Edexcel

OL IGCSE

Computer science

CODE: (4CP0)

Unit 03

Data

Chapter 3.12 - Binary

Binary is a base-2 numeral system using only two digits: 0 and 1. It is a positional notation where digits have place values, like the denary or decimal system that we are familiar with.

Why binary?

Transistors function as switches with two states: on or off, transmitting electric current. **Digital** systems have separate states, while binary systems have two states. **Analogue** systems have continuous ranges between values, unlike digital systems with separate states.

Representing information

When we write, we combine the letters in our alphabet to create words and combine these into sentences to express meaning. In a similar way, the two **binary digits (bits**) can be combined into groups.

Number systems

Binary is a number system based on two digits, 0 and 1.
We are more used to the denary system which has ten digits, 0 to 9, but the binary system functions in the same way. The difference is that the denary system works in powers of 10, while the binary system uses powers of 2.

SUBJECT VOCABULARY

binary information represented by only two values (e.g. a voltage or no voltage; on or off). There are no communication errors or misunderstandings because there are no small differences

digital information represented by certain fixed values (e.g. high, medium or low). Any signal between these values would be meaningless and not used. Sending and receiving systems do not have to be as accurate as for analogue communication

analogue using signals or information represented by a quantity (e.g. an electric voltage or current) that is continuously variable. Changes in the information being represented are indicated by changes in voltage. This method requires very accurate sending and receiving systems

SUBJECT VOCABULARY

binary digit (bit) the smallest unit of data that is represented in a computer. It has a single binary value, either 1 or 0

Denary system

Every digit has a place value, and the one to the left has a value 10 times higher than the one to the right.
This table shows the place values for the digits in the number 3639.

PLACE VALUES	10^3	10^2	10^1	10^0
	1000	100	10	1
	3	6	3	9

Therefore, 3639 can be written as:
(3 * 1000) + (6 * 100) + (3 * 10) + (9 * 1)

SUBJECT VOCABULARY

byte the basic combination of bits used to represent an item of information. A byte typically consists of 8 bits

Binary system

The binary system has similar place values, but they increase by powers of 2.
The table shows the place values for the digits in a **byte**.

PLACE VALUES	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1

Converting binary to denary

This table can be used to convert a binary number to a denary one.
The denary equivalent of the binary number 10101101 can be calculated as shown in the table below:

PLACE VALUES	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
	1	0	1	0	1	1	0	1

(1 * 128) + (1 * 32) + (1 * 8) + (1 * 4) + (1 * 1) = 128 + 32 + 8 + 4 + 1 = 173

Converting denary to binary

The table of place values can be used to convert denary numbers to binary.

WORKED EXAMPLE

Convert denary 213 to binary. Use the table and fill it in as you calculate the equivalent.

PLACE VALUES	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1

Move from left to right:

Step 1: is the number (213) equal to or greater than 128? If it is, place a 1 at this place value and if not, place a 0.

In this case, 213 is greater than 128 so place a 1 in this column and calculate the remainder.

PLACE VALUES	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
	1							

The remainder is 85.

Step 2: is 85 greater than or equal to 64? Yes, therefore place a 1 in this column.

PLACE VALUES	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
	1	1						

The remainder is 21.

Step 3: is 21 greater than or equal to 32. No, therefore place a 0 in this column.

PLACE VALUES	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
	1	1	0					

The remainder is still 21.

Step 4: is 21 equal to or greater than 16? Yes, therefore place a 1 in this column.

PLACE VALUES	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
	1	1	0	1				

The remainder is 5.

Step 5: is 5 equal to or greater than 8? No, therefore place a 0 in this column.

PLACE VALUES	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
	1	1	0	1	0			

The remainder is still 5.

Step 6: is 5 equal to or greater than 4? Yes, therefore place a 1 in this column.

PLACE VALUES	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
	1	1	0	1	0	1		

The remainder is 1.

Step 7: is 1 equal to or greater than 2? No, therefore place a 0 in this column.

PLACE VALUES	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
	1	1	0	1	0	1	0	

The remainder is still 1.

Step 8: is 1 equal to or greater than 1? Obviously, they are equal and therefore place a 1 in the last column.

PLACE VALUES	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
	1	1	0	1	0	1	0	1

Therefore, denary 213 is represented by binary 11010101.

