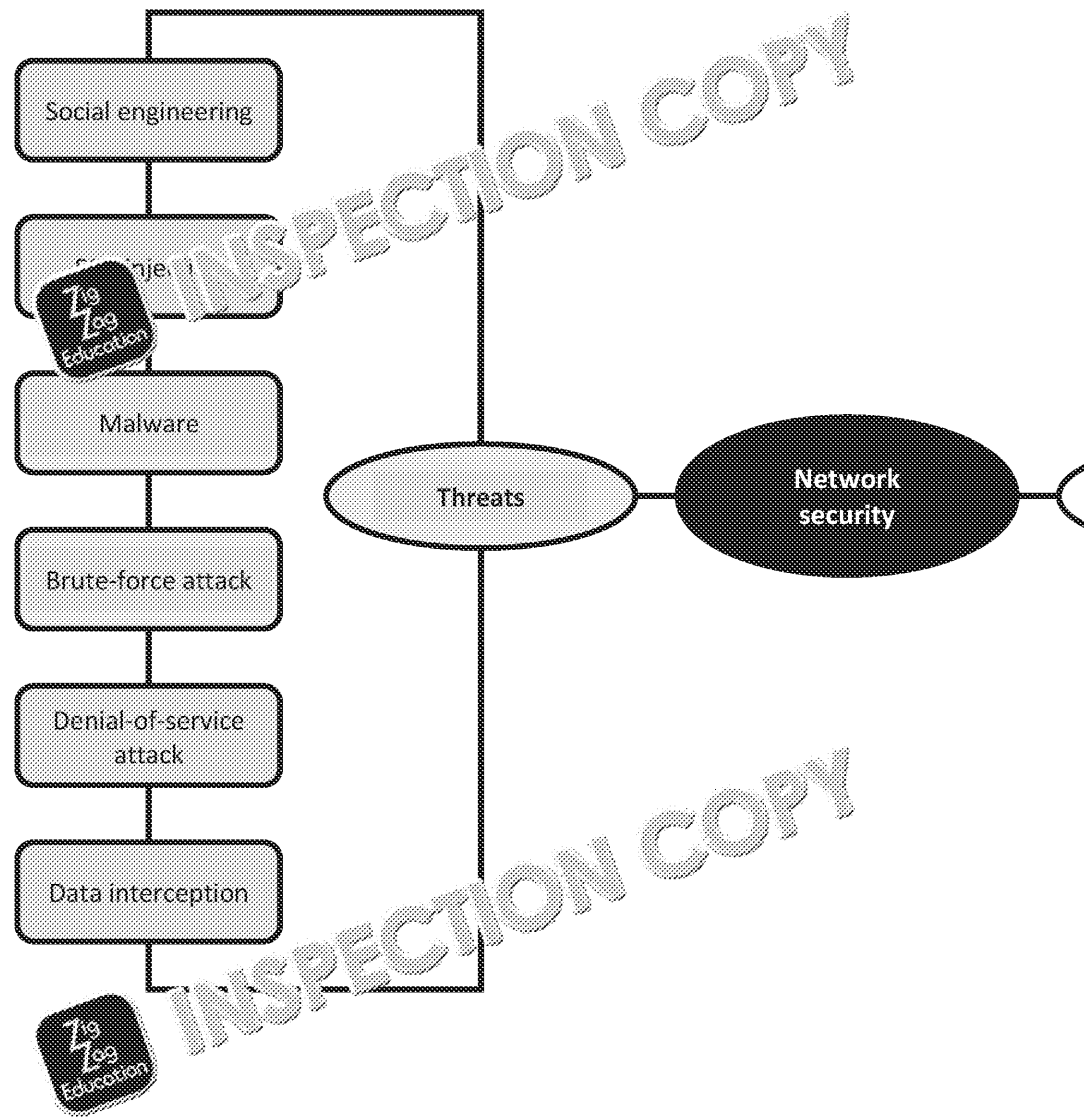
Several tools and techniques are available to guard against attacks:

Penetration testing	Someone tries to hack into a system, but as an employee or friend of the system's owner. Their aim is not to steal or corrupt data, but to find vulnerabilities so that they can be resolved.
Anti-malware software	Also known as antivirus software, anti-malware software scans for malware. If found, it can be deleted, or the affected area of the disk can be restored. Anti-malware software must be regularly updated in order to recognise new threats.
Firewalls	These can be either hardware, software or both. A firewall filters out unwanted traffic. It can block certain traffic (such as all emails or any traffic from a specific IP address) or allow certain traffic (such as from a single, trusted device).
User access levels	Once logged in, users have certain privileges. One user might be able to view their own data; another might be able to look at all the data; a third might be able to edit the data but not change it. These three people have different user access levels.
Passwords	It's good practice to use a strong password, such as \$tr0ngP@ssW0rd\$, and to change it regularly.
Encryption	Scrambling the contents of a file to make it unreadable to anyone who intercepts it. They then <i>decrypt</i> the data, unscrambling it so it can be read.
Physical security	Low-tech means of protecting data, using physical controls and additional technology. For example: <ul style="list-style-type: none"> • Locks on doors • Removal of USB ports, to prevent data theft using removable drives • Security personnel • Storing sensitive data on a device that is not connected to the internet

**COPYRIGHT
PROTECTED**



Network Security Mind Map



INSPECTION COPY

COPYRIGHT
PROTECTED



Sample Examination-style Question

14. The NHS stores a large amount of confidential data about patients. This data is available at healthcare facilities across the country, so it is available online. Security is in place to require users to enter a password in order to view any of the data.

Identify **three** cybersecurity threats to this data, and describe a **different** security measure for **each** threat.

Threat and countermeasure 1

.....

.....

.....

.....

Threat and countermeasure 2

.....

.....

.....

.....

Threat and countermeasure 3

.....

.....

.....

.....

INSPECTION COPY

COPYRIGHT
PROTECTED



1.5. Systems Software

Systems software is responsible for communicating with a computer system's hardware. There are two broad types of systems software, namely the operating system and utility software.

1.5.1. Operating Systems



Operating system – a piece of software that acts as an interface between the user and the hardware, managing all hardware and all other software. If another piece of software is launched from the operating system. Windows, Mac OS, iOS and Android are examples of operating systems.

Operating systems are complex pieces of software, often requiring many years of development. The reason that they are complex is simply that computers are complex, and operating systems are software that requires intelligent management. Among many things managed by the operating system are:

- User interface
- Memory management
- Peripheral management
- User management
- File management

Aspect	Management
User interface	<p>The user interface is the part of any system that allows the user to communicate with the system. Effective user interfaces are easy to learn. Features might include:</p> <ul style="list-style-type: none"> • Icons, which represent tasks, such as a picture of a printer icon that opens the printer settings • Menus, which provide options from which a user can select a task • Messages for the user, including error messages • Windows, each of which can contain a single application window (a generic term, applicable beyond Microsoft Windows) • Pointer or cursor, allowing a user to click and drag <p>A handy acronym to remember these is WIMP, for windows, icons, menus, pointer. The interface can vary based on the system in use. Mobile devices, for instance, do not provide a pointer, and low-specification devices use a system that uses some alternative to windows.</p>
Memory management	<p>When a program is being executed, it is transferred from secondary memory (RAM), and in most modern computers, multiple programs can be in RAM at the same time. In the event that additional working space is required, operating systems use virtual memory, where secondary memory is used as if it were RAM.</p> <p>One role of the operating system is to manage RAM in a way that allows multiple programs that have been loaded. Once enough programs have been loaded, RAM can become full. In these circumstances, the operating system must determine what to unload from RAM in order to make space for new programs.</p> <p>Multitasking operating systems can run multiple processes, but in reality, the focus of the processor is upon multiple processes at once.</p>

INSPECTION COPY

COPYRIGHT
PROTECTED



Aspect	Management
Peripheral management	The operating system can communicate with all connected peripherals), including the VDU, the mouse, the printer and it needs a driver , which is software that tells the operating system one specific device. Operating systems have many drivers, with one operating system to manage data transfer between devices.
User management	Different users can be logged into a computer system. They can have different settings (font sizes, desktop backgrounds, desktop contents, colour schemes) for different programs. Ensuring that this takes place correctly is the job of the operating system. They will also provide for the creation, management and deletion of user accounts, as well as providing security, so users can only access their own files.
File management	Operating systems provide for the following file and folder management: <ul style="list-style-type: none"> • Creation • Deletion • Moving between folders • Naming and renaming <p>Operating systems also determine where, physically on a device, files and folders can be stored.</p>

1.5.2. Utility Software



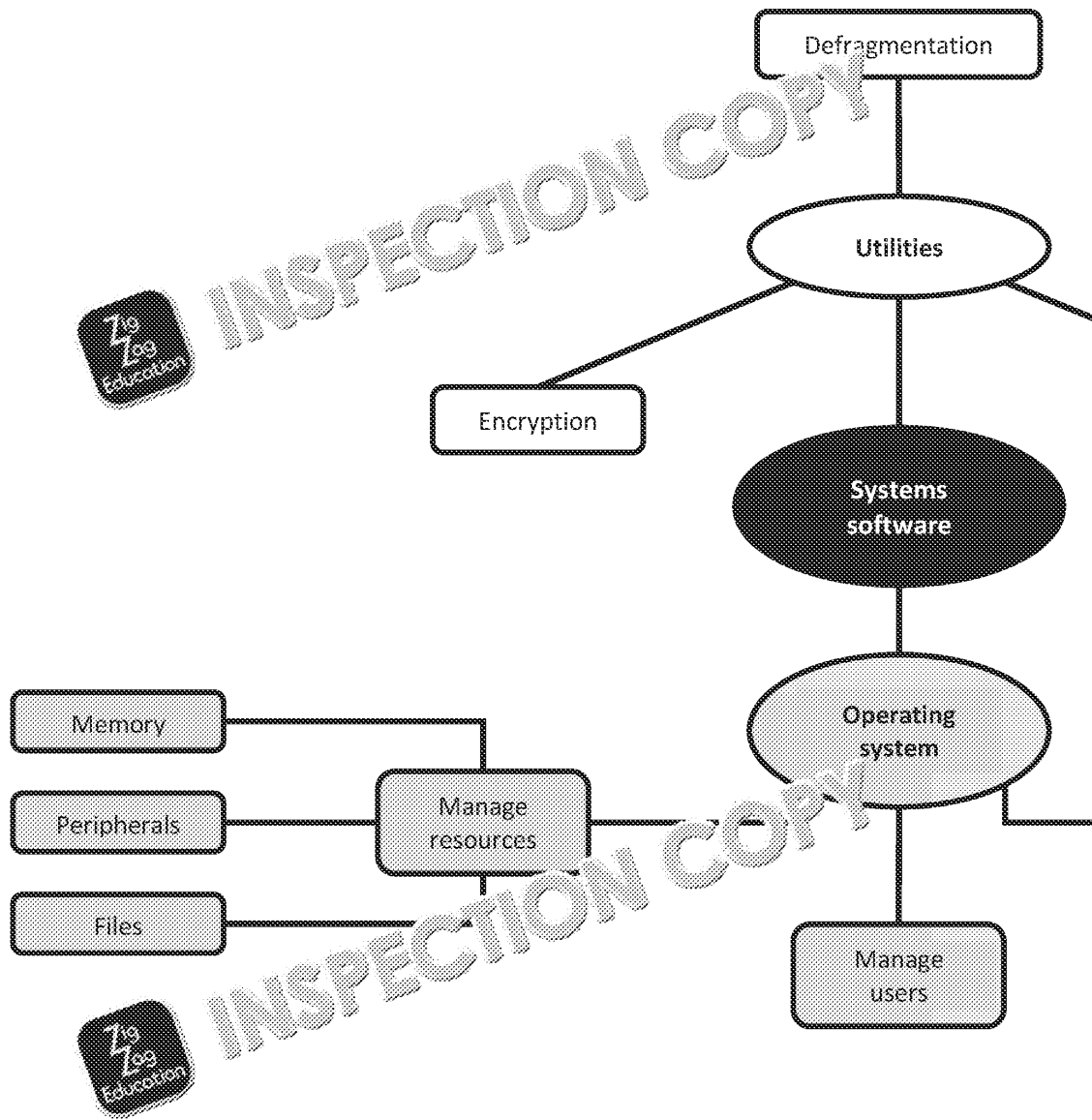
Utility software – programs that keep the computer functioning efficiently by freeing up storage space or by removing viruses.

Utility	Purpose
Encryption	Allowing for data to be scrambled in order to prevent unauthorized users from understanding any files that they see. This might be for secure storage or secure transmission.
Defragmentation	Moving separate parts of a file physically together, to speed up access.
Compression	Reducing the size of a file so that it can be stored using less space or more quickly.

COPYRIGHT
PROTECTED



Systems Software Mind Map



INSPECTION COPY

COPYRIGHT
PROTECTED



Sample Examination-style Questions

15. Describe in detail **three** functions of an operating system.

.....

.....

.....

.....

.....

.....


.....


.....

.....

.....

.....

 INSPECTION COPY

 INSPECTION COPY

INSPECTION COPY

COPYRIGHT
PROTECTED




1.6. Ethical, Legal, Cultural and Environmental Impact of Digital Technology

1.6.1. Ethical, Legal Cultural and Environmental Impact

Ethical Issues



Ethics – this term refers to what is right and what is wrong, although it is not straightforward that they have a single 'right' answer. Often what is right for society as a whole, and vice versa.

Topic	Issues
 Inclusion	<ul style="list-style-type: none"> • There are some groups, especially those of school age, that are around not having the latest technology. • Not everyone in the country has access to the Internet, access to information, to job listings or to a society that is changing. • Government could commit money to solving this inequality but would be spending money unequally. This is an instance where there is no 'right' answer.
Professionalism	<ul style="list-style-type: none"> • There is increasingly an expectation of people to be available 24/7; this is a direct impact of technology. • Although you can apply for jobs internationally, employers are making the process far more competitive. • Social media can be seen by anyone, including prospective employers, blurring the line between private life and professional life.
Artificial intelligence	<ul style="list-style-type: none"> • Artificial intelligence describes any system where a computer mimics human intelligence in some way. This includes decision-making systems that inform subsequent decisions. • Driverless cars now exist, but it might be unclear whether the programmer would be to blame in the event of a crash. • Computers can read CVs to filter out certain types of jobs. This might be people from particular postcode areas or ethnic groups. An unscrupulous developer decides upon this.

INSPECTION COPY

COPYRIGHT
PROTECTED



Legal Issues

Data Protection Act (2018)

This law applies to personal data of living individuals. If an organisation stores personal data of living individuals, that data must be...

- processed fairly and lawfully
- adequate, relevant and not excessive
- not kept for longer than necessary
- held for specified purpose
- kept up to date, accurate
- kept secure

A **data subject** (a person about whom data is stored) has rights under this law:

- They should be informed about how their data is used
- They should be able to access their own personal data
- They have the right to have inaccurate data corrected
- They can have data erased, and processing stopped or restricted

Computer Misuse Act (1990)

This law makes accessing a criminal offence. The following activities are recognised as criminal:

- Accessing material on a computer that you are not authorised to access (for example, accessing a system using someone else's credentials).
- Modifying material on a computer that you are not authorised to modify. If you are allowed to access the data, you are not allowed to modify it.

Copyright, Designs and Patents Act (1988)

This law protects **intellectual property**, meaning it is a criminal offence to copy intellectual property without the permission of the owner of the **intellectual property rights**. Different types of intellectual property are protected:

- **Copyright** applies to anything that can be written (such as web pages, books, music, and images). Once something has been written, copyright exists immediately and applies for it.
- A registered **design** (as applicable to computer science) would apply to logos and icons. If a logo name suggests, such images need to be registered, and they need to have been registered.
- **Patents** can be used to protect inventions. This would apply to a piece of software that implements a new method of printing, but not to program code.

