



Cambridge
IGCSE™ and O Level

Computer Science

Second Edition

David Watson
Helen Williams



Contents

	Introduction	iv			
Section 1	Computer Systems		REVISED	TESTED	EXAM READY
1	Data representation	1			
2	Data transmission	15			
3	Hardware	25			
4	Software	49			
5	The internet and its uses	62			
6	Automated and emerging technologies	80			
Section 2	Algorithms, programming and logic				
7	Algorithm design and problem solving	98			
8	Programming	113			
9	Databases	133			
10	Boolean logic	137			
	Practice Paper 1A Computer Systems	142			
	Practice Paper 2A Algorithms, programming and logic	152			
	Answers to exam-style questions	163			
	Answers to Practice Paper 1A	185			
	Answers to Practice Paper 2A	188			
	Index	196			

1

Data representation

Key objectives

The objectives of this chapter are to revise:

- number systems
- the binary, denary and hexadecimal number systems
- conversion of numbers between all three number systems
- use of the hexadecimal (hex) number system
- binary addition
- overflow error
- logical shifts
- two's complement format for negative and positive binary numbers
- text, sound and images
- ASCII and Unicode character sets
- representation of sound in a computer
- sampling rate and sample resolution
- image representation, including resolution and colour depth
- data storage and file compression
- calculation of file sizes
- the need for data (file) compression
- lossy and lossless compression

1.1 Number systems

REVISED

1.1.1 Binary represents data

No matter how complex the system, the basic building block in all computers is the binary number system. This system is chosen because it consists of 1s and 0s only which correspond to ON and OFF states in the computer system.

1.1.2 Binary, denary and hexadecimal number systems

The binary system

The **binary number system** is based on the number 2; it can only use the two values 0 and 1 (these are referred to as **bits**). The binary heading values are 2^0 , 2^1 , 2^2 , 2^3 and so on.

The maximum size of a binary number you will see in the exam is 16 bits

If an 8-bit system is being used the headings are: 128 (2^7), 64 (2^6), 32 (2^5), 16 (2^4), 8 (2^3), 4 (2^2), 2 (2^1) and 1 (2^0). A typical binary number, based on this system, would be 0 1 1 1 0 0 1.

The denary system is a base 10 number system with column headings: 10^0 (1), 10^1 (10), 10^2 (100), 10^3 (1000) and so on.

Converting from binary to denary

To convert from binary to denary, simply add together all the heading values where a 1-value appears.

For example:

$$0\ 1\ 1\ 1\ 0\ 0\ 1 = 64 + 32 + 16 + 8 + 1 (= 121)$$

$$0\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1 = 1024 + 512 + 256 + 128 + 8 + 2 + 1 (= 1931)$$

... and so on.

Converting from denary to binary

To convert from positive denary to binary, it is necessary to carry out successive divisions by 2 until a zero value results. The remainders are read from bottom to top to give the binary value:

For example, to convert 142 to binary:

2	142	
2	71	remainder: 0
2	35	remainder: 1
2	17	remainder: 1
2	8	remainder: 1
2	4	remainder: 0
2	2	remainder: 0
2	1	remainder: 0
	0	remainder: 1

↑

read the remainder from bottom to top to get the binary number:
1 0 0 0 1 1 1 0

(Note: if the answer is, for example, 111011 and 8-bits are used to represent numbers, then just infill with zeros to give: 00111011.)

The hexadecimal system

The **hexadecimal number system** is based on the number 16. The 16 digits are represented by the numbers 0 to 9 and the letters A to F (representing 10 to 15). The hexadecimal headings are 16^0 , 16^1 , 16^2 , 16^3 , and so on. A typical hexadecimal number would be 1F3A.

Converting from binary to hexadecimal and hexadecimal to binary

To convert a binary number to a hexadecimal number, it is first necessary to split the binary number into 4-bit groups starting from the right-hand side. If the final (left-most group) doesn't contain four binary digits, then infill by zeros is done. Each 4-bit group is then assigned a hexadecimal digit. For example:

1011 1111 0000 1001 becomes (11) (15) (0) (9) that is, BF09

1 0011 1110 0111 must first be rewritten as **0001** 0011 1110 0111 which becomes (1) (3) (14) (7) that is 13E7

To convert from hexadecimal to binary, it is necessary to write the 4-bit binary code for each hexadecimal digit. For example:

45A becomes 0100 0101 1010

E48D becomes 1110 0100 1000 1101

Converting from hexadecimal to denary and from denary to hexadecimal

To convert a hexadecimal number to denary, it is necessary to multiply each hexadecimal digit by its heading value and then add them together. For example,

$$1FD = (1 \times 256) + (15 \times 16) + (12 \times 1) = 508$$

$$4EB5 = (4 \times 4096) + (14 \times 256) + (11 \times 16) + (5 \times 1) = 20149$$

To convert from denary to hexadecimal, it is necessary to carry out successive divisions by 16 until zero value results. The remainders are read from bottom to top to give the hexadecimal value. For example, to convert 2004 to hexadecimal:

16	2004		
16	125	remainder: 4	
16	7	remainder: 13	
	0	remainder: 7	

↑

read the remainder from bottom to top to get the hexadecimal number: 7D4 (D = 13)

1.1.3 Uses of the hexadecimal system

The hexadecimal number system is often used by computer programmers and designers because it is easier to deal with, for example AF01, than the binary equivalent of 1010111100000001. Some of the main uses of the hexadecimal system are listed here.

- **Error codes** refer to memory locations where the error occurs; they are automatically generated by the computer.
- A **Media Access Control (MAC) address** identifies a device on a network (via the NIC). The MAC address is in the format NN-NN-NN-DD-DD-DD (first six digits are the manufacturer code and the last six digits are the serial number of the device).
- An **Internet Protocol (IP) address** is given to a device when it joins a network; there are two types – IPv4 (32-bit code) and IPv6 (uses 128-bit code).
- **Hypertext mark-up language (HTML) colour codes**; the colour of each pixel on screen is made up of a combination of red, green and blue; the amount of each colour is represented by a hex code. For example, # FF 00 00 is pure red, # FF 80 00 is a shade of orange, # B1 89 04 is a tan colour.