Binary arithmetic

Binary numbers can be manipulated in the same way as denary ones.

DENARY ADDITION

When addition is performed in denary, a 'carry over' is used if the result is greater than 9.

SUBJECT VOCABULARY

overflow error this condition occurs when a calculation produces a result that is greater than the computer can deal with or store. When this happens, the microprocessor is informed that an error has occurred

BINARY ADDITION

Binary addition works in the same way, but a 'carry over' is needed if the result is greater than 1.

An overflow error occurs when the last 1 in a calculation is carried out, causing the calculation to be incorrect due to the overflow of bits.

Signed and unsigned numbers

So far, we have been considering whole numbers or integers that are positive, but how can we represent negative numbers? When an integer is indicated as being positive or negative, it is described as signed.

SIGN AND MAGNITUDE

In a multiple-bit binary number, the left-most bit (the one with the greatest value) is called the **most significant bit (MSB)**. We can use this to represent signed integers.

SUBJECT VOCABULARY

integer a whole number (e.g. 3, 6, 9)

most significant bit (MSB) the bit with the highest value in a multiple-bit binary number

Adding signed integers

If the binary equivalents of denary +28 and denary -28 are added, we get the following result.

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

Adding denary +28 and -28 we expect the result to be 0, but the binary result is equivalent to -56.

Also using this method, 0 could be both positive and negative.

Positive zero	0	0	0	0	0	0	0	0
Negative zero	1	0	0	0	0	0	0	0

That clearly doesn't make sense and so another method is needed to represent negative numbers in calculations.

Two's complement

FINDING THE TWO'S COMPLEMENT OF A BINARY NUMBER

To find the two's complement of a binary number:

- Flip all of the bits - change 1s to Os and Os to 1s
- Add 1 to the result.

Number	0	0	0	1	1	1	0	0
Flip	1	1	1	0	0	0	1	1
Add 1 to flipped value	1	1	1	0	0	1	0	0

CONVERTING TWO'S COMPLEMENT NUMBERS BACK INTO DENARY

METHOD 1-REVERSE CALCULATION

The positive equivalent of a negative integer in two's complement format can be found by flipping the bits and adding 1.

Number –28	1	1	1	0	0	1	0	0
Flip	0	0	0	1	1	0	1	1
Add 1 to flipped value	0	0	0	1	1	1	0	0

The result is denary 28.

This shows that 11100100 (-28) is the two's complement representation of 00011100 (+28)

METHOD 2-PLACE VALUES

Using place values, the two's complement number can be converted directly into denary.

PLACE VALUES	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	–128	64	32	16	8	4	2	1
	1	1	1	0	0	1	0	0

The MSB of the two's complement binary number is 1, indicating that it is negative. Therefore, the denary equivalent of the MSB is taken as being negative -128.

The denary equivalent will be

-128 +64 +32 +4=-128 + 100=-28

This shows that two's complement 11100100 is equivalent to denary -28.

ADDING -28 TO +28

The two's complement of 00011100 is 11100100.

So, adding the binary equivalents of denary +28 and -28 gives:

28	0	0	0	1	1	1	0	0
–28	1	1	1	0	0	1	0	0
Result of addition	0	0	0	0	0	0	0	0

There is an overflow, but this time the result is 0.

Binary shifts

In denary, when we want to multiply a number by 10, we move each number one place value to its left and add a 0 to the end. For example, 13* 10 = 130.

PLACE VALUES	10^3	10^2	10^1	10^0
	1000	100	10	1
Original number			1	3
Multiply by 10. The digits are moved to the left and a 0 is added.		1	3	0

In binary left and right, binary shifts can be used for multiplication and division by powers of 2.

LOGICAL SHIFTS

For unsigned numbers, logical shifts are performed in the same way as in denary.

If we multiply the unsigned binary number 00010100 (20) by 2^2 (4), there should be two shifts to the left.

The left-most bits drop off the end and are replaced by Os at the right.

PLACE VALUES	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
	0	0	0	1	0	1	0	0
Result of shift	0	1	0	1	0	0	0	0

The product is (64 * 1) + (16 * 1) = 80 in denary as expected.

If we divide the same number by 4 (2^2), there would be two shifts to the right. This time the right-most bits drop off the end and are replaced by 0s at the left.