Software Licences



Software licence – a document that forms a contract between the licensor (the developer) and the end user. It spells out issues such as how a piece of software can be used and whether it can be copied.

You need to be aware of two forms of software licence:



Open source – the code is freely available to be viewed and edited. So, software developers can use the code to customise the program to their own requirements. In this way, lots of people can work collaboratively on a piece of software, improving it as they go along.



Proprietary – essentially the opposite of open source. The code is owned by an individual or organisation (either the developer or someone who has bought the software from the developer). People who want to use the software usually need to pay for it, and the terms of use are spelled out in a licence (see above), which do not normally include viewing or modifying the code.

COPYRIGHT
PROTECTED



Cultural Issues



Culture – a broad term that essentially means 'how we live'. How we what we value as a society are all considered cultural issues. This are single 'we'; different groups of people have different cultural values.

Questions about cultural issues could cover any one of a vast number of areas. is to try to examine an issue from multiple points of view, as well as examining negative aspects.

We'll use the example of an increased prevalence of technology in the classroom teachers and students, both of whom may experience positive and negative effects in the same way as each other:

	Teacher	
Positive	Learning progress can be tracked, so a full picture of learning can be established covering a whole year, or even longer.	Technology grants learning, reducing falling behind.
Negative	More time is required in order to teach students how to engage with each new piece of technology, and that time is not always available.	Many pieces of technology that students use the without a smart from some learn

A **stakeholder** is simply a person with an interest in an issue. Stakeholders in example, include children, parents, teachers and politicians.

The same approach could be applied when examining the subject of more people home via computer technology). Here, the stakeholders would be the employer

	Employer	
Positive	Money can be saved on rent, heating and lighting, since less office space is now required.	Time can be saved commute every
Negative	Tracking the activities of employees can be difficult, particularly when it comes to offline activities.	There is no phys home. Since the are effectively a

These examples simplify the situation. There could easily be more per stakeholder group, and there can be more than just two stakeholders.

**COPYRIGHT
PROTECTED**



Environmental Issues



Environment – a broad term with several meanings relating to the physical world. Global air pollution is an environmental issue, but so is the distance between your home and where you work.

Below are some examples. These bullet points are just the beginnings of arguments and conflicting ideas against each other to see which one carries the most weight.

Health issues	
<ul style="list-style-type: none"> + Proliferation of health-tracking apps allows people to monitor exercise and calorie intake – people are better informed + Medical technology is continually advancing, allowing for better predicting and diagnosing illnesses + Sharing of health-related information across the Internet helps researchers 	<ul style="list-style-type: none"> - An increase in screen time can lead to an increase in obesity - Some people are addicted to social media, and this can affect their health - Technology may be distracting people away from work
Energy use	
<ul style="list-style-type: none"> + Computer technology can be used to reduce consumption of fossil fuels; it can turn off lights in empty offices and enable vehicles to be more fuel-efficient + Smart meters allow people to track and control their use of electricity and gas at home and at work + People can work from home more, so they commute less 	<ul style="list-style-type: none"> - Computers, tablets, and smartphones all consume energy - The manufacturing of electronic devices requires electricity - Many devices are not energy-efficient, and their production is often unseen, including the energy used in hosting cloud services
Resources	
<ul style="list-style-type: none"> + In theory at least, less paper needs to be used, so fewer trees should be cut down + Some products, such as books and music, can be delivered electronically, with no physical transport needed + One delivery driver, delivering 10 Internet-ordered products on a single delivery run, requires less fuel than 10 people each driving to a shop for one item 	<ul style="list-style-type: none"> - In reality, people still use a lot of paper, and it may not really be recycled - Computers require a lot of energy to produce, and their production, storage, and disposal can be resource-intensive - Not all obsolete electronic devices are recycled; much of it ends up in landfill

Note the combination of positive and negative factors. To say that the impact, or otherwise, of computer technology is 'good' or 'bad' is to oversimplify the issue. It is a complex issue, and acknowledging this in an answer to a high-mark exam question is the right thing to do.

Privacy Issues

As more of our lives is stored and, to varying degrees, accessible online, this raises privacy concerns.

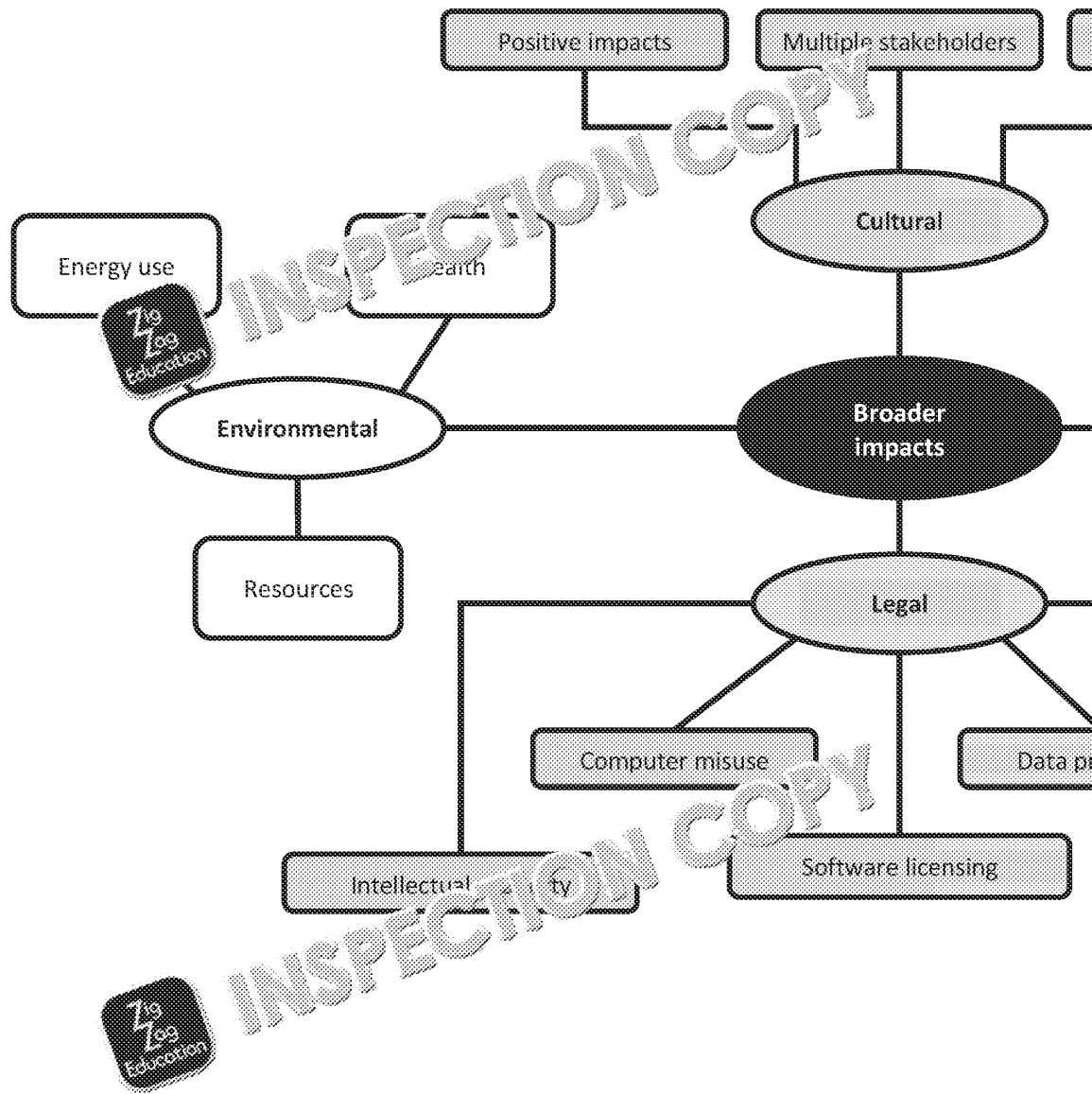
- Can I ensure my social media presence to be completely private from my employer or potential employer? What if I'm applying for a job working with vulnerable people?
- I might need a particular smartphone app in order to access some service. What permissions does it require? Is it fair that I make a choice between using the app and keeping my data private?
- The sponsored links on the web view of some email accounts are based on the content of the emails. Is privacy less of a concern if it's a program reading my emails to show me relevant ads?

These are just some examples, and you should be ready to form a quick opinion on a similar exam question.

**COPYRIGHT
PROTECTED**



Ethical, Legal, Cultural and Environmental Impacts Mind Map



INSPECTION COPY

COPYRIGHT
PROTECTED



Sample Examination-style Questions

16. 'Computers have had both a positive and negative effect on the environment' is a generalised statement. Discuss the positive and negative issues to support this statement.

INSPECTION COPY

17. According to the Data Protection Act (2018), an organisation that stores personal data must ensure that the data is kept securely. Describe **two** other requirements of the Data Protection Act (2018).

INSPECTION COPY

INSPECTION COPY

**COPYRIGHT
PROTECTED**



2.1. Algorithms



Algorithm – a series of instructions that describes how to solve a specific problem

2.1.1. Computational Thinking

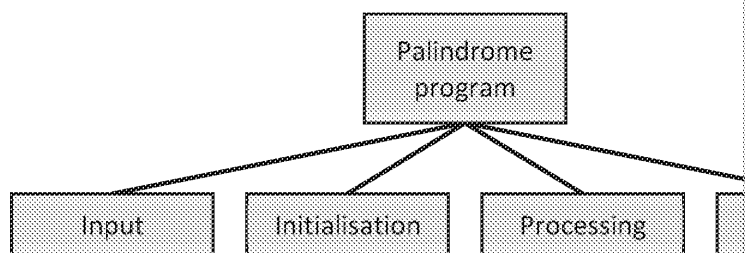
There are certain techniques that can be used to define and refine problems and referred to as 'computational thinking'.



Decomposition – breaking down a problem into smaller 'sub-problems' of advantages:

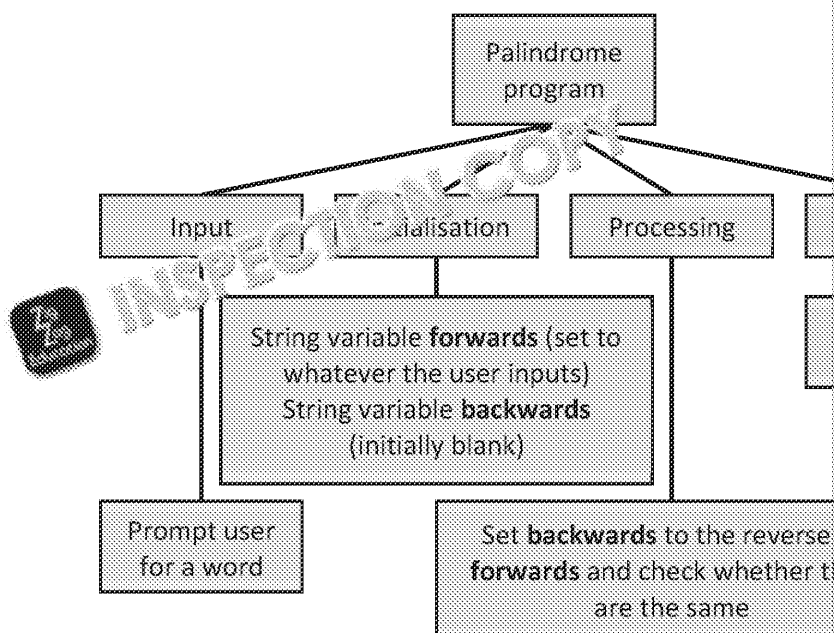
- Smaller problems are easier to solve than larger problems
- Each 'sub-problem' can be developed separately, making planning and development easier
- 'Sub-problems' are easier to distribute among a team than one large problem

Suppose you want to develop a program that identifies whether or not a word is a palindrome (reads forwards as backwards). You might decompose this problem as follows:



Input	What data enters the system, and how?
Initialisation	What variables are needed and what are their initial values?
Processing	What processes (calculations, sorts, etc.) are carried out on the input data?
Output	What data leaves the system and in what format?

You could now examine the **requirements** of each component. What exactly is needed for each?



INSPECTION COPY

**COPYRIGHT
PROTECTED**



At this point, each component can be designed, possibly using a flow chart or pseudocode.

Sometimes, multiple levels of decomposition are required. Looking at the 'process' above diagram, it might be that this particular sub-problem could be broken into several smaller sub-problems.



Abstraction – hiding the layers of complexity within a system, in order to focus on the essential parts of the problem, ignoring the details of complexity.

You might use the **min()** function in Python, which identifies the lowest value in a list. You probably have no idea exactly *how* this function operates. This is not a problem – it works, only what it does. The complexity of **min()** is hidden from you.

You can create your own routines in Python, or in any other language, using the same principle. If you're writing a program for a bank, and you've written a subroutine called **withdraw()** from an account. Other programmers could use this subroutine without understanding how it works, as they understand what it does and what it is called. Hiding complexity in this way is essential for understanding larger problems, one piece at a time.

This 'bank' example is a good way to clarify your understanding of abstraction, but you should always have multiple examples to fall back on. When else might abstraction be useful?



Algorithmic thinking – reaching the solution using a systematic approach (often involving decomposition and abstraction, as opposed to jumping straight to the solution you are usually able to).


COPYRIGHT
PROTECTED



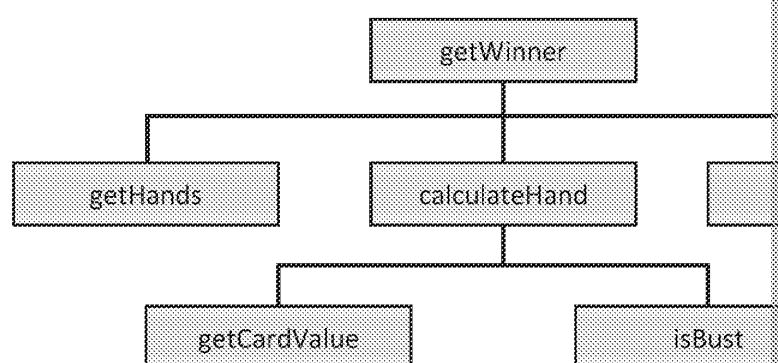
2.1.2. Designing, Creating and Refining Algorithms

When decomposing a problem, the least that can possibly be done is the identification of **inputs** and **outputs**. These parts, particularly processing, can often be decomposed further. The following table provides some examples:

Problem	Inputs	Processes
Palindrome problem	One word	Comparison of that word with a copy of it in reverse
Temperature conversion	Value in Celsius	Conversion: $(\text{Celsius} / 1.8) + 32$
Determining blackjack winner	Values of all cards	Determine nearest to (but not over) 21

 **Structure diagram** – a diagram that shows the parts into which either a problem or an algorithm can be subdivided. Structure diagrams often form the basis for devising subroutines.

The structure diagram for the example above, 'Determining blackjack winner', might look like this:



getWinner identifies the name of the winner – the person whose cards total above 21

getHands a 'hand' is a collection of one person's cards – this subroutine would return the cards for each player

calculateHand assigns a point value (ideally, as far as the player is concerned, 21 is the best hand possible)

getCardValue determines the point value of one card, e.g. a 'king' card is worth 10

isBust determines whether a hand is worth more than 21, in which case the player cannot win the game

sortHands ranks hands in order, making it easier to find the winning hand

The whole structure diagram shows a high-level view of how a winner is determined. Looking at each player's hand, whether part involves calculating their hand. Part involves calculating the value of a single card.