1.1.4 Addition of binary numbers

Binary addition involves a carry and a sum for each of the 2 or 3 bits being added:

binary digit			operation	carry	sum
0	0	0	0 + 0 + 0	0	0
0	0	1	0 + 0 + 1	0	1
0	1	0	0 + 1 + 0	0	1
0	1	1	0 + 1 + 1	1	0
1	0	0	1 + 0 + 0	0	1
1	0	1	1 + 0 + 1	1	0
1	1	0	1 + 1 + 0	1	0
1	1	1	1 + 1 + 1	1	1

For example:

Add 00100111 + 01001010

We will set this out showing carry and sum values:

$$\begin{array}{r}
 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \\
 + 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \\
 \hline
 1 \ 1 \ 1 \quad \leftarrow \text{carry values} \\
 \hline
 0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1 \quad \leftarrow \text{sum values}
 \end{array}$$

Answer: 01110001

column 1: $1 + 0 = 1$ no carry
 column 2: $1 + 1 = 0$ carry 1 to next column
 column 3: $1 + 0 + 1 = 0$ carry 1 to next column
 column 4: $0 + 1 + 1 = 0$ carry 1 to next column
 column 5: $0 + 0 + 1 = 1$ no carry
 column 6: $1 + 0 = 1$ no carry
 column 7: $0 + 1 = 1$ no carry
 column 8: $0 + 0 = 0$ no carry

Overflow

Overflow occurs if the result of a calculation is too large for the allocated word size (for example a word size of 8 bits can represent a maximum value of 255).

For example:

Add 0 1 1 0 1 1 1 0 and 1 1 0 1 1 1 1 0 (using an 8-bit word size)

$$\begin{array}{r}
 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0 \\
 + 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \ 0 \\
 \hline
 1) \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \quad \leftarrow \text{carry values} \\
 \hline
 \text{ninth bit} \rightarrow 1) \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \quad \leftarrow \text{sum values}
 \end{array}$$

This addition has generated a ninth bit. The 8 bits of the answer 0 1 0 0 1 1 0 0 give the denary value $(64 + 8 + 4)$ of 76 which is clearly incorrect (the denary value of the addition is $110 + 222 = 332$).

The generation of a ninth bit is a clear indication that the sum has exceeded the maximum value possible for 8 bits; that is, 255 ($2^8 - 1$). This is known as an **overflow error** and is an indication that a number is too big to be stored in the computer using, in this case, an 8-bit register.

This shows that the greater the number of bits which can be used to represent a number then the larger the number that can be stored. For example, a 16-bit register would allow a maximum value of $2^{16} - 1$ (= 65 535) to be stored, a 32-bit register would allow a maximum value of $2^{32} - 1$ (= 4 294 967 295), and so on.

1.1.5 Logical binary shifts

Logical shifts involve shifting (moving) bits to the left (multiplying by 2 for each shift) or the right (dividing by 2 for each shift). If shifting to the left or right results in a loss of 1-bits, then this would result in an error.

When shifting a binary number, any gaps created by the shift operation can be filled by zeros. For example, the denary number 54 is 00110110 in binary. If we put this into an 8-bit register:

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



The left-most bit is often referred to as the **most significant bit**

If we now shift the bits in this register two places to the left:

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



Note how the two right-most bit positions are now filled with 0s

The value of the binary bits is now $54 \times 2^2 = 216$.

Suppose we now shift the original number one place to the right:

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



Note how the left-most bit position is now filled with a 0

The value of the binary bits is now $54 \div 2^1 = 27$.

Suppose we now shift the original binary number four places to the left:

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

This should give $54 \times 2^4 = 864$, but actually gives 96 which is clearly incorrect. Since two of the 1-bits were lost following a logical shift, an error would be generated. Similarly, if we shift the original binary number four places to the right:

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

Again, an error would be generated since the result of the right shift should be $54 \div 2^4 = 3.375$, but actually results in the value 3.

1.1.6 Two's complement (binary numbers)

To allow for the possibility of representing negative integers we make use of **two's complement** notation. For example:

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

This represents a negative number:
 $-128 + 64 + 32 + 4 + 2 = -26$

We can still store positive values. For example,
 this represents $64 + 32 + 4 + 2 = 102$

Converting denary numbers into binary in two's complement format, involves placing 1-bits in the appropriate position remembering that the right-most bit now represents -128.

To convert negative denary numbers into binary in two's complement format can be done in two ways.

Consider the number +67 in 8-bit (two's complement) binary format:

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

One method of finding the binary equivalent to -67 is to simply put 1-bits in their correct places:

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

$$-128 + 32 + 16 + 8 + 4 + 1 = -67$$

Looking at the two binary numbers above, this gives us another possible way of finding the binary representation of a negative denary number:

first, write the positive binary value, such as 67

0	1	0	0	0	0	1	1
1	0	1	1	1	1	0	0
							1
1	0	1	1	1	1	0	1

then invert each binary value

then add 1 to that number

this gives us the binary for -67

Sample questions and answers

REVISED

- Write the denary number 44 as an 8-bit binary number.
- Carry out a logical shift two places left on your binary number found in part a). Comment on your answer.
- Carry out a logical shift two places right on your binary number found in part a). Comment on your answer.
- Write the denary number 220 as an 8-bit binary number. Add this binary number to your binary number found in part a). Comment on your answer.
- Write -44 as an 8-bit binary number using two's complement format. [9]

Tips

When a comment about your answer is required, explain whether the result you get is what you would have expected; then give a reason why it is (or is not) as expected. Where a mathematical sequence of operations is needed (as in parts d) and e)), it is imperative that you show all your working so that some marks can still be gained even if your answer is incorrect. Throughout questions of this type, keep your work logical and thorough so that the examiner can easily follow your logic.