PLACE VALUES	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
	0	0	0	1	0	1	0	0
Result of shift	0	0	0	0	0	1	0	1

The result is (4 * 1) + (1 * 1) = 5 in denary as expected.

A logical shift on integers can make a number less precise. For example, if we divide 00100001 (33 in denary) by 2 the digits would be shifted one place to the right.

PLACE VALUES	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
	0	0	1	0	0	0	0	1
Result of shift	0	0	0	1	0	0	0	0

The result is 00010000, which is 16, but the correct result should be 16.5

ARITHMETIC SHIFTS

Arithmetic shifts are used with signed numbers expressed in two's complement format.

LEFT ARITHMETIC SHIFT

A left arithmetic shift is identical to a left logical shift except that the left-most bit (MSB) is not included because it must remain in place to indicate the sign.

RIGHT ARITHMETIC SHIFT

When dividing signed binary numbers in two's complement format by powers of 2, the bits are shifted to the right. They are replaced at the left by copies of the MSB.

SUBJECT VOCABULARY

binary shift an operation done on all the bits of a binary value in which they are moved by a specific number of places to either the left or right

SUBJECT VOCABULARY

arithmetic shift used for signed binary numbers. When performing a right shift, the bits at the left are replaced by copies of the most significant bit

Hexadecimal numbers

Hexadecimal numbers are commonly used for data and commands, as humans struggle with large binary numbers. They use 16 digits from 0 to 15, unlike the denary system, which uses 0 to 9, representing higher hexadecimal digits by upper case letters.

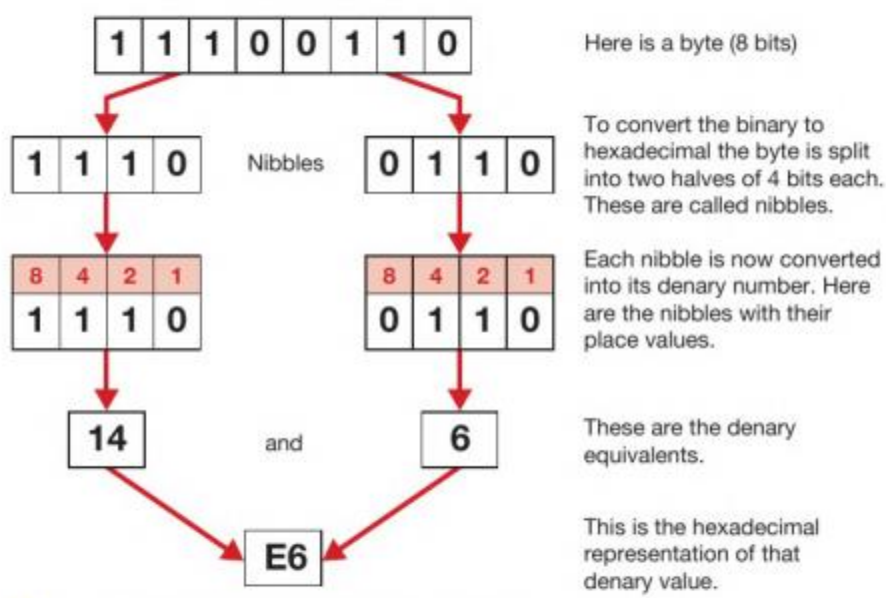
SUBJECT VOCABULARY

hexadecimal a base-16 number system. There are 16 digits and the place values increase in powers of 16

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Converting binary to hexadecimal

The following diagram shows how to convert binary to hexadecimal.

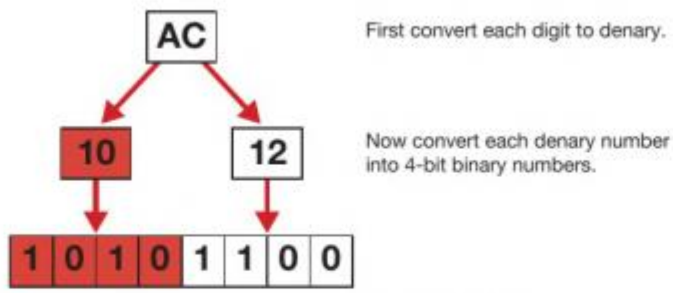


▲ **Figure 3.1** Converting binary numbers to hexadecimal

It is far easier to say and remember 'E6' than '11100110'.