Although you can learn a lot from looking at a structure diagram, there's plenty that you can't learn from it:

- The order in which events happen is not clear – although it's common practice for subparts to occur left to right, that's not necessarily the case.
- The number of times a process occurs isn't clear either. Presumably, 'getCardValue' need to happen many times, but it's only included in the diagram once.
- Whether a process is optional is not clear. If everyone's hand is bust, then there's no need to sort the hands in order (since everyone's lost), so 'sortHands' might not always be needed.

**COPYRIGHT
PROTECTED**



Defining Algorithms

Commonly used methods of defining algorithms include **pseudocode**, **flow charts** and **programming language**.

The exam might contain questions on any combination of these methods. One way to practise is to practise converting between them. Try turning a flow chart into pseudocode.



Pseudocode – a cross between English and a generic-looking programming language. If pseudocode would not compile, a competent programmer could convert it into a real programming language.

A pseudocode algorithm for selling tickets might be as follows. Larger purchases get a discount per ticket:

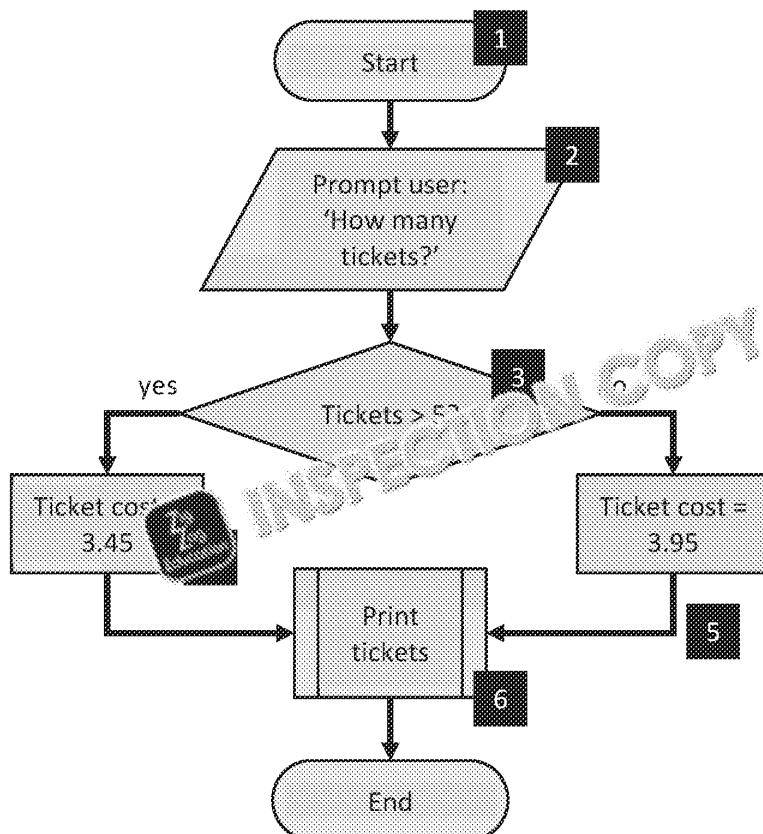
```

OUTPUT "How many tickets?"
INPUT tickets
If tickets > 5
    Display: tickets * 3.45
Else
    Display: tickets * 3.95
Endif
    
```



Flow chart – a means of defining an algorithm using shapes and arrows.

A flow chart does the same job as pseudocode in defining an algorithm, but it is more visual and easier for someone who is not a programmer. The flow chart segment below defines a simple algorithm for selling tickets, similar to the pseudocode above, but it looks very different:



1. Terminator – used to start and end a program and one that is used elsewhere.
2. Input/output – a process where data is entering or leaving the system.
3. Decision – a diamond shape where the available answers (yes/no) are written and the flow goes to the next step in this shape.
4. Process – a rectangle where an instruction or calculation is common.
5. Arrow – indicates the flow of events taking place.
6. Subprogram – a small program that needs to be called within a larger program. In this flow chart, it is used to calculate the ticket cost. This allows the program to be divided into smaller parts.

**COPYRIGHT
PROTECTED**





High-level programming language – another means of defining an algorithm in a program code. Unless you're programming in Assembly or machine code language, and examples include Python, Visual Basic, Java and C#. Also known as a programming language, designing first using flow charts or pseudocode for more complex projects.

Identifying Common Errors

Probably the best way to practise dealing with common errors is to write lots of code. You'll cause plenty of errors of your own, and you'll learn how to fix them. That's the way it's working – it's the best way. You're supposed to make mistakes (and learn from them, which is a computer science skill). Here are some of the more common errors:

- Placing a data item of the wrong type into a variable (e.g. a string into an integer)
- Writing a loop after another loop, instead of writing one inside another (or vice versa)
- Writing code in a loop that should be before or after it (or vice versa)
- Confusing 'less than' and 'greater than' symbols (less than < points left)
- Confusing 'less than' with 'less than or equal to' (and likewise with 'greater than')
- Writing a loop that never ends (unless it's never supposed to, as with traffic lights)

Trace Tables

You may be provided with an algorithm in OCR Exam Reference Language, and asked to trace it, so you need to be able to read and understand algorithms.

```
1  number = 3
2  result = 1
3  while number > 1
4      result = result * number
5      number = number - 1
6  endwhile
7  print(result)
```

Interpreting an algorithm in OCR Exam Reference Language when you do so one line at a time, as a computer would. A trace table should be used to keep track of the variables that change throughout the algorithm. The variables in this program are 'number' and 'result'.

number	result	output	Commentary
3	1		In lines '1' and '2', the variables are given these values.
3	1		In line 3, which is the start of a loop, we are told that 'number' is greater than '1'. Since 'number' is '3', the loop runs.
3	3		In line 4, 'result' is set to itself multiplied by 'number'. '1' times '3' is '3'.
2	3		Line 5 says 'number' should have 1 subtracted from it.
2	3		Line 6 marks the end of the loop, so we go back to line 3.
2	3		'number' is still greater than '1', so the loop runs again.
2	6		In line 4, 'result' is set to itself multiplied by 'number'. '3' times '2' is '6'.
1	6		Line 5 says 'number' should be reduced by '1' again on line '5'.
1	6		Line 6 marks the end of the loop, so we go back to line 3.
1	6		The loop will not run a third time because 'number' is '1' ('1' is not greater than '1'), so we jump to the first line after the loop.
1	6	6	'result' is displayed, which is currently '6'.

COPYRIGHT
PROTECTED



Looking at an algorithm as a whole can be daunting, but following it one line at a time is a good deal simpler; no individual line is particularly complicated, and errors can be spotted more easily.

As for what this algorithm does, it provides you with the **factorial** of 'number'. A factorial of a whole number below it:

Factorial 3:	$3 * 2 * 1$	6
Factorial 4:	$4 * 3 * 2 * 1$	24
Factorial 5:	$5 * 4 * 3 * 2 * 1$	120

In Python, one way to implement this algorithm would be as follows:

```
number = input("Enter a number: ")
answer = 1
while number > 1:
    answer = answer * number
    number = number - 1
print(answer)
```

The same algorithm in OCR Exam Reference Language would be as follows:

```
number = int(input("Enter a number: "))
answer = 1
while number > 1
    answer = answer * number
    number = number - 1
endwhile
print(answer)
```

You will not be asked to write program code of a specific language in the exam, but being able to do so in practice. The more fluent you are in any programming language, the better you will be at writing it (and writing) is likely to be in an exam.

INSPECTION COPY

COPYRIGHT
PROTECTED



2.1.3. Searching and Sorting Algorithms



Standard algorithm – one that would appear in many different programs so often, in so many applications, that virtually all software developers know how to program it.

Standard Searching Algorithms



Searching – determining whether a specific piece of data exists within a set. If it exists, a search algorithm will reveal its location.



Linear search – a search algorithm that begins at one end of a data structure and examines each item in turn until the required item is found, or the end of the structure is reached.



5	8	4	2	9	6	7
---	---	---	---	---	---	---

If a linear search were being used to find the number 9, the numbers 5, 8, 4, 2 are examined in order. If the number 1 were being sought, it would not be found, but each element would be examined to verify this.

If the item being searched for is found, the code returns the location within the data structure. The first element is numbered '0', the second '1', the third '2', and so on:

5	8	4	2	9	6	7
0	1	2	3	4	5	6



Binary search – a search algorithm that begins in the middle of a data structure and divides the remaining data with each pass. Binary searches are only appropriate for sorted data structures.

A binary search for the number 9 in a different collection of values:

2	4	5	8	9	11	15
---	---	---	---	---	----	----

The value in the middle is 8; the value we are searching for is larger than that and must be found somewhere to the right of 8. Consequently, the 8, and everything to its left, can be disregarded.



2	4	5	8	9	11	15
---	---	---	---	---	----	----

There are now three elements to be searched through. Again, the binary search algorithm can be used. The first two elements can be disregarded because they can only contain numbers larger than 9.

2	4	5	8	9	11	15
---	---	---	---	---	----	----

COPYRIGHT
PROTECTED



The number we were searching for has been found. If we were searching for a different number, we could conclude at this point that this number isn't present, without needing to check the rest of the data. Binary searches are quicker because, at each stage, half of the remaining data is discarded. If one million elements was to be searched using a binary search, it would take no more than 20 for any particular piece of data or to discover that the data is not contained within the array. In other words, items would only require 30 iterations.

Standard Sorting Algorithms



Sorting – putting data into order, whether that be numerical order, alphabetical order (A–Z) or descending (Z–A) – or chronological order.

Not sorted:

5	8	1	9	6	7
---	---	---	---	---	---

Sorted:



1	4	5	6	7	8	9
---	---	---	---	---	---	---

There are several different methods for sorting data items into order, and here we will look at three of the most common:

- Bubble sort
- Insertion sort
- Merge sort

There is no single 'best' sort algorithm, and it's always good, as a programmer, to be able to attempt to solve any problem.

Bubble sort

First pass

(2 4 7 5 3) ← The 2 and the 4 have been switched, as 4 is greater than 2

(2 4 7 5 3) ← The 4 and the 7 are not switched, as they are already in the correct order

(2 4 5 7 3) ← The 5 and the 7 are switched, as 7 is greater than 5

(2 4 5 3 7) ← The 3 and the 7 are switched, as 7 is greater than 3

Second pass

(2 4 5 3 7) ← The 2 and the 4 are not switched, as they are already in the correct order

(2 4 5 3 7) ← The 4 and the 5 are not switched, as they are already in the correct order

(2 4 3 5 7) ← The 3 and the 5 are switched, as 5 is greater than 3

(2 4 3 5 7) ← The 5 and the 7 are not switched, as they are already in the correct order

Third pass

(2 4 3 5 7) ← The 2 and the 4 are not switched, as they are already in the correct order

(2 3 4 5 7) ← The 4 and the 3 are switched, as 4 is greater than 3

(2 3 4 5 7) ← The 4 and the 5 are not switched, as they are already in the correct order

(2 3 4 5 7) ← The 5 and the 7 are not switched, as they are already in the correct order

At this stage the list is sorted, but one more pass would be performed because a complete pass has yielded no changes or when $n-1$ (n being the number of elements) has been taken place.

**COPYRIGHT
PROTECTED**



Insertion sort

This type of sort takes each element in the data set (not necessarily in any specific correct place within the data set. Initially, it is assumed that only the first number is in the correct order. In each pass, the number of data items assumed to be in the correct order increases until the whole data set is in the correct order. It is suitable only for smaller data sets.

- (4 2 7 5 3) ← The initial data set – the sorted section is of length 1
 (2 4 5 7 3) ← The number 2 has been added to the sorted section before the number 4
 (2 4 7 5 3) ← The number 7 has been added to the sorted section after the number 4
 (2 4 5 7 3) ← The number 5 has been added to the sorted section between the numbers 4 and 7
 (2 3 4 5 7) ← The number 3 is added between 2 and 4 and the sort is complete

Merge sort

- (4 9 5 1 3 2 7 8) ← In sorted data set
 (4) (9) (5) (1) (3) (2) (7) (8) ← Data is split into individual units
 (4 9) (5 1) (3 2) (7 8) ← The first pairing, 4 and 9, is brought together
 (4 9) (1 5) (3 2) (7 8) ← The next pairing is 5 and 1, whose positions are swapped
 (4 9) (1 5) (2 3) (7 8) ← In this way, all data are merged into sorted pairs

Next, the pairs must be merged into groupings of four. We'll look at the first two pairs.

(4 9) (1 5)

The values '4' and '1' are compared. Since these are the first within their respective pairs, the first of the four must be one of these two:

(4 9) (1 5) (1)

Next, the first *remaining* value in each pairing is compared with the other. Value

(4 9) (1 5) (1 4)

At any given point, two numbers are examined, with the next in order being added to the sorted set. The copy is always in order:

(4 9) (1 5) (1 4 5 9)

The same principle is applied to the other two pairings to leave two sorted clusters

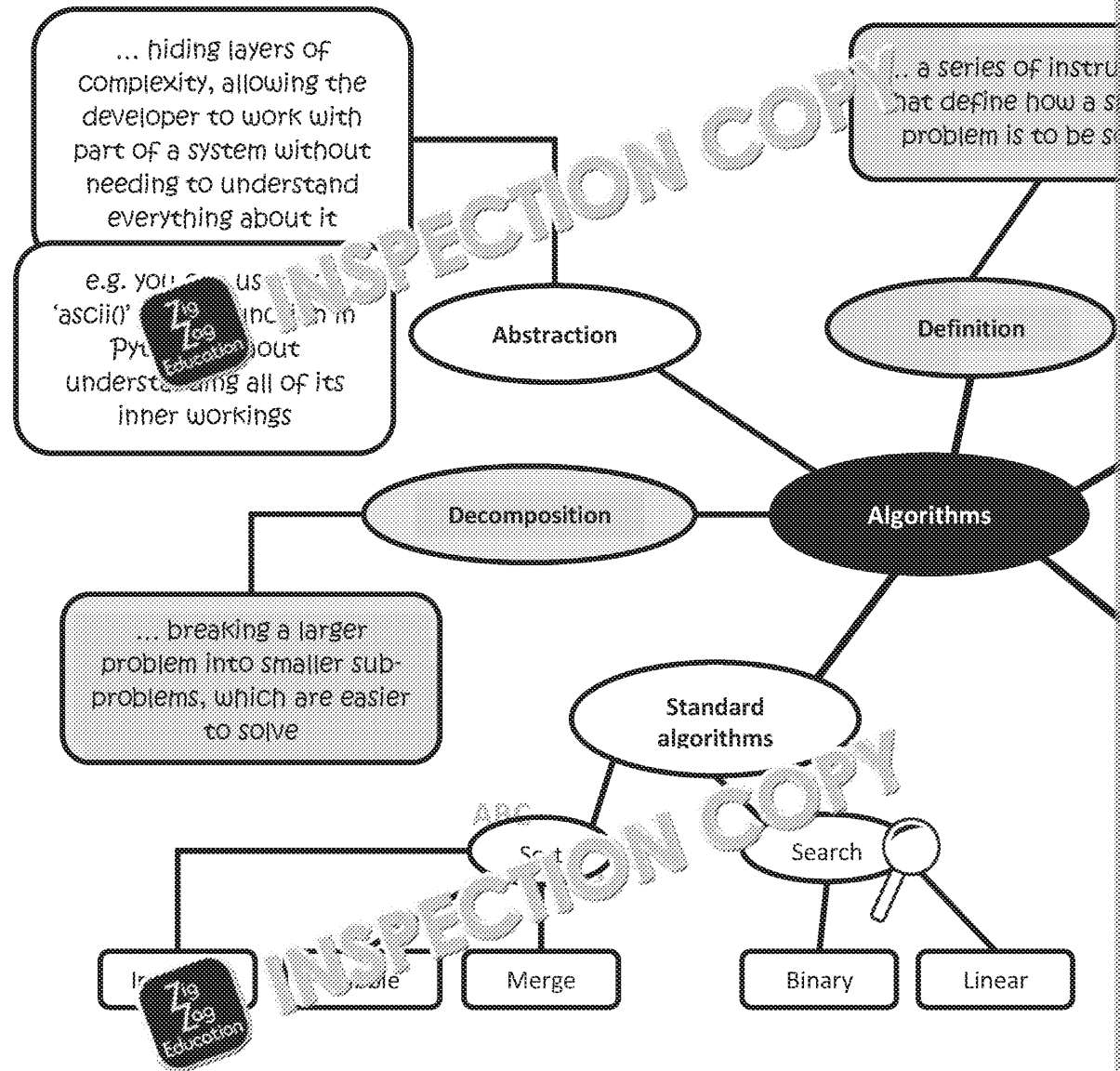
- (1 4 5 9) (2 3 7 8) ← Two sorted clusters of four data items
 (1 4 5 9) (2 3 7 8) (1) ← '1' and '2' were compared
 (1 4 5 9) (2 3 7 8) (1 2) ← '4' and '2' were compared
 (1 4 5 9) (2 3 7 8) (1 2 3) ← '4' and '3' were compared
 (1 4 5 9) (2 3 7 8) (1 2 3 4) ← '4' and '7' were compared

Eventually, this would result in a single, sorted data set, comprising eight data items. The process is repeated to sort a data set containing any number of data items.