To reduce the possibility of errors, it is a good idea to write your 8-bit binary number in register/word format:

128	64	32	16	8	4	2	1

Sample high-level answer

a)

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

b)

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

This is equivalent to $128 + 32 + 16 = 176$. Shifting two places left should give the result $44 \times 2^2 = 176$, which means the actual result is the same as the expected result.

c)

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

This is equivalent to $8 + 2 + 1 = 11$. Shifting two places right should give the result $44 \div 2^2 = 11$, which means the actual result is the same as the expected result.

d)

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

This is the 8-bit binary representation of 220. Adding this to the original binary number from part (a): 00101100 results in the answer:

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



A ninth bit is generated following this binary addition

The expected result for this addition ($220 + 44$) is 264. However, the value 8 is generated. This is clearly incorrect and is due to the fact that the result of the sum exceeds the maximum value which can be represented by an 8-bit word (that is, 255). An **overflow error** has occurred.

e)

44:	0	0	1	0	1	1	0	0
inverted:	1	1	0	1	0	0	1	1
add 1:								1
result:	1	1	0	1	0	1	0	0
[-128 + 64 + 16 + 4 = -44]								

Teacher's comments

The first student has given a very well-explained answer and they have used the 8-bit word format; this greatly helps in parts b), c) and e) of this question.

Sample low-level answer

a)	2	44	
	2	22	remainder: 0
	2	11	remainder: 0
	2	5	remainder: 1
	2	2	remainder: 1
	2	1	remainder: 0
		0	remainder: 1

Gives the answer: 001101

b) 001101 becomes 1101

c) 001101 becomes 00001101

d)	220	2	
	110	2	remainder: 0
	55	2	remainder: 0
	27	2	remainder: 1
	13	2	remainder: 1
	6	2	remainder: 1
	3	2	remainder: 0
	1	2	remainder: 1
	0		remainder: 1

Gives the answer: 00111011

e) -001101

Teacher's comments

The second student has used the correct conversion method in parts a) and d), but they have written the binary numbers in the wrong order and the answer to part a) is not in 8-bit format. Answer b) is incorrect since they haven't added extra zeros (they could have gained a follow-through mark from part a) if they added the extra two zeros). Part c) would gain a follow through mark since this time the additional zeros were added. Part e) is completely wrong. Probably two or three marks maximum out of 9.

Exam-style questions

- 1 a) i) Convert the 16-bit binary number 1100 0000 1101 1110 to hexadecimal.
 ii) Convert the hexadecimal number 2A9F to a 16-bit binary number. [3]
- b) i) Convert the hexadecimal number 3FC to a denary number.
 ii) Convert the denary number 2816 to a hexadecimal number. [3]
- 2 a) Convert the following denary numbers into 8-bit binary numbers:
 i) 95
 ii) 30
 iii) 205 [3]
 b) i) Carry out the binary addition of parts a)i) and a)ii).
 ii) Carry out the binary addition of parts a)i) and a)iii). Comment on your answer. [3]
- 3 Describe three uses of the hexadecimal number system. [6]
- 4 Convert the denary number 75 into an 8-bit binary number using the two's complement format. [3]
- 5 a) i) Convert the denary number 116 into a binary 8-bit number.
 ii) Carry out a logical shift two places to the right on the binary number obtained in part a)i).
 iii) Carry out a logical shift three places to the left on the binary number obtained in part a)i). Comment on your answer. [5]
 b) i) Write the hexadecimal numbers 3C and 44 as 8-bit binary numbers.
 ii) Add the two binary numbers found in part b)i).
 iii) Carry out a logical shift six places to the right on your answer to part b)ii). Comment on your answer. [5]

1.2 Text, sound and images

REVISED

1.2.1 Text

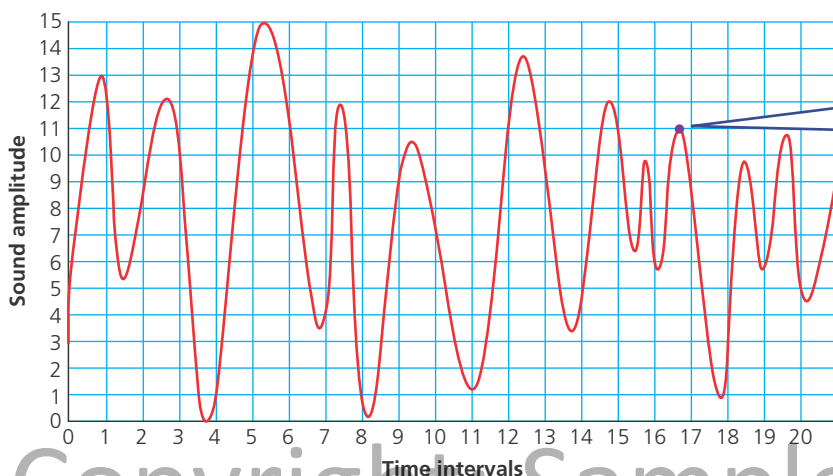
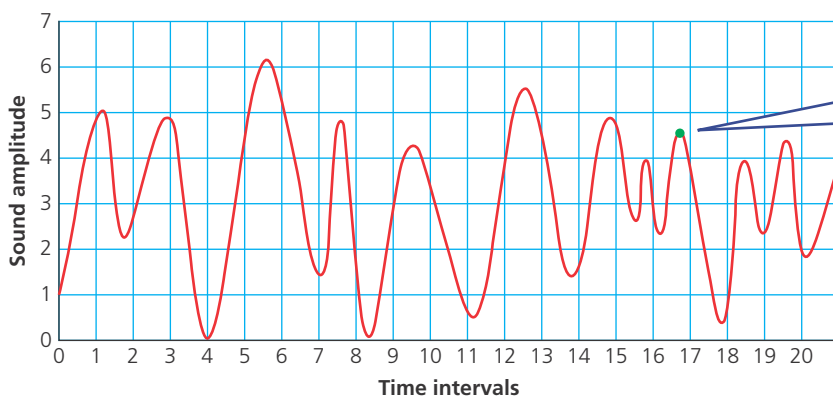
All keyboard characters (including control codes) are represented in a computer using 7-bit American Standard Code for Information Interchange (**ASCII code**) or 8-bit Extended ASCII code **character set**. For example, each ASCII value is found in a stored table when a key is pressed on the keyboard. The main drawback of the ASCII code system is it can't be used to represent non-Western languages, such as Chinese or Japanese characters. One way round this is to use **Unicode**, which can support up to 4 bytes per character (that is, up to 32 bits per character).

1.2.2 Sound

Sound is analogue data. To store sound in a computer, it is necessary to convert the analogue data into a digital format. The digital data can then be played back through a loudspeaker once it has been converted back to electrical signals (see Chapter 3 for more details).