Converting hexadecimal to binary

This is the reverse of the method used above.



▲ Figure 3.2 Converting hexadecimals to binary numbers

Chapter 3. 13

Data representation

Representation of text

A computer's **character set** is the defined list of characters recognized by its hardware and software, including 1s and Os, not characters like 'c', '@', or 'D'.

Originally, the ASCII code consisted of 7 bits and so 128 characters could be represented. All of the lower- and upper-case English characters, punctuation marks and control actions such as backspace, shift on, shift off and carriage return were represented.

The printable characters of the 7-bit ASCII code are shown in Figure 3.3. The codes for control actions (mentioned above) are not shown.

D	B	C	D	B	C	D	B	C	D	B	C
32	00100000	space	57	00111001	9	82	01010010	R	107	01101011	k
33	00100001	!	58	00111010	:	83	01010011	S	108	01101100	l
34	00100010	"	59	00111011	;	84	01010100	T	109	01101101	m
35	00100011	#	60	00111100	<	85	01010101	U	110	01101110	n
36	00100100	$	61	00111101	=	86	01010110	V	111	01101111	o
37	00100101	%	62	00111110	>	87	01010111	W	112	01110000	p
38	00100110	&	63	00111111	?	88	01011000	X	113	01110001	q
39	00100111	'	64	01000000	@	89	01011001	Y	114	01110010	r
40	00101000	(	65	01000001	A	90	01011010	Z	115	01110011	s
41	00101001	)	66	01000010	B	91	01011011	[	116	01110100	t
42	00101010	*	67	01000011	C	92	01011100	\	117	01110101	u
43	00101011	+	68	01000100	D	93	01011101	]	118	01110110	v
44	00101100	,	69	01000101	E	94	01011110	^	119	01110111	w
45	00101101	-	70	01000110	F	95	01011111	_	120	01111000	x
46	00101110	.	71	01000111	G	96	01100000	`	121	01111001	y
47	00101111	/	72	01001000	H	97	01100001	a	122	01111010	z
48	00110000	0	73	01001001	I	98	01100010	b	123	01111011	{
49	00110001	1	74	01001010	J	99	01100011	c	124	01111100	|
50	00110010	2	75	01001011	K	100	01100100	d	125	01111101	}
51	00110011	3	76	01001100	L	101	01100101	e	126	01111110	~
52	00110100	4	77	01001101	M	102	01100110	f	127	01111111	DEL
53	00110101	5	78	01001110	N	103	01100111	g			
54	00110110	6	79	01001111	O	104	01101000	h			
55	00110111	7	80	01010000	P	105	01101001	i			
56	00111000	8	81	01010001	Q	106	01101010	j			

KEY: D = denary, B = binary, C = character

▲ Figure 3.3 The printable characters of the 7-bit ASCII code

Analyzing a string

To a computer, a string is just a stream of binary codes that represent the characters. Programming languages have functions that will return the denary equivalents of those codes and vice versa.

SUBJECT VOCABULARY

encryption the process of turning information (e.g. a message) into a form that only the **intended recipient** can decode, or **decrypt** and read. The message is **encoded** using an agreed method or algorithm. This is called the key. The encrypted message is called a cipher

The Pearson Edexcel pseudocode does not have these functions built in, but in the Python programming language they are `ord()` and `chr()`.

Therefore

- `ord('c')` would return 99, and
- `chr(100)` would return 'd'.

Using the `ord()` function, the ASCII codes of the characters can be found when traversing a string.

By inserting the `ord()` function into the Pearson Edexcel pseudocode, the algorithm that does this can be displayed.

```
SEND "Please enter the text" TO DISPLAY
RECEIVE myString FROM (STRING) KEYBOARD
FOR index FROM 0 TO LENGTH(myString) – 1 DO
   SET number TO ord(myString[index])      #The function is called using the character
                                           in the string as an argument.
   SEND number TO DISPLAY
END FOR
```

By using the `chr()` function you can build up a string using characters entered by their denary codes.

The following algorithm will allow you to enter ten numbers which will be converted to characters and appended to the string.

```
SET myString TO ""                         #This will create an empty string.
FOR index FROM 0 TO 10 DO
   SEND "Please enter a number in the range 65 to 90 or in the range 97 to 122"
TO DISPLAY
   RECEIVE number FROM (INTEGER) KEYBOARD
   SET character TO chr(number)
   SET myString TO myString + character
END FOR
```

Creating a functions

The pseudocode does not have these functions built in. You need to define them using the commands and statements that are available so that you can call them when required.