**COPYRIGHT
PROTECTED**



Algorithms Mind Map



INSPECTION COPY

COPYRIGHT
PROTECTED



Sample Examination-style Questions

18. What is meant by the term **algorithm**?

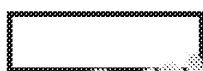
.....

.....

19. State the name of each of the following flow chart shapes.



.....



.....



.....



20. Describe the operation of a **bubble sort** algorithm.

.....

.....

.....

.....

.....

.....



INSPECTION COPY

COPYRIGHT
PROTECTED



2.2. Programming Fundamentals

2.2.1. Programming Fundamentals



Variable – a named space in memory, large enough to store a single piece of data of a data type. Although some languages, including Python, do not require a data type, each variable still has one.



Assignment – the process of putting a value into a variable. Most languages do this. In the instruction `x = 5`, the value '5' has been assigned to the variable `x`.



Constant – a named space in memory with a value that can never change while the program is running. Useful for values which will never change or VAT (which seldom changes). Some languages allow for constants, but other languages do not.



Operator – in the context of computer science, an operator performs an operation on one or more pieces of data in order to produce additional data. There are **arithmetic operators** and **Boolean operators**.



Arithmetic operator – performs a process on one or more numbers. The basic arithmetic operators are: `+` `-` `*` `/`

OCR Exam Reference Language	Python code	
<code>total = 5 + 10</code>	<code>total = 5 + 10</code>	Addition (15)
<code>result = 10 - 5</code>	<code>result = 10 - 5</code>	Subtraction (5)
<code>product = 5 * 10</code>	<code>product = 5 * 10</code>	Multiplication (50)
<code>answer = 10 / 5</code>	<code>answer = 10 / 5</code>	Division (2)
<code>outcome = 13 DIV 5</code>	<code>outcome = 13 // 5</code>	Quotient, also known as integer division – the remainder is discarded
<code>solution = 13 MOD 5</code>	<code>solution = 13 % 5</code>	Modulo (3) – the remainder of the division
<code>effect = 5 ^ 3</code>	<code>effect = 5 ** 3</code>	Exponentiation (125) – 5 to the power of 3, which can also be written as '5 times five' (125)



Comparison operator – performs comparisons between two values to determine if they are equal, or whether one is less than or greater than the other. These are found in 'if' statements and as part of loops.

INSPECTION COPY

COPYRIGHT
PROTECTED



OCR Exam Reference Language	Python code
if x > y then ... endif	if x > y:
while a < b ... endwhile	while a < b:
if q >= r then ... endif	if q >= r:
while j <= k ... endwhile	while j <= k:
if e == f then ... endif	if e == f:
while m != n ... endwhile	while m != n:



Boolean operator – a logic expression can have one of only two outcomes. A Boolean operator connects together logic expressions to produce a more complex expression. AND and OR are the most commonly used logic operators.

OCR Exam Reference Language	Python code	
if age > 15 AND age < 65 then ... endif	if age > 15 and age < 65:	Age between 15 and 65
if age < 16 OR age > 64 then ... endif	if age < 16 or age > 64:	The age is less than 16 or more than 64
if NOT(age < 16) then ... endif	if not(age < 16):	True if age is 16 or more, False otherwise

Note the difference in case between Python (and, or, not) and OCR Exam Reference Language (AND, OR, NOT)

**COPYRIGHT
PROTECTED**



Here is a piece of Python code that combines all three types of operator. It calculates the cost for adults and children entering a zoo. How much would it cost for two adults and two children?

```
adults = int(input("How many adults: "))
children = int(input("How many children: "))

if (adults + children >= 5) or (adults >= 2):
    cost = adults * 9 + children * 4.5
else:
    cost = adults * 10 + children * 5
print(cost)
```

Here is the same functionality represented using OCR-A Reference Language

```
adults = input("How many adults: ")
children = input("How many children: ")

if ((adults + children >= 5) or (adults >= 2)) then
    cost = adults * 9 + children * 4.5
else
    cost = adults * 10 + children * 5
endif
print(cost)
```

INSPECTION COPY

COPYRIGHT
PROTECTED



Programming Constructs

Sequence

Sequence means that instructions will always execute in the order in which they are written. Each instruction will be executed once and only once.

```
hourlyRate = float(input("Hourly rate: "))
hours = int(input("Hours worked: "))
print(hourlyRate * hours)
```

This program asks for the hourly rate and the number of hours worked before calculating the total pay (the two figures multiplied). At this point, the program ends. Note that there is no difference between the way in which this program is written in the Exam Reference Language in the way in which this program is written.

Selection

Selection is when one or more lines of code are chosen where multiple possibilities exist. The word "if" is a common indicator of selection, although there are other words that can be used.

PYTHON:

```
hours = int(input("Number of hours: "))
rate = float(input("Rate per hour: "))
if hours > 40:
    print("Normal Earnings:      " + str(40 * rate))
    print("Overtime Earnings:    " + str((hours - 40) * rate))
else:
    print("Normal Earnings:      " + str(hours * rate))
    print("No Overtime")
```

OCR EXAM REFERENCE LANGUAGE:

```
hours = input("Number of hours: ")
rate = input("Rate per hour: ")
if hours > 40 then
    print("Normal Earnings:      " + str(40 * rate))
    print("Overtime Earnings:    " + str((hours - 40) * rate))
else
    print("Normal Earnings:      " + str(hours * rate))
    print("No Overtime")
endif
```

The first two lines will always run; the program asks the user for the number of hours and the rate, storing each value in a separate variable. Then a decision is made. If 'hours' is greater than 40, the first two indented lines will run. Otherwise, the last two indented lines will run. There will be no output when all four would be executed.

INSPECTION COPY

COPYRIGHT
PROTECTED



Condition-controlled iteration

Iteration means 'looping'. Code that is iterative might be executed multiple times, but the code is only written once.

PYTHON

```
looping = True
while looping == True:
    hourlyRate = float(input("Hourly rate: "))
    hours = int(input("Hours worked: "))
    print(hourlyRate * hours)
    userInput = input("Another? (y/n)")
    if userInput == "n":
        looping = False
```

OCR EXAM REFERENCE LANGUAGE

```
looping = True
while looping == True
    hourlyRate = input("Hourly rate: ")
    hours = input("Hours worked: ")
    print(hourlyRate * hours)
    userInput = input("Another? (y/n)")
    if userInput == "n" then
        looping = False
    endif
endwhile
```

The second line specifies the condition that will make the rest of the code loop until the 'looping' variable is set to false.

Count-controlled iteration

The above code loops until the user enters the letter 'n'. It is impossible to know how many times the loop will run. Sometimes, code is required that runs a predetermined number of times.

PYTHON

```
for x in range(1, 13):
    print(x * 2)
```

OCR EXAM REFERENCE LANGUAGE

```
for x = 1 to 12
    print(x * 2)
next x
```

This code will run 12 times, with the variable 'x' taking the values 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12. The purpose of this code is to calculate the two times table.

INSPECTION COPY

COPYRIGHT
PROTECTED



2.2.2. Data Types

Data type	Description	
Boolean	Can be either true or false – requires only 1 bit of storage. Examples of Boolean data might include whether or not a student has passed an exam, or whether or not an iteration (loop) is going to run again.	a
Character	A single letter, number, punctuation mark, space, etc. – requires 1 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.	b
String	Multiple characters as part of the same data item. These characters can be any combination of numbers, letters, symbols and even variable characters.	c
Integer	Whole numbers – the more storage space that 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.	d
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. In Visual Basic, the data type for 'real' numbers is often 'double'.	e



Casting – converting a piece of data to a specific data type. For example, a string, and the program might convert it to an integer, in order to allow it to be used in a calculation.

Casting to integer	<pre>a = "123 "</pre> <pre>b = int(a)</pre> <p>The variable 'a' contains a string made up of numeric characters. The program converts this string to the integer 123, storing that integer in the variable 'b'.</p>
Casting to real	<pre>c = "123.456 "</pre> <pre>d = float(c)</pre> <p>The variable 'c' now contains a string that looks like a decimal number. The program converts it to a real data type, storing it in 'd'.</p>
Casting to string	<pre>e = 789</pre> <pre>f = str(e)</pre> <p>The variable 'e' contains an integer, which is converted to a string and stored in 'f'.</p>
Casting to Boolean	<pre>g = "True"</pre> <pre>h = bool(g)</pre> <p>The variable 'g' contains the string 'True', and this is cast as a Boolean, which is then stored in 'h'.</p>
Note that the code for Python and OCR Exam Reference Language is identical.	

**COPYRIGHT
PROTECTED**



2.2.3. Additional Programming Techniques

Basic String Manipulation

Python / OCR Exam Reference Language	Explanation
<pre>first = "Richard" last = "Lee" fullName = first + " " + last (same in both languages)</pre>	The third line uses the '+' operator (join them together), along with a space string is then stored in its own variable. This would be useful if a person's first and last names are entered at different locations or entered on different lines.
<p>Python</p> <pre>first = "Richard" print(len(first))</pre> <p>OCR Exam Reference Language</p> <pre>first = "Richard" print(first.length)</pre>	Displays the length (the number of characters) of the string. This would output the number 8.
<p>Python</p> <pre>fullName = "Richard Lee" print(fullName[0:7])</pre> <p>OCR Exam Reference Language</p> <pre>fullName = "Richard Lee" print(fullName.substring(0,7))</pre>	Displays a substring (part of a string) from the start (0) and lasting for seven characters (7).

Basic File-handling Operations (Python)

Python	Explanation
<pre>file = open("file.txt", "w") file.write("Hello World!") file.close()</pre>	Opens a file to write (w) and then writes the string "Hello World!". It then closes the file. If the file does not exist, it will be created by the 'w' mode.
<pre>file = open("file.txt", "a") file.write("Hello again!") file.close()</pre>	This has the same effect as the 'w' mode, but it means append , so the 'Hello again!' is added to the end of the file's current content, instead of replacing it.
<pre>file = open("file.txt", "r") print(file.read()) file.close()</pre>	The 'r' means 'read'. These instructions will display the entire contents of the file.
<pre>file = open("file.txt", "r") print(file.read(4)) file.close()</pre>	This code displays only the first four characters of the file.


**COPYRIGHT
PROTECTED**



Basic File-handling Operations (OCR Exam Reference Language)

OCR Exam Reference Language	Explanation
<pre>newFile("myFile.txt") file = open("myFile.txt")</pre>	Creates a new empty file in the first line and opens it in the second line.
<pre>file = open("file.txt") file.writeLine("Hello World!") file.close()</pre>	Opens a file and adds the text "Hello World!". The file is only closed at the end of the file but saves it.
<pre>file = open("file.txt") while NOT file.endOfFile() print(file.readLine()) endwhile file.close()</pre>	These instructions open the file and read it line by line until the end of the file is reached.

Records



Records are a data structure that can accept multiple data items that do not have a specific data type. As far as Python is concerned, there is no difference between lists and records, so they are managed in the same way. One record stores a student's name, year group and average test score. The next record would store the details of another student.

Python code	Explanation
<pre>student1 = ["Bob", 8, 89.2]</pre>	Creates a record called 'student1'; which Python supports, but not all other languages.
<pre>student1[1] = 9</pre>	'Bob' has been moved from Year 8 to Year 9.
<pre>print(student1[2])</pre>	Displays 89.2 – Bob's average test score.

COPYRIGHT
PROTECTED



Structured Query Language (SQL)



SQL – Structure Query Language is used to manipulate the content of a database. It can be used to create and remove tables.

SQL can be entered directly into a database console or embedded in another language. A Python program could execute a line of SQL to add a new record to a table.

All SQL commands in this chapter relate to the following database table, which is called *student*.

StudentID	LastName	MathsScore
S123	Evans	75
T456	Green	85

The fields *StudentID* and *LastName* are of a text format, while *MathsScore* is a number between 0 and 100 (SQL ignores leading zeros and treats them in the same way).

SQL statement	Explanation
<code>SELECT * FROM student</code>	Reads everything from the table. The * symbol means 'all'. <i>student</i> is the name of the table.
<code>SELECT MathsScore FROM student</code>	Reads the named field, <i>MathsScore</i> , from the table, ignoring other fields. This returns the average score, where name is irrelevant.
<code>SELECT LastName FROM student WHERE MathsScore > 80</code>	Reads the last names from the table where the score is above 80.
<code>SELECT MathsScore FROM student WHERE StudentID = "T456"</code>	Reads the maths score of the student with ID 'T456'. Notice how text (T) is in quotes but numbers (80, in the previous example) are not.

Arrays

An array can have any number of dimensions, from one upwards. At this level, you will only use one-dimensional and two-dimensional arrays.



One-dimensional array – a data structure for storing multiple data items. Think of a one-dimensional array as a row of variables. Instead of each variable having a separate name, the whole array has a single name, and each **element** in the array has an index number below it. A one-dimensional array might store a pupil's most recent sports scores.

	0	1	2	3	4	5	6	7

The first element is always numbered '0'.

**COPYRIGHT
PROTECTED**



Python is quite unique among programming languages in that it does not use arrays. A **list** is similar to an array in that it can store multiple data items under one name, but it will extend itself as more space is required to store more data. Arrays can only store a fixed number of items.

Code	Expl
Python <code>myArray = [4,6,1,2,9,0]</code> OCR Exam Reference Language <code>array myArray = [4,6,1,2,9,0]</code>	Creates an array (OCR) or list (Python) called 'my' with six integers.
Both languages <code>myArray[0] = 5</code>	Places the number '5' in the first position of the array or list.
Both languages <code>print(myArray[2])</code>	Displays the third element in the array or list, in this case, '1'.

Two-dimensional array – a data structure for storing multiple data items, much like a one-dimensional array, but can be considered as a grid rather than a list.