To convert sound to digital, the sound waves must be sampled at regular time intervals. The amplitude (loudness) of the sound uses a number of bits to represent the range (for example, 0 to 15 bits). The greater the number of bits used to represent the amplitude, the greater the accuracy of the sampled sound. The number of bits per sample is called the **sampling resolution**; the **sampling rate** is the number of sound samples taken per second. Look at these two diagrams to show the difference.

In the first diagram, only 8 bits (0 to 7) are used to represent the amplitude, whereas 16 bits are used in the second diagram. This means the second diagram allows 16 distinct values to represent amplitude, whereas the first diagram only has eight values to represent the same amplitude range.



1.2.3 Representation of (bitmap) images

Bitmap images are made up of **pixels** (picture elements). An image is made up of a two-dimensional matrix of pixels. Each pixel can be represented as a binary number, so bitmap images are stored as a series of binary numbers, so that:

- a black and white image only requires 1 bit per pixel (1 = white, 0 = black)
- if each pixel is represented by 2 bits, there are 2^2 (= 4) possible values (00, 01, 10 and 11) – therefore, four colours could be represented or four shades of grey
- if each pixel is represented by 3 bits, there are 2^3 (= 8) possible values – therefore, eight colours could be represented or eight shades of grey; and so on.

The number of bits to represent each possible colour is called the **colour depth**. **Image resolution** refers to the number of pixels that make up an image, for example 4096×3072 (= 12 582 912) pixels could be used to make up an image. Each pixel will be represented by a number of bits (for example, a colour depth of 32 bits).

Tip

As colour depth and/or resolution increase, the quality of the image will improve; but this also causes an increase in file size which impacts on the storage/memory requirements.

1.3 Data storage and file compression

REVISED

1.3.1 Measurement of data storage

Recall that a **bit** refers to each binary digit and is the smallest unit; four bits make up a nibble (an old unit) and eight bits make up a byte. Memory size and storage size are both measured in terms of bytes

Data storage and memory is measured in terms of bytes:

- 1 KiB (kibibyte) = 2^{10} bytes
- 1 MiB (mebibyte) = 2^{20} bytes
- 1 GiB (gibibyte) = 2^{30} bytes
- 1 TiB (tebibyte) = 2^{40} bytes
- 1 PiB (pebibyte) = 2^{50} bytes
- 1 EiB (exbibyte) = 2^{60} bytes

Tip

Remember that answers must be given in the units specified by the question.

1.3.2 Calculation of file size

The file size of an image is calculated by:

image resolution (number of pixels) \times colour depth (in bits)

For example, a photograph is taken by a camera that uses a colour depth of 32 bits; the photograph is 1024×1080 pixels in size. We can work out the file size as follows:

$$1024 \times 1080 \times 32 = 35\,389\,440 \text{ bits} \equiv 4\,423\,680 \text{ bytes} \equiv 4.22 \text{ MiB}$$

The file size of a sound file is calculated by:

sample rate (in Hz) \times sample resolution (bits) \times length of sample (secs)

For example, an audio file which is 60 minutes in length uses a sample rate of 44 100 and a sample resolution of 16 bits. We can work out the file size as follows:

$$44\,100 \times 16 \times (60 \times 60) = 25\,401\,600\,000 \text{ bits} \equiv 317\,520\,000 \text{ bytes} \equiv 302.8 \text{ MiB}$$

1.3.3 Data compression

Files are often **compressed** to save storage used, reduce streaming and downloading/uploading times, reduce the **bandwidth** requirements and reduce costs (for example, if storing files using cloud storage).

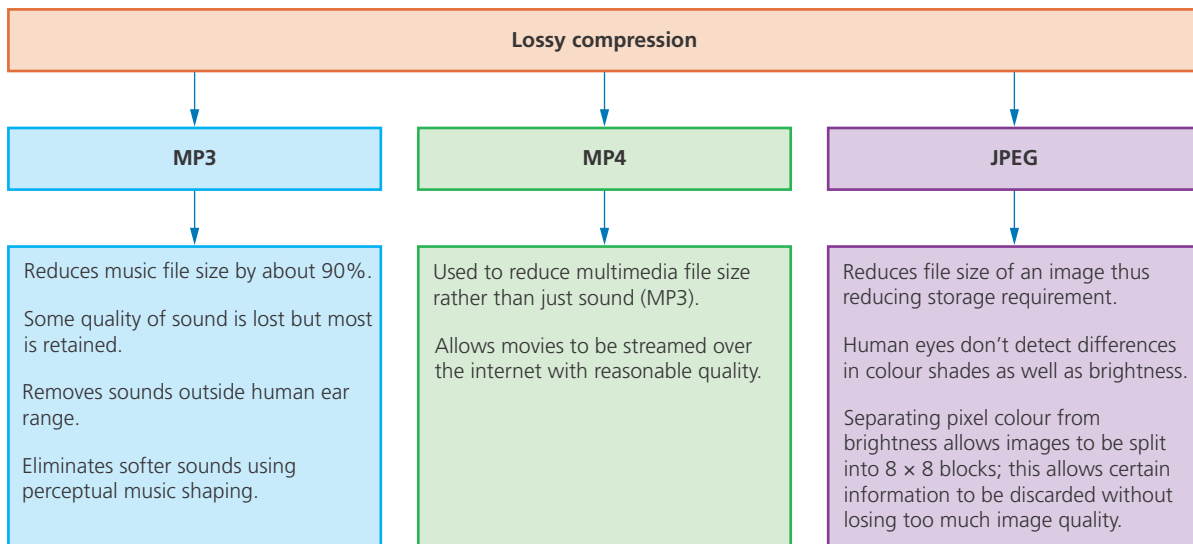
1.3.4 Lossy and lossless file compression

Two common types of (file) compression are **lossy** and **lossless**.

Lossy	Lossless
<ul style="list-style-type: none"> File compression algorithms eliminate unnecessary data. The original file cannot be reconstructed once it has been compressed. The files are smaller than those produced by lossless algorithms. Examples include MPEG and JPEG. 	<ul style="list-style-type: none"> Data from the original uncompressed file can be reconstructed following compression. No data is lost following the application of the lossless algorithms. Most common example is RLE.

Lossy file compression

Examples of lossy file compression include the following.