The `ord()` function needs to take the character as a parameter and return its denary code.

You can do this in the following way.

```
FUNCTION ord(character)
BEGIN FUNCTION
   SET arrayAscii TO [["!"][33], ["""][34], …["z"][122]]   #A two-dimensional array containing all
                                                            of the characters with their denary codes.
   FOR index FROM 0 TO LENGTH(arrayAscii) – 1 DO
      IF character = arrayAscii[index, 0] THEN
         code = arrayAscii[index, 1]
      END IF
   END FOR
RETURN code
END FUNCTION
```

The extended ASCII code

ASCII code, with 96 printable characters, was extended to 8-bit code for 256 codes, accommodating foreign languages, mathematical symbols, and drawing pictures. Manufacturers like IBM and Apple created unique versions with different characters.

Unicode

The Unicode Consortium was established to address ASCII's multiple versions and create a universal standard for text representation. It is used by major manufacturers like IBM, Apple, Microsoft, Oracle, and SunR, and is compatible with ASCII characters 0 to 127.

Representation of BITMAP images

IMAGE SIZE AND RESOLUTION

Image size is usually written as two numbers, for example 640 x 480, 2240 x 1680, or 4064 x 2704.

The first number indicates the number of pixels along the width of the image and the second is the number in the height. Thus a 640 x 480 image is made up of 307 200 pixels and a 4064 x 2704 image comprises 10 989 056 pixels.

The following images are the same size but have different resolutions.

▲ 4288 × 2824 pixels

▲ 100 × 66 pixels

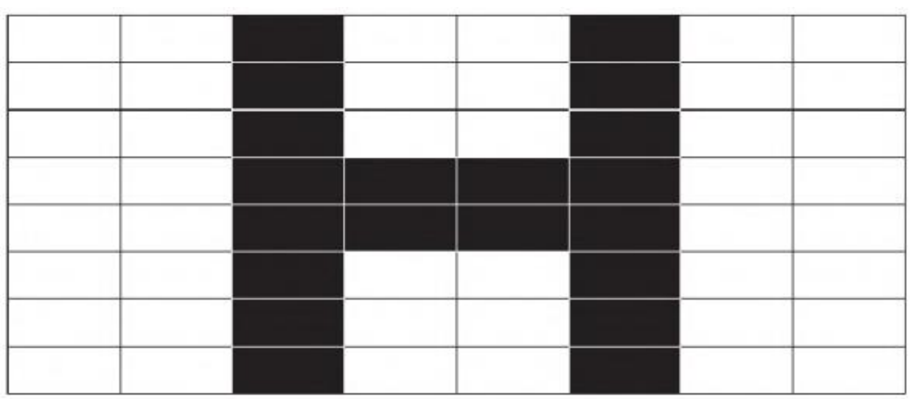
Encoding the pixel information

Figure 3.4 shows an encoding system with 1 bit used to represent each pixel.

.

As the colour depth increases so does the number of colours that can be represented.

- If 1 bit is used, then the number of colours is 21 or 2.
- If 3 bits are used the number of colours is 23 or 8.



▲ **Figure 3.4** 1-bit encoding forming the letter 'H'

- If 8 bits are used the number of colours is 2o or 256.

The current standard represents the colour of each pixel in 24 bits. There are 8 bits for each of the red, blue and green primary colours.

There are therefore 256 variations of each primary colour contributing to the overall colour of the pixel and that gives 256 x 256 x 256 or 16777216 different colours.

File sizes

The file size for a bitmap image is calculated by finding the total number of pixels and multiplying that by the number of bits used to represent each pixel, or:

Width x Height x Colour depth

The file size of the left-hand image on page 125 is:

4288 (width) x 2848 (height) x 24 (bit colour depth) = 293 093 376 bits
That is, 36 636 672 bytes.

Representation of sound

Vibrations in objects like vocal cords or guitar strings create compressed sound waves. These waves reach our ears, triggering sensory hairs in the inner ear, which send nerve impulses to the brain.