0, 0	0, 1	0, 2	0, 3
1, 0	1, 1	1, 2	1, 3
2, 0	2, 1	2, 2	2, 3

Elements are referred to, as seen here, with two numbers, much like a grid. If the grid were larger, it could be used as a grid for a computerised game of noughts and crosses.

Code	Expl				
Python <code>myArray2 = [["a","b"],["c","d"]]</code> OCR Exam Reference Language <code>array myArray2[2,2]</code> <code>myArray2[0,0] = "a"</code> <code>myArray2[0,1] = "b"</code> <code>myArray2[1,0] = "c"</code> <code>myArray2[1,1] = "d"</code>	Creates a two-dimensional array (OCR) or list (in Python) called 'myArray2' containing two rows and two columns of data. <table border="1" data-bbox="730 1167 943 1245"> <tr> <td>a</td><td>b</td></tr> <tr> <td>c</td><td>d</td></tr> </table> It might help to visualise it in a grid, but the computer does not store arrays in a grid.	a	b	c	d
a	b				
c	d				
Python <code>myArray2[1][0] = "z"</code> OCR Exam Reference Language <code>myArray2[1,0] = "z"</code>	This would replace 'c' with 'z'.				
Python <code>print(myArray2[0][1])</code> OCR Exam Reference Language <code>print(myArray2[0,1])</code>	Displays the requested letter 'b'.				

COPYRIGHT
PROTECTED



Subprograms

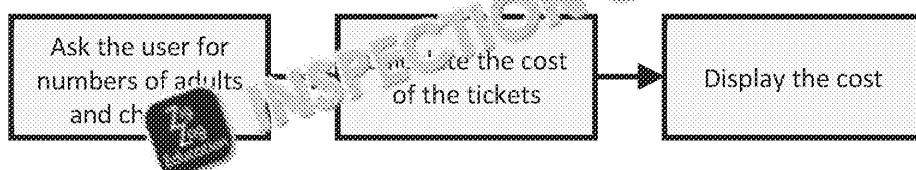


Subprogram – a small subsection of the whole program that performs a specific task. If a program were to be broken into subprograms, one subprogram might return a book, one might add a new member, etc.

There are benefits to using subprograms to write a program:

- Each subprogram can be given to a different programmer, so working as a team.
- Subprograms can each be separately tested, without waiting for the whole program to be finished.
- Subprograms that are commonly used, such as to sort a data set, can be reused, saving time.

We could break a ticketing program into the following subroutines:



In Python, the subprograms would be declared like this:

```
def promptUser():
def calculateTotal():
def displayTotal(total):
```

The 'def' keyword means we are *defining* a subprogram. The name of the first subprogram is 'promptUser'. We do with variables, subprograms should be given names that reveal their purpose.



Parameter – a piece of data that is passed into a subprogram in order to perform a specific job. In the above example, 'total' is a parameter that will be passed to 'displayTotal'. The other subprograms have no parameters.

The code within a subprogram will only happen if that subprogram is called.



Calling – a process whereby an instruction in one part of the code tells another part of the code to run. If you have a subprogram called 'promptUser', the code within it will never happen without that subprogram being called.

```
promptUser()
total = calculateTotal()
displayTotal(total)
```

**COPYRIGHT
PROTECTED**



These three instructions tell the subprograms to happen in a particular order. Call the three subprograms, so that the finished listing looks like this. Notice how spacing

```

1  def promptUser():
2      global adults, children
3      adults = int(input("How many adults: "))
4      children = int(input("How many children: "))
5
6  def calculateTotal():
7      print("adults: " + str(adults))
8      if ((adults + children >= 5) or (adults >= 2)):
9          total = (adults * 9) + (children * 4.5)
10     else:
11         total = (adults * 10) + (children * 5)
12     print(total)
13     return total
14
15 def displayTotal(total):
16     print("Total cost: " + str(total))
17
18
19 promptUser()
20 total = calculateTotal()
21 displayTotal(total)

```

Interpreting an algorithm works best when you do so one line at a time, as a **trace table** can and should be used to keep track of variables that change through an algorithm. The variables in this program are called 'number' and 'result'.

Two types of subprogram are functions and procedures.



Function – a subprogram that returns a value to the line of code from

```

1  def addTwoNumbers(a, b):
2      c = a + b
3      return c
4
5  print(addTwoNumbers(3, 4))

```

The function is declared on line 1, and called from line 5. The values 3 and 4 are then added together with the result returned. The result, 7, is what is printed on



Procedure – a subprogram that does not return a value, unlike a function

```

1  def addTwoNumbers(a, b):
2      c = a + b
3      print(c)
4
5  addTwoNumbers(3, 4)

```

**COPYRIGHT
PROTECTED**



In OCR Exam Reference Language, procedures and functions are declared differently to other programming languages.

To declare a procedure:

```
procedure printError()
    print("Error")
endprocedure
```

To declare a function:

```
function getZero()
    return 0
endfunction
```

Random Number Generation

Random numbers can be useful in many situations, including controlling enemy behaviour and generating events in simulations. In Python, in common with most other languages, you can generate random numbers.

Python code	Explanation
<code>from numpy import random</code>	This line must be placed at the top of the program before any random number generation. Without this line, the following lines will work, as 'random' would be imported from the standard library.
<code>x = random.rand()</code>	Places into the variable 'x' a random real number between 0 and 1.
<code>x = random.randint(10)</code>	Places into the variable 'x' a random integer between 0 and 10. Whatever the value in the brackets, the random number is up to, but not including, this value.
<code>x = random.randint(5, 10)</code>	Places into the variable 'x' a random integer between 5 and 10. When two parameters are used, the random number is between the first parameter and the second parameter, but the second parameter is not included. The random number might be 5 (and might be 6, 7, 8, 9).

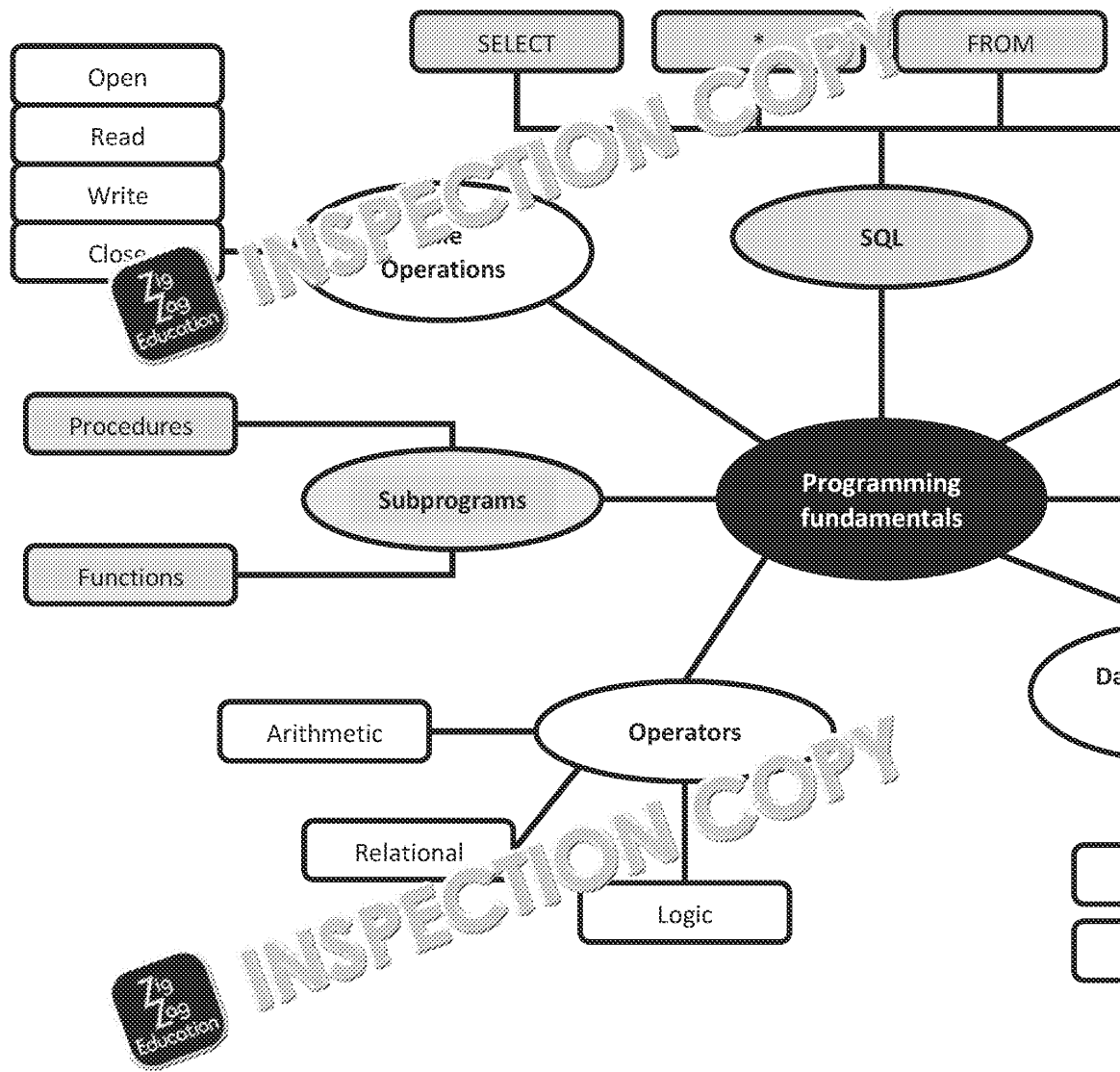
In OCR Exam Reference Language, the syntax is a little different:

OCR Exam Reference Language	Explanation
<code>x = random(1,10)</code>	Creates a random integer between 1 and 10. Both 1 and 10 are included, so 1 is a possible value, so is 10, so is every integer in between. The random number is stored in the variable 'x'.
<code>y = random(-2.5, 3.0)</code>	Creates a random real number between -2.5 and 3.0. Both -2.5 and 3.0 are included, so -2.5 is a possible value, so is 3.0, so is every real number in between. The random number is stored in the variable 'y'.

COPYRIGHT
PROTECTED



Programming Fundamentals Mind Map



INSPECTION COPY

COPYRIGHT
PROTECTED



Sample Examination-style Questions

21. This question is based on the following program written using OCR Exam Res

```
total = 0                //keeps a running total
count = 0                //logs how many are entered
mean = 0.0              //to store the mean

in = input("Enter an integer")
while in > -1
    total = total + in
    count = count + 1
    in = input("Enter an integer")
endwhile
mean = total / count
output mean
```

a. Give an example of each of the following from the code:

i. Variable

.....

ii. Comment

.....

iii. Arithmetic operator

.....

iv. Comparison operator

.....

v. Iteration keyword

.....

b. When the program is running, what would the user enter in order to stop all of their numbers?

.....

.....

INSPECTION COPY

**COPYRIGHT
PROTECTED**



22. A bank stores several pieces of data about each bank account. Identify the type of each of the following pieces of data. Some data types are used more than once.

Place **one** tick in each row.

Data	Integer	Real	Boolean
Money in account			
Account holder's name			
Number of whole years the account has been active			
Account holder's gender			
Whether or not an overdraft is permitted			
A single-letter code that identifies the account type			

23. An array called `data` contains 10 numbers, each being an integer between 0 and 10. Write pseudocode or a programming language with which you are familiar, to find out how many numbers in the array are higher than five.

INSPECTION COPY

COPYRIGHT
PROTECTED



2.3. Producing Robust Programs



Robust – normally, this means 'of sturdy construction'. In terms of programs, it means that the software will continue to function if it encounters errors, whether those errors are caused by the user (entering a letter where a number is needed) or the computer (a required resource is unavailable).

2.3.1. Defensive Design

Defensive design entails the following:

- Misuse
- Authentication
- Validation
- Maintainability



Misuse – using a program in some way other than how it was intended. This usually means someone trying to break your program (although this also means that some users may interact with your program in unexpected ways).

Problem	Solution
The user does not click on buttons in the order that they should.	Consider hiding or disabling buttons until the correct time for them to be used.
The user enters numbers with currency symbols and commas, which requires additional coding.	Instead of allowing the user to enter numbers, provide them with buttons for the numbers, like the Windows calculator.
The user clicks repeatedly on a button, which ultimately causes a process to be performed more times than is needed.	Provide feedback, such as a visual indicator, to an egg-timer cursor when the button is clicked on.



Authentication – the software process of ensuring that the person accessing the system is who is *supposed* to access that system. The following might be used:

- Usernames and passwords
- Memorable information – prompting for something only the real user would know, such as a favourite place or the name of a first pet
- Checking that the user is using their usual computer, by logging the IP address

Authentication techniques are used throughout the cybersecurity world.

INSPECTION COPY

COPYRIGHT
PROTECTED



Input Validation



Validation – ensuring that data entered into the computer is reasonable. For example, checking that a person's date of birth isn't in the future. Validation does not mean the data is correct.

Different types of data can be input, so a range of validation checks exist to suit:

- Range check** ensures that data is within a specified range, e.g. ensuring a person's age is between 0 and 30 seconds, or ensuring a person's birthdate is within a certain range.
- Type check** ensures that the correct data type has been entered (e.g. ensuring a string contains a valid number of characters).
- Length check** ensures that a string contains a valid number of characters (e.g. ensuring a National Insurance number is 11 characters long).
- Lookup check** checks that what has been entered exists on a list, such as a list of countries. For shorter lists, the user can select the item from a dropdown menu.
- Presence check** simply checks that the user has entered something.

This piece of Python code shows how a range check would be implemented. The while loop repeats until a number of '1' or above is entered:

```
isValid = False
while isValid == False:
    bedrooms = int(input("How many bedrooms: "))
    if bedrooms >= 1:
        isValid = True
    else:
        print("Please enter a positive number")
```

The following piece of OCR Exam Reference Language performs an identical role:

```
isValid = False
while isValid == False
    bedrooms = input("How many bedrooms: ")
    if bedrooms >= 1 then
        isValid = True
    else
        print("Please enter a positive number")
    endif
endwhile
```

Validation checks vary quite a lot, but this type of code structure does not. A loop repeats indefinitely, only ending when valid data has been entered. The only sign of this would be to the 'if' line.

**COPYRIGHT
PROTECTED**



Maintainability



Maintenance – changing a program after it has been produced, usually for the following reasons:

- Errors were discovered in the original code
- The requirements have changed
- Additional features could make the program even better

Maintainability is a measure of how quickly, easily and securely maintenance can be carried out.

There are several techniques that can be used to enhance the maintainability of a program.

```
#Calculate income tax at 20%
def calculateTax(salary):
    tax = salary * 0.2
    return tax

#Calculate National Insurance at 11%
def calculateNI(salary):
    ni = salary * 0.11
    return ni

#Print payslip
salary = 30000.0
deductions = calculateTax(salary) + calculateNI(salary)
print("Gross:", salary)
print("Net: ", (salary-deductions))
```

1. **Use of subprograms.** Decomposing a program into smaller subprograms not only makes it easier to write and code, it also makes it easier to maintain the program by rewriting a single part without having to rewrite the whole thing.
2. **Naming conventions.** Variable names such as 'tax' and 'salary' are intuitive and clear. The two functions 'calculateTax' and 'calculateNI' both have names that clearly indicate their purposes are also clear.
3. **Indentation.** In Python, indentation is essential in order for a program to work. In the case in other languages. Indentation in other languages is optional, but it makes the code easier to read, as do the gaps between different sections of code.
4. **Commenting.** You might form part of a programming team, or you might come back to a program after you started writing it. The comments, ignored by the computer and written for humans, are descriptions to help your colleagues and your future self to understand the program. In Python, the '#' identifies a comment, but other languages have their own ways as well.