Lossless file compression

Run length encoding (RLE) is an example of lossless compression. It works by:

- reducing the size of a string of adjacent, identical data items
- the repeating unit is encoded into two values:
 - first value represents number of identical data items
 - second value represents code (such as ASCII) of data item.

Using RLE on text data

For example aaaaaaa/bbbbbbbbbb/c/d/c/d/c/d/eeeeeeee becomes:

255 08 97 // 255 10 98 // 99 /100 /99 /100 /99 /100 // 255 08 101

This is the ASCII code of the repeating unit

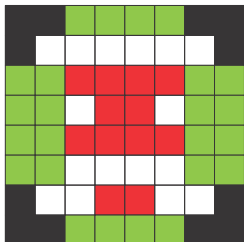
This is the number of times the character unit is repeated





255 is a flag indicating that the two values that follow are the number of repeating units and the ASCII code of the repeating unit

Using RLE with images

This example shows how the file size of a colour image can be reduced using RLE.

The figure below shows an object in four colours. Each colour is made up of red, green and blue (RGB) according to the code on the right.



Square colour	Components		
	Red	Green	Blue
	0	0	0
	255	255	255
	0	255	0
	255	0	0

This produces the following data:

```
2 0 0 0 4 0 255 0 3 0 0 0 6 255 255 255 1 0 0 0 2 0 255 0 4 255 0 0 4 0
255 0 1 255 255 255 2 255 0 0 1 255 255 255 4 0 255 0 4 255 0 0 4 0 255
0 4 255 255 255 2 0 255 0 1 0 0 0 2 255 255 255 2 255 0 0 2 255 255 255
3 0 0 0 4 0 255 0 2 0 0 0
```

The original image (8×8 square) would need three bytes per square (to include all three RGB values). Therefore, the uncompressed file for this image is:

$$8 \times 8 \times 3 = 192 \text{ bytes.}$$

The RLE code has 92 values, which means the compressed file will be 92 bytes in size. This gives a file reduction of about 52%. It should be noted that the file reductions in reality will not be as large as this due to other data which needs to be stored with the compressed file (for example, a file header).

Sample questions and answers

REVISED

- a) i) Explain the two terms **lossy** and **lossless** file compression. [2]
 ii) Give **two** advantages of compressing files and data. [2]
 iii) Give **one** drawback of using lossy file compression and **one** drawback of using lossless file compression. [2]
 b) A camera detector has an array of 2048 by 3072 pixels and uses a colour depth of 32 bits. The camera has a 64 GiB memory capacity. Calculate how many typical images could be stored on the camera. [3]

Tips

Since the first part is an 'explain' question, it is necessary to give a detailed explanation of the two terms mentioned in the question. Parts a) ii) and a)iii) just require a brief description since you are asked to give examples. Do not elaborate too much here since it will simply waste time without any gain in marks. Part b) is a calculation, so it is vital that you show **every** step in your calculation to show your logic and gain credit if your final answer is incorrect.

Sample high-level answer

- a) i) With lossy file compression, the file compression algorithms eliminate all unnecessary data and the original file can no longer be reconstructed; some data is irretrievably lost. The resultant files are much smaller than the original files. Examples include MPEG and JPEG.
 With lossless file compression, data from the original uncompressed file can be reconstructed following application of the lossless compression algorithms. No data is lost following the application of the lossless compression algorithms. A typical example is run length encoding (RLE).
 ii) Two advantages include: reduction in storage space used to store the files, faster download/upload of files across networks since they are much smaller. It is less expensive to store the files if cloud storage is used.

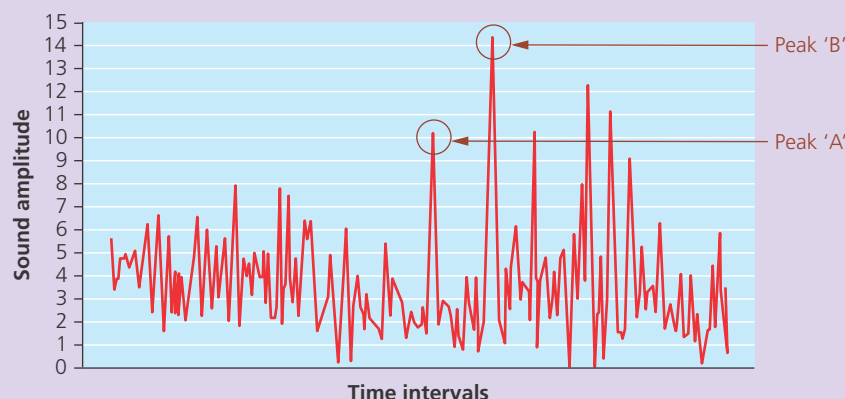
- iii) One drawback of lossy file compression is that data is permanently lost so that the original file cannot be reconstructed. One drawback of lossless file compression is that the compressed files are still larger than those created from lossy compression.
- b) number of bits = $2048 \times 3072 \times 32 = 201\,326\,592$ bits
 divide by 8 to convert to bytes = $25\,165\,824$ bytes
 camera memory size in bytes = $64 \times 1024 \times 1024 \times 1024 = 68\,719\,476\,736$
 number of images = $(68\,719\,476\,736) \div (25\,165\,824) = 2730$ images

Sample low-level answer

- a) i) lossy means data is lost permanently when a file is compressed whereas lossless doesn't lose any of the data for ever.
 ii) uses up less space and it is faster and easier to send files over the internet
 iii) lossy – lose data
 lossless – more complicated compression algorithm
- b) 2048×3072 pixels = $6\,291\,456$ bytes
 Number of images = $(6\,292\,456) \div 64 = 98\,304$ images stored.