Calculating income tax and National Insurance is a little more involved than this example, but the code that does these things will still use subprograms, sound naming conventions, indentation and commenting.

**COPYRIGHT
PROTECTED**



2.3.2. Testing

The purpose of testing is to ensure that a program performs as expected, and to ensure that anyone who has programmed understands how easy it is to make a mistake while programming. Testing is required to address such mistakes. Testing falls into two categories:

- **Iterative testing**
- **Terminal testing**, also known as **final testing**



Iterative testing – testing that takes place alongside development. If you write a program, incrementally testing it and changing code as you go, you're doing iterative testing.



Terminal testing – requires the program to be finished, at which point a type of testing looks at whether the component parts, which were iteratively tested, are put together correctly.

A sound testing strategy would involve combining elements of each of these.

Identifying Errors

Even the most experienced programmers write code with errors. So many errors are divided into different types:



Syntax error – the grammar or rules of the language have not been followed. For example, such as 'for' has been misspelled, or perhaps indentation has not been correct. A syntax error causes a program to stop working or perhaps not even start.



Logic error – a piece of code is written incorrectly in a way that does not work as intended. The programmer might have added when they meant to subtract, or used 100 instead of 10 for a percentage calculation.

Error messages are helpful if you can understand them, but not all of them are clear.

```
sh-4.3$ python3 main.py
Traceback (most recent call last):
  File "main.py", line 3, in <module>
    prit("Hello World!");
NameError: name 'prit' is not defined
sh-4.3$
```

This has appeared as a result of 'print' being misspelled as 'prit', although this error message is as user-friendly as it could be. There are many potential error messages to see here's how you can better prepare yourself for the next one:

1. Deliberately cause an error. You might remove a bracket, indent something incorrectly, or anything else you can think of.
2. Type out the error message, alongside a description of what caused it.
3. Repeat this for as many different errors as you can think of.

Now, the next time you see an error message that you *didn't* cause deliberately and you'll hopefully see a potential cause for that error. If you don't see one, you'll have to solve the problem the hard way – by going through the code line by line. At this point, the error message, and its explanation, can be added to your list.

Only ask for help *after* you've tried to solve it yourself.

**COPYRIGHT
PROTECTED**



Selecting Suitable Test Data

Choice of test data is important. Suppose you have written a program for an estate agent to calculate the number of bedrooms in a house, which must be an integer between 1 and 15:

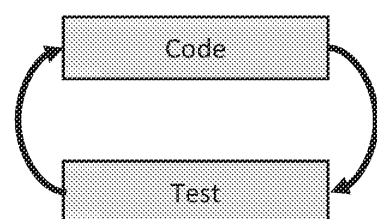
Type of test data	Description
Typical	Data that is valid and that represents how the program would be used.
Boundary	Data that is just barely valid, to check that the extreme ranges of normal input work correctly.
Erroneous	Data that should not be accepted by the system. This is included to test whether a program's validation and error messages work correctly.

If you have written code that is supposed to sort eight positive integers into ascending order, then your test data should have a specific purpose:

Test data	Explanation
1 2 3 4 5 6 7 8	The data is already sorted, so the program should be checked).
8 7 6 5 4 3 2 1	The values are in descending order, which requires as far from being in ascending order as possible.
2 6 4 3 8 7 5	How does the program react to having a number in the middle?
-1 2 3 4 5 6 7 8	How does the program react to having a negative number?
1 2 3 4.2 5 6 7 8	How does the program react to one piece of data not being an integer?
5 4 4 2 9 8 8 5	Does the sort still work correctly if there are duplicate values?

Refining Algorithms

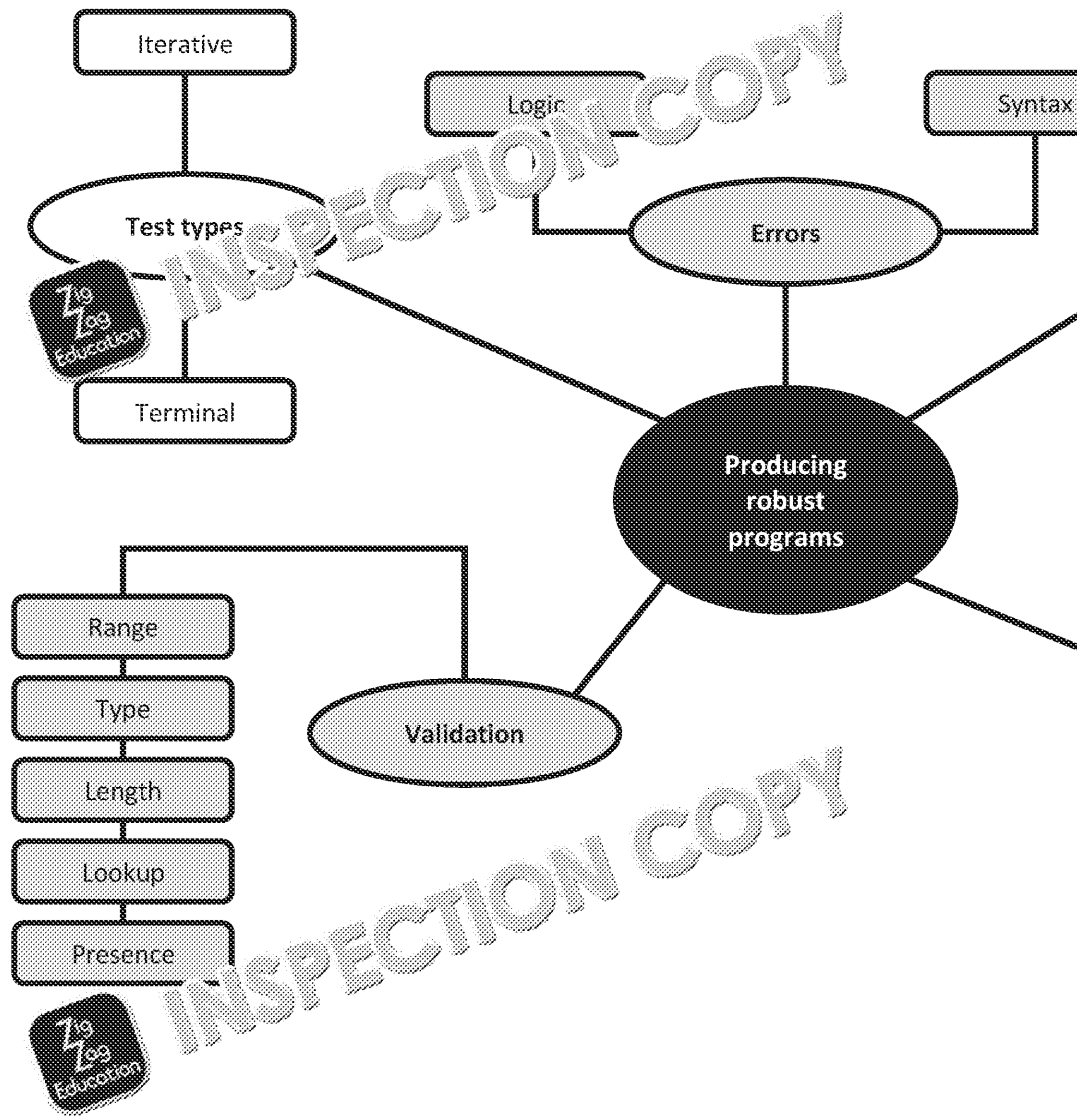
Testing is supposed to uncover **bugs** (problems with code, evident due to errors) and then you should resolve it by altering your code, then retesting.



COPYRIGHT
PROTECTED



Producing Robust Programs Mind Map



INSPECTION COPY

COPYRIGHT
PROTECTED



Sample Examination-style Questions

24. This question is about the following OCR Exam Reference Language algo

```
array data = [4, 2, 7, 9, 8]
a = 0
b = 9
c = False
while a < data.length AND c == False
    if b = data[a]
        c = True
    endif
    a = a + 1
endwhile
print(c)
```

- a. Complete the following trace table. You may not need to use all rows.

a	data[a]	b	
0	4	9	

- b. What is the purpose of this algorithm?

.....

.....

.....

.....

25. Describe, with examples, each of the following error types:

Type of error	Description	
Logic		
Syntax		

INSPECTION COPY

COPYRIGHT
PROTECTED






2.4. Boolean Logic



Boolean logic – in computer science, this refers to **Boolean** outputs (0 or 1) produced according to combinations of Boolean inputs. These are common since many computer components have two states. Circuits can be designed to hold a charge or not hold a charge, magnetic fragments can be aligned or not.

2.4.1. Boolean Logic

Logic expression	Gate	Truth table															
AND all inputs must be '1' for the output to be '1'		<table><tr><th>A</th><th>B</th><th>Output</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Output	0	0	0	0	1	0	1	0	0	1	1	1
A	B	Output															
0	0	0															
0	1	0															
1	0	0															
1	1	1															
OR if either input, or both inputs, is '1', the output is '1'		<table><tr><th>A</th><th>B</th><th>Output</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Output	0	0	0	0	1	1	1	0	1	1	1	1
A	B	Output															
0	0	0															
0	1	1															
1	0	1															
1	1	1															
NOT the output is simply the opposite of the input		<table><tr><th>A</th><th>Output</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	Output	0	1	1	0									
A	Output																
0	1																
1	0																

A truth table can be more complex. If we consider a greenhouse, we might want a sprinkler (water = 1) only if the soil is dry (moisture = 0), it is daytime (light = 1) and the door is closed (door = 0). In all other events, the sprinkler is turned off (water = 0).

Inputs			Output
Moisture	Light	Door	Water
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

INSPECTION COPY

COPYRIGHT
PROTECTED



Logic can be found throughout program code, specifically within 'if' and 'while' statements.

```
IF (((age < 16) OR (age < 19 AND student = TRUE)) AND
```

There are three variables here – 'age', 'student' and 'income'. Let's give them some values.

```
age = 16
student = TRUE
income = 21000
```

How would we determine whether this expression evaluates to 'true' or 'false'?

1. Replace logical expressions with TRUE or FALSE.

```
IF (((FALSE) OR (TRUE AND TRUE)) AND (FALSE))
```

2. The AND in the middle is more nested in brackets than any other, so that can be done first. $(1 + 1 = 1)$

```
IF (((FALSE) OR (TRUE)) AND (FALSE))
```

3. Moving outward, $(FALSE) OR (TRUE) = TRUE$ $(1 + 0 = 1)$

```
IF ((TRUE) AND (FALSE))
```

4. Next, $(TRUE) AND (FALSE) = FALSE$ $(1.0 = 0)$

```
IF (FALSE)
```

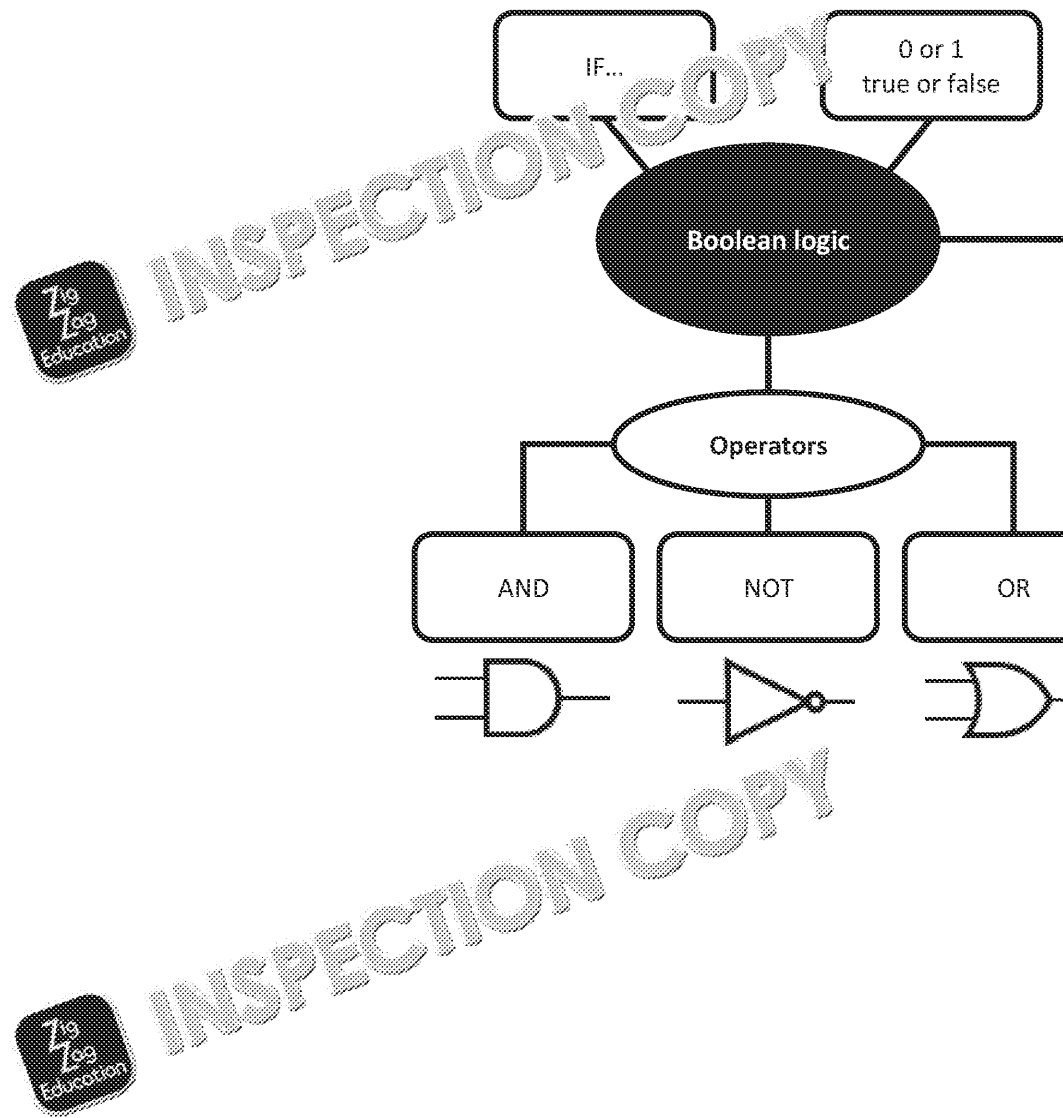
Finally, you're left with a single TRUE or FALSE; FALSE in this case. This would mean the code block following the IF statement would not be executed. If there is an ELSE line later in the code, you would jump to that. If not, you would skip to the end of the IF structure and continue. Had this evaluated to TRUE, whatever that might be, would be executed.

INSPECTION COPY

COPYRIGHT
PROTECTED



Boolean Logic Mind Map



INSPECTION COPY

COPYRIGHT
PROTECTED



Sample Examination-style Questions

26. A system that automatically waters a lawn will turn on sprinklers if **both** of the following conditions are met:
- The temperature is above 20 degrees Celsius
 - The lawn was last watered more than five days ago
- a. State the name of the logic operator that would apply between these conditions.
-
- b. Draw the truth table for this logic operation. You can assume two input variables:
- Q is set to 1 if the temperature is above 20°C, otherwise it is set to 0
 - P is set to 1 if the lawn was last watered more than five days ago, otherwise it is set to 0



INSPECTION COPY

INSPECTION COPY

COPYRIGHT
PROTECTED



27. A credit card company issues two types of credit card: a *gold card* and a *standard card*. If a person applies for a card, the company will do one of three things:
- offer a gold card
 - offer a standard card
 - reject the application

The decision depends on the person's income and whether or not they are a homeowner. The logic used is shown below in OCR Exam Reference Language:

```
income = int(input("Enter income"))
homeowner = bool(input("Homeowner?"))
if ((income > 25,000) AND (homeowner == True))
    print("Gold Card")
elseif (((income > 15,000) AND (homeowner == True))
    print("Standard Card")
else
    print("Rejected")
endif
```

What will be the output for each of the following?