Exam-style questions

- 6 The following diagram shows the sampling of a sound source:

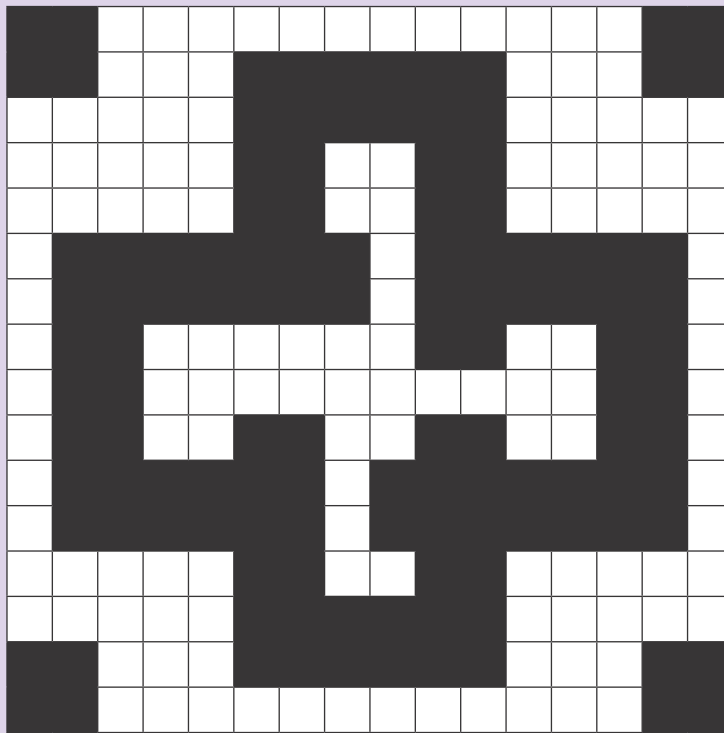


- a) What type of data is natural sound? [1]
 b) The sound resolution being used is 16 bits. Write down the binary values of
 i) peak 'A' and peak 'B';
 ii) peak 'B' has two potential values; describe how this problem could be resolved. [3]
 c) Explain what is meant by:
 i) sampling resolution
 ii) sampling rate. [2]
 d) A sound source is being sampled at 20 000 samples per second. The sampling resolution used is 32 bits.
 i) Give **one** advantage of using 32 bits rather than 8 bits.
 ii) Give **one** disadvantage of using 32 bits rather than 8 bits.
 iii) Estimate the file size if a 30 second sound sample was being recorded.
 Give your answer in MiB. [4]

Teacher's comments

The first answer is a comprehensive description of lossy and lossless data compression and would probably gain full marks. The calculation shows all of the steps, which is a good exam technique – if you make any errors in your calculation, by showing all steps, you could still gain a good mark even if your final answer is incorrect.

The second answer would probably gain one mark for the reference to lossy files losing data permanently in contrast to lossless. It would also gain one mark for a)ii) for reference to transfer of files over the internet. However, the statement 'uses up less space' won't get any marks – it needs to refer to storage space or memory space (the word 'space' on its own is worth 0 marks). The first answer to part a) iii) is just a repeat of a) i) and the second answer isn't necessarily true. The calculation in part b) is a mess but they would probably gain a mark for the calculation of number of bits; but the rest of their answer is confused or simply incorrect.



- 7 An ancient Roman mosaic was being scanned by archaeologists and the pattern saved on a computer. Each black tile has the binary value 0 and each white tile has the binary value 1.
- What is the file size of the raw data? [1]
 - Describe how run length encoding (RLE) could be used to reduce the size of the raw file. Your answer should include calculations to show how the file size could be reduced. [6]
- 8 Explain the following four terms:
- MP3 file
 - JPEG file
 - 1 Terabyte
 - pixel [8]
- 9 A student gave the following answers in an end-of-term test on computer science. In each case, explain why the student's answers were incorrect. Also suggest what answer the student should have given in each case.
- 1 MiB is equivalent to 1 000 000 bytes of data.
 - A file which undergoes a lossy compression algorithm can be restored to its original file whenever required.
 - RLE works effectively on any run of repeating units.
 - ASCII code is a subset of Unicode.
 - To play back a sound file stored on a computer through a set of external speakers requires the use of an analogue to digital converter (ADC). [10]
- 10 When zooming in to an image (that is, increasing its size on screen), it may become pixelated.
- What is meant by the term **pixelated**? [2]
 - Explain why an image can become pixelated. [2]
 - Colour images are made up of red, green and blue elements. Each of the three colour elements can be represented by 256 bit combinations (from 0 to 255, where 0 = colour not present and 255 = maximum colour element present).
 - What is the hexadecimal equivalent of 255? [4]
 - Describe how it is possible to represent millions of different colours using red, green and blue elements. (You may assume, for example, that pure red is represented by # FF 00 00).

2

Data transmission

Key objectives

The objectives of this chapter are to revise:

- methods of data transmission:
 - data packets
 - structure of data packets
 - packet switching
 - serial, parallel, simplex, half-duplex and full-duplex
 - universal serial bus (USB)
- error detection to include parity checks, checksum, echo checks and Automatic Repeat Requests (ARQ)
- encryption to include symmetric and asymmetric encryption

2.1 Types and methods of data transmission

REVISED

2.1.1 Data packets

Data is broken up into **packets** before it is transmitted. Data packets are split up into:

- **packet header** (containing the IP address of the sending station and receiving station, the sequence number of the packet, so it can be reassembled, and packet size, to ensure the receiving station knows that the whole packet has been received)
- **payload** (the actual data)
- **packet trailer** (containing **cyclic redundancy check (CRC)** (error check) and a way of identifying the end of the data packet).

CRC is found by adding 1-bits and send as a hex value

Data packets allow data to be sent in manageable chunks that can be sent along the most efficient route from A to B. **Routers** (known as **nodes**) are used to control the path a data packet takes from sending station to receiving station. This is called **packet switching** where each data packet can take a different route; each route taken is independent of each other. Since data packets take different routes, they could arrive at their destination in the wrong order. A sequence number in the packet header allows all the data packets to be reassembled in the right order at the receiving station. **Hop numbers** are added to packet headers to ensure they don't 'bounce around' from router to router and eventually become effectively lost.

Benefits of packet switching	Drawbacks of packet switching
<ul style="list-style-type: none">● There is no need to tie up a single communication line.● It is possible to overcome failed, busy or faulty lines by simply re-routing packets.● It is relatively easy to expand package usage.● A high data transmission rate is possible.	<ul style="list-style-type: none">● Packets can be lost and need to be re-sent.● The method doesn't work well with real-time streaming (for example, a live sporting event being transmitted over the internet).● There is a delay at the destination whilst the packets are being re-ordered.

It is illegal to photocopy this page

Sample questions and answers

REVISED

A video conference is taking place between delegates in the USA and Mauritius. Packet switching is used to send video and sound data between delegates.

Describe:

- a) the potential problems with sound and video quality
- b) how the problems in part a) could be overcome.

[6]

Sample high-level answer

- a) Video conferencing is in real time. Video and audio data is split up into data packets before it is sent over the internet. This means that each data packet could potentially follow a different route from delegate to delegate. Data packets could therefore take different times from sender to recipient, or some may even become 'lost'. The arriving packets also need to be reassembled into the correct sequence, which can also cause a time delay. All of this could lead to gaps in data received, freezing of images, drop out or voice and mouth 'out of synchronisation' – this would be caused by audio and video data taking different times from sender to receiver.
- b) Since the problems seem to be due to timing issues, the best solution would be to eliminate these time differences. A single path for all data packets would resolve this; that could be achieved using a dedicated communication link between delegates.

A mention of circuit switching could also be accepted here if a candidate has gone outside the syllabus.

Tips

Since this is a 'describe' question, it is necessary to give all the main facts, features and/or characteristics about video conferencing when using packet switching. It is important to notice the reference to part a) in part b) – this means that your answer to part b) must refer back to potential problems identified in part a).

Teacher's comments

The first answer shows that the student has realised the connection between video conferencing and packet switching. They have correctly referred to 'lost' packets or packets out of order as a reason why there can be sound and vision issues. Their solution of using circuit switching (a dedicated route) is the kind of answer you would expect from a top-level answer (it goes slightly beyond the syllabus, but would still gain credit).

The second answer hasn't made the link between video conferencing and packet switching. But they have mentioned some of the problems that can occur without mentioning how or why they occur. The answer would gain two of the available marks in part a). One mark could also be awarded in part b) for suggesting faster internet could go some way to resolving the issue.

Sample low-level answer

- a) Sometimes you get echo or lag on the sound or it may even drop out on occasions. The video and sound don't always synchronise where a person's lips don't match with the words heard. Sometimes the image just freezes.
- b) This can be overcome using better cameras and microphones and getting a faster internet connection.

Exam-style questions

- 1 Explain how packet switching could be used to download some software from a website. [4]
- 2 Data being shared between two computers is split up into a number of data packets prior to transmission. The data packets contain a header and a trailer.
 - a) Give **three** pieces of information found in the header. [3]
 - b) The trailer contains a cyclic redundancy check (CRC).
 - i) Name another item found in the trailer. [1]
 - ii) The following data is being sent in the payload:
11110000 00111111 00110011 11001100 11111110 11101110
00001100 11011111 11001001
Use this data to show how CRC is used to ensure no errors occurred during data transmission. [3]
 - c) i) Explain how it is possible for data packets to be lost during their transmission across a network. [2]
 - ii) Describe how it is possible to deal with lost packets so that they don't cause network problems, such as 'clogging up' the system. [2]

2.1.2 Data transmission

The modes of data transmission are:

- **simplex**: data can be sent in one direction only (for example, sending data to a printer)
- **half-duplex**: data can be sent in both directions, but not at the same time (for example, using a walkie-talkie to send and receive messages)
- **full-duplex**: data can be sent in both directions at the same time (for example, when using a broadband internet connection).

Serial and parallel data transmission

- **Serial data transmission**: data is sent one bit at a time down a single wire/channel (for example, using a USB connection).
- **Parallel data transmission**: several bits of data are sent down several wires/channels at the same time – each wire/channel transmits each bit (for example, transmitting data using the internal circuits of a computer).

Take care, do not use the word 'cable' instead of wire.

It is important to remember that serial transmission and parallel transmission can use serial, half-duplex or full-duplex as a method of data transmission.

The table below shows the comparison between serial and parallel data transmission.

Features of serial transmission	Features of parallel transmission
<ul style="list-style-type: none"> ● Less risk of external interference than with parallel. ● More reliable transmission over longer distances. ● Transmitted bits won't have the risk of being skewed. ● Used if the amount of data being sent is relatively small, since transmission rate is slower than parallel. ● Used to send data over long distances. ● Less expensive than parallel due to fewer hardware requirements. 	<ul style="list-style-type: none"> ● Faster rate of data transmission than serial, which makes it the preferred method where speed is important (such as internal connections in a computer). ● Works well over shorter distances. ● Due to several wires/channels being used, data can become skewed over long distances (no longer synchronised). ● Easier to program input/output operations when parallel used. ● Preferred method when sending large amounts of data. ● The most appropriate transmission method if data is time-sensitive. ● Requires more hardware, making it more expensive to implement than serial ports.

2.1.3 Universal serial bus

The **universal serial bus (USB-A)** is a form of serial data transmission. It is the industry standard. When a USB is used to connect a device to a computer:

- the computer automatically detects the device
- the device is automatically recognised and the appropriate device driver is loaded.

The following table considers the benefits and drawbacks of the USB system.

USB benefits	USB drawbacks
<ul style="list-style-type: none"> ● Devices plugged into the computer are automatically detected and device drivers are automatically loaded up. ● Connections can only fit one way preventing incorrect connections being made. ● Has become an industry standard. ● Can support different data transmission rates. ● No need for external power source since cable supplies +5V power. ● USB protocol notifies the transmitter to re-transmit data if any errors are detected. ● Relatively easy to add more USB ports if necessary by using USB hubs. ● Backward compatible. 	<ul style="list-style-type: none"> ● Standard USB only supports a maximum cable length of 5 m; beyond that, USB hubs are needed to extend the cable length. ● Even though USB is backward compatible, very early USB standards (V1) may not always be supported by the latest computers. ● Even the latest version 3 (V3) and version 4 (V4) USB-C systems have a data transfer rate which is slow compared with, for example, Ethernet connections.

USB-C is the latest type of USB connector, which uses a totally symmetrical 24-pin USB connection, so it will fit into a USB port either way round since there is no up or down orientation. USB-C is actually USB 3.1 but has been named USB-C to distinguish it from the old format which was known as USB-A.

The main advantages (compared to existing USB-A) of the USB-C connector are:

- it is much smaller and thinner which suits the current trend in making devices much thinner
- it offers 100 watt (20 volt) power connectivity which means full-sized devices can now be charged
- it can carry data at 10 Gigabits per second (10Gbps); this means it can now support 4K video delivery
- the USB-C connection is backward compatible (to USB 2.0 and 3.0) provided a suitable adaptor is used.