- a. A homeowner with an income of £13,000.

.....

- b. A homeowner with an income of £30,000.

.....

- c. A non-homeowner with an income of £22,000.

.....

COPYRIGHT
PROTECTED



2.5. Programming Languages and Development Environment

2.5.1. Languages

Levels of Programming Language

	High level	Low level
What is it?	Understandable to humans because it resembles natural language a little, but is often more easily executed by computers.	More difficult to understand, often because it is often written in binary code.
Examples	<ul style="list-style-type: none"> • Java • Visual Basic • Python • C# 	<ul style="list-style-type: none"> • Machine code • Assembly language
What do you need in order to run code written in these languages?	Either an interpreter or a compiler to enable the code that is typed to be translated into machine code so that the computer can run it.	Machine code is translated into Assembly language.
What does code look like?	<p>One line of code might do <i>several</i> things, e.g.</p> <p>A = B + C</p> <p>This instruction finds out the values of B and C, adds them together, then stores the result in A.</p> <p>As one line of code can do several things, one line of a high-level language often translates into several lines of machine code.</p>	<p>One line of assembly language might do the same task as most people would do, but it is written in binary code and '0's, but it translates into machine code.</p> <p>LDA B ADD C STA A</p> <p>These instructions do the same task as most people would do, but it is written in binary code and '0's, but it translates into machine code.</p>
Suitability	<ul style="list-style-type: none"> • More appropriate if the program is to be used on a variety of different computer files. • Fewer people are proficient in high-level than low-level languages, and this may dictate the language type used. 	<ul style="list-style-type: none"> • Likely to be used on a variety of different computer files. • Suited to where the program is quickly executed.

INSPECTION COPY

COPYRIGHT
PROTECTED



Translators



Translator – a program that translates **source code** (which is written in a programming language) into **object code** (which can be executed by the computer). There are two types of translator used for GCSE:

- **Interpreters** translate then execute high-level code, one line at a time.
- **Compilers** translate an entire high-level program before executing it.

As in other areas, there is no 'best' choice, but there are advantages and disadvantages.

Interpreter	
+ A program that contains errors can still be run, up to where the error exists	+ A compiled object code is faster than reinterpreting source code
+ Debugging is easier as the source code line can be more easily pinpointed	+ It is more difficult to modify a compiled program
- Every time you run the program, it needs to be interpreted, which is time-consuming	- More memory is used in the process than for a compiled program
- It is easier for unauthorised people to access your source code	- The entire program must be translated in order for it to compile

2.5.2. The Integrated Development Environment (IDE)



Integrated Development Environment (IDE) – a program that provides a development environment for a programming language. Examples include Eclipse (for Java, among other languages), PyCharm (for Python), and Visual Studio (for Visual Basic, among other languages).

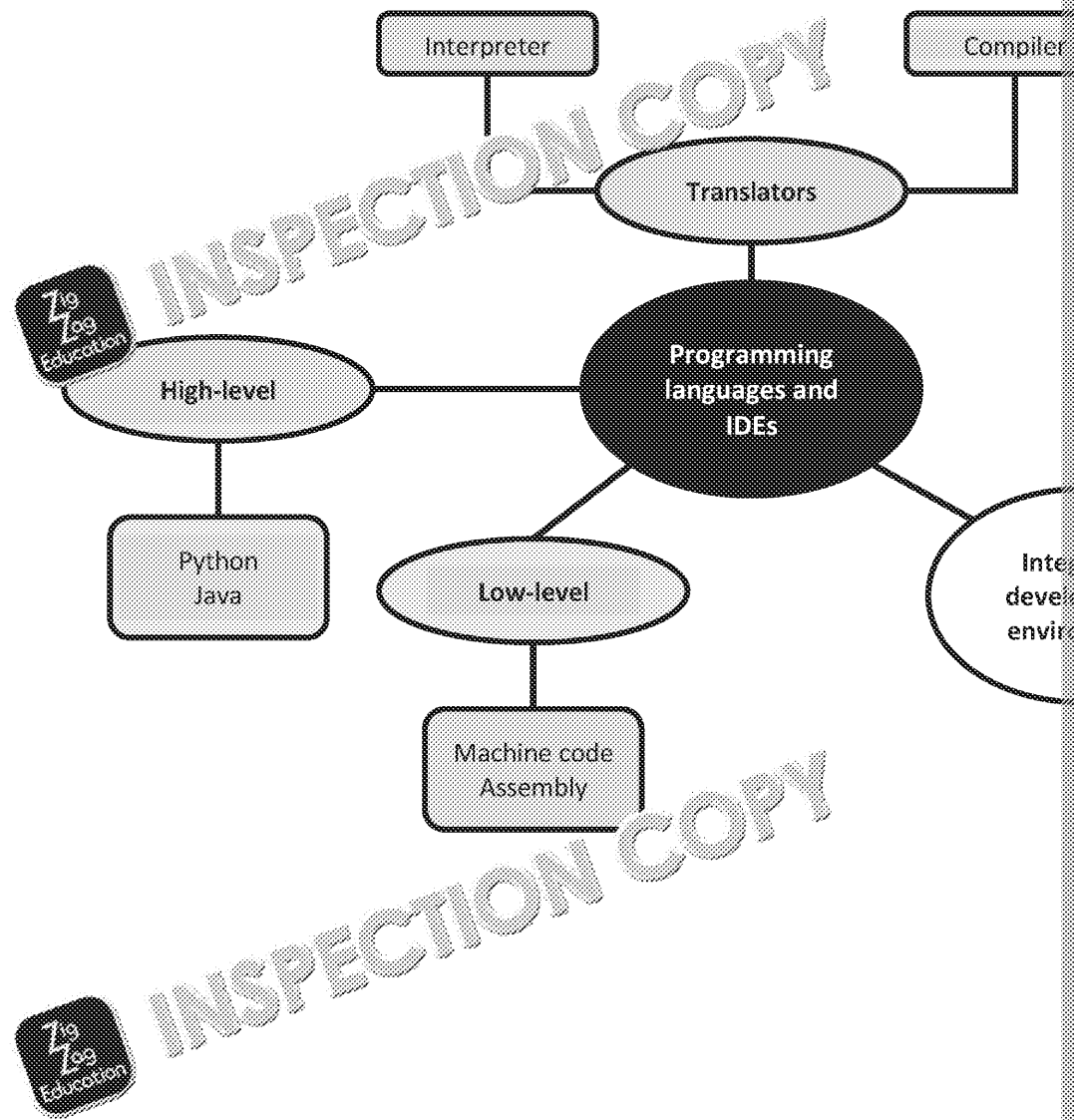
The tools available differ between IDEs, although they usually include the following:

Editor	This is the part of the IDE where the developer edits the source code. It often includes a graphical user interface using drag-and-drop tools.
Error diagnostics	As well as highlighting where an error has been made, the IDE can often identify the type of error and even offer possible solutions.
Runtime environment	This feature allows you to run your program as you would on a real computer, often with a 'play' button, or equivalent.
Translators	IDEs for high-level languages will provide a compiler. Assembly IDEs will include an assembler.

COPYRIGHT
PROTECTED



Programming Languages and IDEs Mind Map



INSPECTION COPY

COPYRIGHT
PROTECTED



Sample Examination-style Questions

28. Programs can be written using **low-level** code or **high-level** code. State **one** program in **each** of these.

.....

.....

.....

.....

29. Code written in a high-level language can be translated by an **interpreter** or a **compiler**. State **one** difference between these two types of translator, and state **one** advantage of each.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

INSPECTION COPY

COPYRIGHT
PROTECTED



Sample Answers

These answers are examples of what a good answer to the exam-style questions might be. There may be other possible answers.

1.1. Systems Architecture

1. An embedded system is a computer that forms part of a larger electrical or mechanical system. Examples might include microwaves and petrol pumps.

2.

Component	Description
Program counter	Stores the memory address of the next instruction to be executed.
Accumulator	Stores the result of arithmetic and logical calculations.
Control unit	Sends signals to other components to coordinate their operation.

i

Some exam questions are structured in this way, helpfully using partially completed tables or prompts, such as 'program counter' above. This is not always the case, and you may need to be sufficient to be able to construct a table like this from a starting point of a blank page.

1.2. Memory and Storage

3. Random Access Memory stores data and instructions for programs currently running. Read Only Memory stores data that should not be deleted or edited.

RAM might store a word-processed document that is currently being edited. ROM is likely to store bootstrapping instructions for an operating system.

4. a. Durability is important; a device must be sturdy enough to survive being dropped many times over. Portability is also important; it should not be cumbersome if it is to be used on the go. Cost can be important; the price per gigabyte of storage might be particularly relevant. b. One suitable device would be a USB flash drive.

5. a. $128 + 16 + 8 + 2 + 1 = 155$
b. $1001 = 9$; $1011 = B$; 9B

i

You will probably be asked to convert between binary, denary and hexadecimal. At the end of the exam, if you have time, a good way to check your answers is to convert them back to the original format. Did you convert a binary number to decimal? Convert it back to binary. Did you convert a decimal number to hexadecimal? Convert it back to decimal. Does the result of this conversion match the original number?

6. a. 01101000
b. No. It is multiplied by four.
c. Overflow would occur. There are not enough bits in this number to store the result.
7. a. A collection of every possible letter, number, symbol, etc. available to a computer.
b. 77

INSPECTION COPY

**COPYRIGHT
PROTECTED**



8. a. i. The number of different colours possible within a particular image
 ii. 256
- b. $640 * 480 = 307,200$ bytes
 $307,200 / 1,000 = \underline{307.2 \text{ kilobytes}}$



When you try exam questions that require calculations, make sure you show only if it is a comprehensive attempt at every single mark, it increases the likelihood of marks, even if your calculations contain an error.

9. Part of secondary storage is treated as an extension of main memory. Data is full or nearly full.

1.3. Computer Networks. Communications and Protocols

10. One piece of hardware is a network interface card. This is a printed circuit board that connects the computer to the Ethernet cable is plugged in. If the network interface card has an antenna. Another piece of hardware is a router. The router connects different networks and this interconnection allows data to be passed around the Internet.
11. A switch connects devices within the same LAN, while a router provides connectivity between different networks.
12. A URL is the web address of a resource that the user would like to access, such as a website. It tells the browser which resource to access. A web server stores web pages and related files, which are transmitted across the Internet when requested.
13. a. A protocol is a set of rules governing how one device communicates with another.
 b. POP3 – Post Office Protocol Version 3. This retrieves emails from the server once they are retrieved. IMAP – Internet Message Access Protocol. This allows emails to be retrieved by several devices, each accessing the same email account.

1.4. Network Security

14. People can read staff members' passwords over their shoulders. This can be prevented by using two-factor authentication, which requires the user to authenticate the login with their password and a second factor, such as a security token or a mobile phone.

Data can be intercepted as it is transmitted between devices. Encryption can be used to prevent this by scrambling data so that it only makes sense when accessed by the intended recipient.

Malware might be installed on a system, which could steal or corrupt patient data. Regular updates and antivirus software is installed and kept up to date is one means of countering this.

1.5. Systems Software

15. One role is the management of hardware. The operating system is responsible for managing the hardware and software. It can only do this if the correct drivers are installed. Another role is process management. When lots of tasks require the processor's attention, it is the job of the operating system to prioritise and decide the order in which they should be executed. A third role is file management. The operating system allows the user to add, delete or edit files, and also decides where each file is to be stored.



Questions that begin with the term 'describe in detail' need to be addressed in detail. With these questions, the mark schemes usually give the examiners a list of points which are worth a mark. The alternative, listing every possible answer is impractical. This counts in your favour, but it doesn't do any harm to aim for maximum marks. Where there's a 6-mark 'describe-in-detail' question, aim for 9 marks.

**COPYRIGHT
PROTECTED**



1.6. Ethical, Legal, Cultural and Environmental Impacts of Digital Technology

16. Computers have had a positive environmental impact. The volume of fossil fuel needed to produce a computer is weighed against the volume needed to post a letter is tiny. Even products that are in digital form, such as books and CDs, can be delivered across the Internet, saving on the need for driving to the shop. Computers can even run programs that can reduce the need for energy. A motion sensor, a microprocessor and a light can be connected in a way that turns the light off if there is a person in the room. Satellite navigation technology can adjust a route to be the shortest, meaning less petrol is consumed.

However, there are negative impacts. In order to produce a computer, various raw materials are needed, which can be a blight on the landscape that most computer users never have seen. The production of computers, as well as the transport of these raw materials, requires large amounts of energy, which is offset by the fuel saved by computerisation.

i

There will usually be a question like this, which will be worth a large number of marks. It will depend on good-quality written communication. The question is quite likely to be about the ethical, legal, cultural or environmental issues, but it could be anything (as you look through this revision guide, the more likely it is to feature in a question such as this).

There are some guidelines you can follow for such questions:

- Plan your answer, rather than jumping straight in. The model answer above shows two paragraphs for each side of the discussion, and each paragraph makes several points.
- Know your subject – it is unlikely that you will score highly on a topic you have not written your answer is.
- Provide examples to support any points you make (examples above include satellite navigation).
- Unless it's indicated that you should choose one side over another in a debate, you should discuss both sides equally.
- Proofread your finished answer. Too many spelling mistakes or illegible writing will reduce your marks.

17. Personal data must be accurate and up to date. If a person's data changes, they must update the data in their system. Additionally, personal data must be obtained for one purpose cannot be reused for another purpose without the person's consent.

i

This was a 4-mark question, though the question only asked explicitly for two points. In this case, that means that each piece of information must be worth 2 marks. The Data Protection Act will only ever be worth 1 mark, and additional insight is needed. The question requested by the question, a description was provided. Different questions might ask for 'how' or 'why' and/or examples, but those marks need to come from somewhere else.

**COPYRIGHT
PROTECTED**



2.1. Algorithms

18. An algorithm is a series of instructions that describes how to solve a specific problem.

i

As you've seen by this point, this guide is full of definitions. It is highly likely that definitions will be required in the exam, although it's impossible to determine which ones, assuming you have revised well, are the easiest marks available. One way to prepare is to make flash cards, with the word on one side and its definition on the other. Making them your own, and it's easy for others to help you to study by testing you on the definitions.

19. Terminator Process Decision

20. Pairs of numbers are compared (first with second, second with third, third with fourth, etc.). If they are not in order, they are switched. Once all pairs have been compared, a new pass will occur until either $n-1$ passes have occurred, or an entire pass occurs with no numbers being switched.

i

This question is fairly complex. In circumstances like this, you might find it helpful to write a diagram. While it probably wouldn't be worth any marks, it could clarify any points that you haven't worded as well as you would have liked.

2.2. Programming Fundamentals

21. a. i. total iv. >
ii. keeps a running total v. while
iii. +

b. They would need to input a value of '-1' or less.

i

The best way to prepare for questions like this one is to write code – lots of it. In a program, you should make an effort to understand every part of every line. It's not enough to simply having a program that works; make sure you can understand why it works. Write down any lines of code that you don't understand.

22.

Data	Integer	Real	Boolean
Money in account		✓	
Account holder's name			
Number of whole years the account has been active	✓		
Account holder's postcode			
Whether or not an overdraft is permitted			
A single-letter code that identifies the account type			

23.

```
highCount = 0
for loopCount = 0 to 9
    if data[loopCount] > 5 then
        highCount = highCount + 1
    endif
next loopCount
```

**COPYRIGHT
PROTECTED**



2.3. Producing Robust Programs

24. a.

a	data[a]	b	c
0	4	9	False
1	2	9	False
2	7	9	False
3	9	9	True
4			

i

Trace tables are an effective way of assessing whether or not you understand a piece of advice that can be given here is to work through the code as if you're executing it. While you're executing, for instance, if the values 1 and 4 do not exist. Compute at a time, so you need to emulate the code. Even if, halfway through completing a table, you see how the rest of the table will look, don't rush.

b. It uses a linear search, checking each item until B is found.

25.

Type of error	Description	
Logic	An error that causes abnormal behaviour without causing the program to crash.	Code is wrong by 1,000 in a percentage
Syntax	An error that involves code that breaks the rules of grammar of the language.	Misspelling instead of

2.4. Boolean Logic

26. a. AND

b.

Q	P	Output
0	0	0
0	1	0
1	0	0
1	1	1

27. a. Reject

b. Gold card

c. Standard card

2.5. Program Languages and Integrated Development Environments

28. An advantage of low-level programming is that the instructions are generally easier to understand. An advantage of high-level programming is that the code is easier for people to understand.

29. An interpreter translates each line of source code into object code, executing it immediately. A compiler translates the entire source file into an object file before any instructions are executed. An advantage of an interpreter is that one part of the program can be tested, even if there are errors elsewhere. An advantage of a compiler is that a compiled program executes more quickly than an interpreted program.

i

Look closely at the answers to question 28 (worth 2 marks) and question 29 (worth 2 marks). In this case, there is a clear attempt at every individual mark. There is no overlap between marks, and each mark has a single sentence. This puts the examiner in a good mood, as they know exactly where each mark is supposed to go (and an examiner in a good mood is in a good mood for you). It also puts your mind at ease, as you've clearly attempted each individual mark.

COPYRIGHT
PROTECTED



Glossary

Term	Definition
Abstraction	The practice of hiding layers of complexity within a problem to focus on a specific aspect.
Accumulator	Stores the intermediate result of a calculation.
Algorithm	A series of instructions to solve a problem.
Algorithmic thinking	Systematically solving a problem by using tools such as abstraction.
Analogue	Signals that are continuous, i.e. the midpoint between two values can be represented.
Analogue to digital converter	Converts analogue signals, such as sound, and represents them as digital data.
Arithmetic Logic Unit	A component of the CPU that performs calculations and logical operations.
Arithmetic operator	Performs a numeric operation, such as addition or multiplication.
Array	A data type in which multiple data items of the same type are stored in a single memory location.
Artificial intelligence	The branch of computing where technology attempts to perform tasks that normally require human intelligence.
ASCII	A character set consisting of 128 characters.
Assignment	The practice of providing a value to a variable.
Authentication	The process of ensuring that a user of a system is who they claim to be.
Bandwidth	The amount of data a network can transmit over a given amount of time, often measured in megabits per second.
Binary	A number system comprising two symbols: 0, 1.
Binary search	A search algorithm that begins in the middle of a sorted data set and repeatedly divides the data items with each item that it examines.
Binary shift	Moving the digits of a binary number to the left (to multiply by two) or to the right (to divide by two).
Bit	The smallest unit of binary data – a binary digit – which can be either 0 or 1.
Boolean	A data type that can store one of two values – true or false.
Boolean logic	Determining whether an output is true or false, based on Boolean values and operations such as AND, OR, NOT.
Boolean operation	Operations performed upon Boolean values (true or false) to produce other Boolean values. Examples include AND, OR, NOT.
Boundary data	Test data that represents the highest or lowest permissible values for a data set.
Brute-force attack	Gaining unauthorised access to a system by attempting every possible combination of characters until the login credentials (such as a password) are found.
Bubble sort	A sorting algorithm that works by repeatedly comparing pairs of data items and swapping their positions as necessary.
Bus	A connection between computer components, along which data is transferred.

INSPECTION COPY

**COPYRIGHT
PROTECTED**



Term	Definition
Byte	A sequence of eight bits.
Cache	Memory with shorter response times than RAM, so used to store recently used data or instructions.
Calling	The process of telling a subprogram to take place from elsewhere.
Capacity	A characteristic of secondary storage that describes how much data it can store.
Casting	The process of converting data of one type to be stored in a different type.
Central Processing Unit (CPU)	A computer component that performs calculations and controls, interpreting and executing instructions.
Character	Either a single letter (upper or lower case), a single numeral, or a single invisible character (such as a tab or space). Most keyboards have 256 characters.
Character set	The characters available within a particular system, each with an individual code.
Client-server	A usage model in which a server provides a service, such as access to that service.
Clock	A component of the CPU that synchronises activities.
Cloud computing	Storage and processing takes place remotely rather than locally.
Colour depth	The number of distinct colours available (though not necessarily used).
Comment	A plain English line (or multiple lines) added to program code by a computer but is useful to other programmers.
Comparison operator	Used to compare two values to see which is larger, or whether they are equal.
Compiler	A translator that translates an entire program before running it.
Compression	The process by which a file is made smaller in order to be stored more efficiently.
Computer Misuse Act	A law that makes unauthorised access of a computer system illegal.
Condition-controlled iteration	A program structure in which a section of code repeats until a condition is met.
Constant	A named location in memory can be used for storing a single data value that does not change during program execution.
Control Unit	A component of the CPU that coordinates the activity of other components.
Copyright, Design Rights and Patents	A law that provides protection for intellectual property, such as software.
Core	A processing unit with a control unit, arithmetic logic unit, and cache. A system can have one or more cores.
Count-controlled iteration	A program structure in which a section of code repeats a certain number of times.
Culture	A broad term that covers everything under the description of a particular society.

COPYRIGHT
PROTECTED

Term	Definition
Data interception	Obtaining a copy of data as it is being transmitted from one device to another.
Data Protection Act	A law that governs how personal information is stored and processed.
Decomposition	The practice of taking a larger problem and dividing it into smaller problems.
Defragmentation	Reordering the contents of a disk so that a file is stored, as a single unit, in one location, rather than in pieces across the disk.
Denary	A number system comprising 10 symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
Denial-of-service attack	Making so many requests for a service – such as a web page – that the service is unable to get a response.
Digital	Information that are comprised of only 1s and 0s, with nothing in between.
Durability	A characteristic of secondary storage that describes how well it can withstand wear or damage.
Editor	Part of an IDE that allows either the entry of code or the modification of the user interface components.
Embedded system	A computer that forms part of a larger electrical or mechanical system.
Encryption	The process of converting data into a code, thereby preventing unauthorized access.
Erroneous data	Test data that should not be accepted by a system – it is typically data that appears as and when they are supposed to appear.
Error diagnostic	Part of an IDE that helps programmers to identify and remove errors in their code.
Error rate	The percentage of data packets that is not received exactly as it was sent.
Ethics	The practice of determining right from wrong, often in connection with the use of technology.
Fetch–execute cycle	The process by which programs are run by a computer; instructions are fetched from memory into the CPU, and executed (carried out), in a cycle.
Firewall	Hardware or software technology that filters network traffic between a private network or an individual computer and the Internet.
Flow chart	A diagrammatic means of representing algorithms, using shapes and arrows to show the sequence of steps.
Function	A type of subprogram that returns a value.
Gigabyte	One billion bytes or one thousand megabytes.
Hexadecimal	A number system comprising 16 symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
High-level programming language	A programming language in which one instruction can translate into many machine instructions. Examples include Python, Java and Visual Basic.
Hosting	Storing a website, and related items, in such a way that it can be accessed over the Internet.
Insertion sort	A sorting algorithm in which a data structure is divided into two parts: a sorted part and an unsorted part. Data items are moved from the unsorted part to the sorted part.
Integer	A data type comprising whole numbers, including positive, negative and zero.

**COPYRIGHT
PROTECTED**



Term	Definition
Integrated Development Environment (IDE)	A piece of software with which programmers can create applications.
Internet	A network that spans the globe, connecting together a huge number of smaller networks.
Internet Protocol (IP) address	A series of numbers that uniquely identifies each device that is connected to the Internet at any one time.
Interpreter	A translator that translates and executes line by line.
Iteration	A program structure in which a section of code might be executed repeatedly, typically due to a FOR, WHILE or UNTIL statement.
Iterative testing	Testing that takes place repeatedly as a system is under development.
Kilobyte	One thousand bytes.
Latency	A measure of the time taken for data to travel from one point to another.
Licence	An agreement between a vendor (who makes or sells a product) and a user (who buys or uses it).
Linear search	A search algorithm that begins at one end of a data structure and checks each item in turn.
Local area network	A series of interconnected devices over a small geographic area, such as a school or a campus.
Logic error	An error in which code runs but produces the wrong output because an incorrect arithmetic operator is used.
Lossless	A category of compression in which no data is lost as a result of compression.
Lossy	A category of compression in which data may be irretrievably lost as a result of compression.
Low-level programming language	A programming language in which one instruction translates to one machine instruction. Assembly and machine code are low-level languages.
MAC address	Media Access Control address – a unique identifier built into a network card in order to identify it on a network. Unlike an IP address, it does not change between sessions.
Magnetic	A category of storage in which data is stored in the form of magnetised particles.
Maintenance	The process of continually ensuring that a system meets requirements that may change.
Malware	Any piece of software that causes harm to a computer system.
Megabyte	One million bytes or one thousand kilobytes.
Memory address register	Stores the memory location to be accessed next by the CPU.
Memory data register	Stores a piece of data that has either just been read from memory or is about to be written to memory.
Merge sort	A sorting algorithm that divides a data structure into individual items, sorts the data into pairs, then groups of four, groups of eight, etc.
Mesh	A network topology in which all devices are connected to all other devices.
Misuse	Any attempt to use a system in some way other than how it was intended.
Monochrome	Describes an image where only varying tones of one colour are used.

COPYRIGHT
PROTECTED

Term	Definition
Network interface card	A computer component that facilitates connection of a computer to a network.
Nibble	A sequence of four bits.
One-dimensional array	An array in which each element is identified by a sequential index.
Open source	A category of software in which anyone can use, alter or distribute the source code.
Operating system	The software that manages the hardware, from which applications can be launched.
Operator	A symbol used to represent an operation performed on one or more operands.
Optical	A category of storage in which data is read and written using light.
Overflow	Occurs when the result of a calculation is too large to be stored in the allocated space.
Parameter	A piece of data that can be passed into a subprogram.
Peer-to-peer	A usage model in which all devices are potentially both clients and servers (such as files) can be spread across a network, and are often shared.
Penetration testing	Simulation of a cyberattack on a computer system, in order to identify weaknesses that they are discovered by hackers.
Peripheral	Any device that connects to a computer, such as a mouse or keyboard.
Petabyte	One quadrillion bytes or one thousand terabytes.
Phishing	Obtaining personal data or login credentials by presenting a fake website or service that people trust.
Physical security	Low-tech security solutions such as locks on doors or removal of sensitive data from devices.
Pixel	The smallest unit within an image; can be considered a dot of a specific colour, and cannot be subdivided.
Portability	A characteristic of secondary storage that describes how easy it is to move data between computers.
Primary storage	Also known as RAM or main memory, stores data and applications that are currently in use.
Procedure	A type of subprogram that does not return a value.
Program counter	Stores the memory address of the next instruction to be executed.
Proprietary	A category of software in which rights to use it are controlled by the creator, with those rights typically being enforced by law.
Protocol	A set of rules governing how data is transmitted across a network.
Pseudocode	A cross between plain English and a programming language, used to describe an algorithm.
Random-access memory	Also known as primary storage or main memory, stores data and applications that are currently in use.
Read-only memory	Any memory that can be read from but not written to.
Real	A data type comprising numbers that can include fractions; typically used for financial calculations.
Record	A data structure in which multiple data items of different types are stored together.
Register	A low-capacity data store – typically 32 or 64 bits – that forms part of a processor.

COPYRIGHT
PROTECTED

Term	Definition
Reliability	A characteristic of secondary storage that describes how likely it is to continue to work.
Resolution	The number of pixels that make up an image, often described in terms of width and height.
Return	The last instruction that executes within a function, which returns control to the caller of that function.
Router	A network device that allows communication between Local Area Networks (LANs).
Runtime environment	Part of an IDE that simulates another computer. An application can be run, even if the system it is being developed for is different from the one used to develop it.
Sample size	The number of bits that make up each sample of a sound.
Sampling frequency	The number of times per second a sound is sampled by a device.
Search	An algorithm used to determine whether a particular data item exists in a data structure, and often its position within that data structure.
Secondary storage	Non-volatile storage (i.e. storage that does not require constant power) used for applications that are not currently in use.
Selection	A program structure in which one of two or more paths can be chosen, typically determined by the evaluation of an IF statement.
Sequence	A program structure in which each instruction occurs once, in the order in which it is written.
Social engineering	Exploiting people as the weakness in any computer system.
Solid state	A category of storage that stores data electronically, with no moving parts.
Sort	An algorithm used to place data items into some kind of order, such as alphabetical or numeric.
SQL injection	Entering SQL code into a data entry form in order to manipulate the data in a database.
Standard	Any commonly established way of working, so that people from different countries work with compatible technology.
Standard algorithm	A commonly used algorithm that would be useful in a wide range of situations.
Star	A network topology with a central device – typically a switch or router – to which all other devices are connected.
String	A data type consisting of a sequence of characters.
Structure diagram	A diagram showing, using a diagram, the component parts of a system and how they are related.
Structure Language (SQL)	A language for reading and altering the contents of a database.
Subprogram	A named set of instructions that forms part of a program.
Switch	A network device that enables communication between devices on a Local Area Network.
Syntax error	An error that comes from failing to follow the rules of grammar in a programming language.

**COPYRIGHT
PROTECTED**



Term	Definition
TCP/IP stack	A layered set of common communication protocols.
Terabyte	One trillion bytes or one thousand gigabytes.
Terminal testing	Testing that takes place after development.
Topology	The logical or physical layout of a network – where components are connected to one another.
Trace table	A table used to track the values of variables as an algorithm runs.
Translator	A piece of software that converts source code (written by a programmer) into machine code (executable by the computer).
Transmission medium	Any means by which data can be transmitted between devices.
Truth table	Maps all possible combinations of Boolean inputs to the corresponding output of a function.
Two-dimensional array	An array in which each element is identified by a pair of indices. Any point on a Cartesian plane can be identified using X and Y coordinates.
Typical data	Test data that represents normal use of a system.
Unicode	A character set that includes ASCII, as well as many other characters from other alphabets.
Uniform resource locator	The address of a web resource, typed into a browser.
User access levels	A security arrangement in which a user's login details will only grant them access to a system that they need to do their job.
User interface	The means by which a human and a computer interact with each other.
Utility	An application that maintains a computer in some way, such as defragmenting a hard drive.
Validation	The process of ensuring that data entered into a system is suitable for its intended use.
Variable	A named location in memory capable of storing a single data item.
Virtual memory	Part of secondary storage is used as an extension to main memory for applications that are currently in use. Typically used when the main memory is full.
Von Neumann architecture	A means of organising a computer system in which both data and instructions use the same memory unit, and data and instructions use a common bus to move between the memory and the processor.
Wide area network	A series of interconnected devices over a large geographic area, spanning multiple countries.
Wired	A means of connecting devices on a network that uses wires, such as Ethernet.
Wireless	A means of connecting devices on a network that does not use wires, such as Wi-Fi.
Wireless point-to-point	A network device that allows Wi-Fi-enabled devices to connect to each other without the need for a central access point.
World Wide Web	An information system that uses the Internet as its means of communication.

COPYRIGHT
PROTECTED