Sample question and answer

REVISED

Compare the USB-A and USB-C types of connection.

[4]

Sample high-level answer

Both types of connector are backward compatible and have become the industry standard. Since USB cables also supply power, there is no need for an external power supply. It is easy to add additional USB-A and USB-C ports by using a USB hub.

The USB-A connector can only fit one way round, whereas the USB-C connector is non-orientated. Data transfer rate using USB-C is considerably faster than for USB-A. Although both types of connector supply power, the USB-C allows 20 V power connectivity which means it can be used to charge much larger devices.

Tip

Since this is a 'compare' question, it is necessary to consider all the similarities and all the differences between the two types of connector.

Sample low-level answer

USB-C is the newer type of USB connection. It is used by some devices like mobile phones and laptops. Its advantage is the cable works any way round since both ends are the same. Data access is much faster than with USB-A.

Teacher's comments

The first answer contains four distinct points citing the similarities and differences between USB-A and USB-C. This gives the answer a good balance.

The second answer is much briefer even though four points were made.

- The first point, although true, is not worth a mark.
- The second point is true, but it is not a comparison as asked for in the question.
- The answer confused a symmetrical connection that allows the USB-C to fit into a device any way round with having the same connector on both ends of the cable – which is incorrect.
- Data access isn't faster, but data transfer is faster.

Overall, no marks were gained even though four points were made.

Exam-style questions

3 Explain what is meant by the following **four** terms:

- a) Data skewing
- b) USB-C connection
- c) Parallel data transmission
- d) Half-duplex data transmission

[8]

4 Four statements about data transmission are shown in the following table.

Tick (✓) the appropriate columns to indicate the method and direction of data transmission being described by each statement.

[4]

Statement	Transmission method		Direction of data transmission		
	Serial	Parallel	Simplex	Half-duplex	Full-duplex
Data is being sent in both directions, one bit at a time along a single wire, but not at the same time.					
16 bits of data are being sent along 16 individual channels in both directions simultaneously.					
Data is being sent 8 bits at a time down eight wires in one direction only.					
Data is being sent one bit at a time down a single wire; the transmission occurs in both directions simultaneously.					

2.2 Methods of error detection

REVISED

2.2.1 The need to check for errors

When data is transferred there is always a risk that the data may be corrupted, lost or even gained. Errors can occur during data transmission due to:

- electrical interference – can corrupt data
- packet switching – can lead to data being lost or out of synchronisation
- skewing of data – bits arrive at their destination no longer synchronised.

The next section considers methods used to check for errors following data transmission.

2.2.2 Parity checks, checksum and echo checks

Parity checks

A **parity check** is a method used to check whether data has been changed or corrupted following data transmission. Parity can be even (even number of 1-bits) or odd (odd number of 1-bits). The left-most bit is reserved for a **parity bit**.

If two bits have been changed (for example, 11110000 changed to 00111100) the parity may stay the same and wouldn't be picked up by a parity check. In such cases, another error detection method, such as a checksum, needs to be carried out.

Parity blocks can be used to determine exactly which bit has been corrupted or changed following data transmission.

Worked example

In this example, nine bytes of data have been transmitted. Another byte, known as the parity byte, has also been sent (this byte consists entirely of the parity bits produced by the vertical parity check and also indicates the end of the block of data).

Even parity is being used. The table shows how the data arrived at the receiving end. And each row and column needs to be checked to see if they still show even parity.

	Parity bit	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
Byte 1	1	1	1	1	0	1	1	0
Byte 2	1	0	0	1	0	1	0	1
Byte 3	0	1	1	1	1	1	1	0
Byte 4	1	0	0	0	0	0	1	0
Byte 5	0	1	1	0	1	0	0	1
Byte 6	1	0	0	0	1	0	0	0
Byte 7	1	0	1	0	1	1	1	1
Byte 8	0	0	0	1	1	0	1	0
Byte 9	0	0	0	1	0	0	1	0
Parity byte	1	1	0	1	0	0	0	1

The table shows:

- byte 8 (row 8) now has incorrect parity (there are three 1-bits)
- bit 5 (column 5) also now has incorrect parity (there are five 1-bits).

First, the table shows that an error has occurred following data transmission (there has been a change in parity in one of the bytes).

Second, at the intersection of row 8 and column 5, the position of the incorrect bit value (which caused the error) can be found. The 1-bit at this intersection should be a 0-bit; this means that byte 8 should have been:

0	0	0	1	0	0	1	0
---	---	---	---	---	---	---	---

which would also correct column five, giving an even vertical parity (now has four 1-bits).

Checksum

A **checksum** is another method used to check if data has been changed/corrupted following data transmission. The checksum is calculated by the sending computer, from the block of data using an agreed algorithm. The data is then sent as a block along with the checksum. The checksum is re-calculated by the receiving computer using the same algorithm used by the sending computer. Any differences in the checksum indicates an error.

Echo check

An **echo check** requires data to be sent back to the sending computer where it is compared with the data originally sent. Any errors and the data is re-sent. This is not a very reliable method; if the two sets of data are different, it is not known whether the error occurred when sending the data originally or if the error occurred when sending the data back for checking.

Enter the exam with the confidence to achieve your best grade using a concise summary of the syllabus content and step-by-step guidance in exam technique.

- » Benefit from expert advice and tips on skills and knowledge from experienced subject authors.
- » Target revision and focus on important concepts and skills with key objectives at the beginning of every chapter.
- » Keep track of your own progress with a handy revision planner.
- » Consolidate and apply your understanding of key content with exam-style questions.
- » Apply your understanding of theoretical content and practical skills with sample practice papers, written by the authors, at the end of the book and online.

This title has not been through the Cambridge Assessment International Education endorsement process.

Also available for
Cambridge IGCSE™
and **O Level Computer Science:**



For over 30 years we have been trusted by Cambridge schools around the world to provide quality support for teaching and learning.

For this reason we have been selected by Cambridge Assessment International Education as an official publisher of endorsed material for their syllabuses.



Boost

This title is also available as an **eBook** with **learning support**.

Visit hoddereducation.com/boost to find out more.

HODDER EDUCATION

e: education@hachette.co.uk

w: hoddereducation.com

ISBN 978-1-3983-1848-9