▶ **Figure 3.5** Sound waves travelling through the air from a vibrating bell

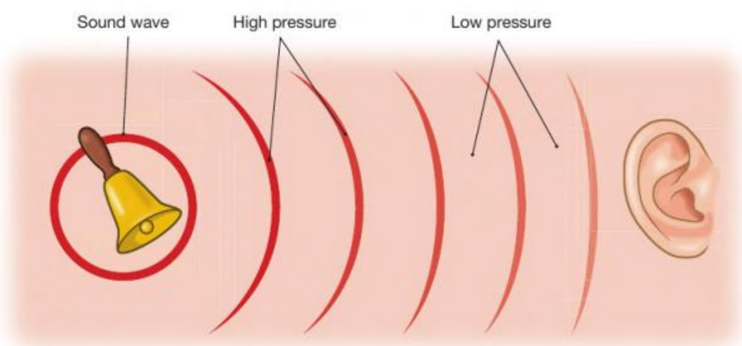
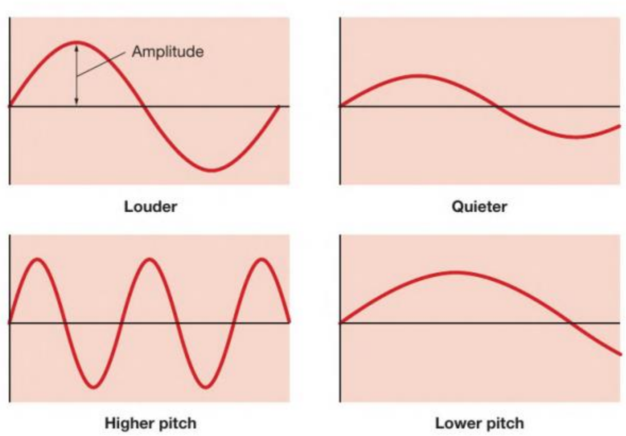


Figure 3.6 illustrates the different waves for loud and quiet and high- and low-pitched sounds.

▶ **Figure 3.6** The bigger the wave, the louder the sound; the greater the frequency of the waves, the higher the pitch



Recording sound waves

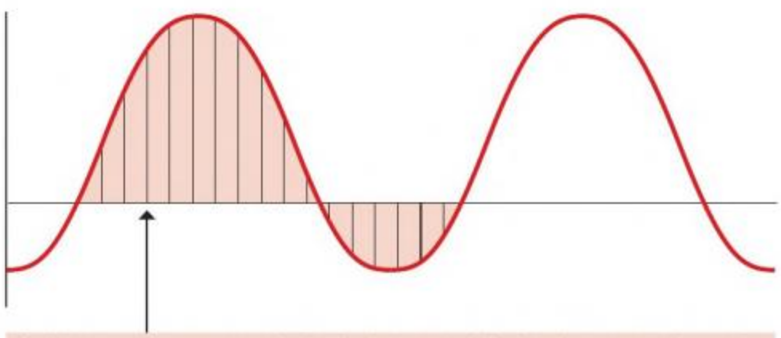
Analogue recording

Analogue recording involves recording the continuous changes in air pressure caused by sound waves, as changes in voltage. These changes are stored in grooves or magnetized areas on records. When played, the stylus' movements convert back into voltage changes, causing the speaker's diaphragm to vibrate, reproducing the original sound.

Digital recording

Digital recording involves capturing sound data as streams of 1s and Os, which cannot represent continuous change. Digital recordings take a series of sound 'snapshots', called samples, which are played back rapidly in succession, creating a continuous sound produced through **sampling.**

Figure 3.7 shows an analogue sound wave being sampled.



Each line represents a new sample. The time between each line/sample represents the sampling period, which equals 1/44 100 of a second, for a CD with a sampling rate of 44.1 kHz.

SUBJECT VOCABULARY

sampling taking measurements of a sound wave at regular but different intervals of time (e.g. 44 100 samples per second)

The complete wave cannot be recorded and so a digital recording can never be as accurate or have the same accuracy, or fidelity, as an analogue recording, but it does have other benefits such as:

- ■ Equipment to record and process digital sound is relatively cheap and has allowed people to record music at home.
- ■It is easily edited using computer equipment.
- ■It is easily copied.
- ■Unlike vinyl and tape, digital files do not physically deteriorate with use, although they may become corrupted if the storage medium is damaged.
- ■Digital files are more portable than records and tapes and can be copied to any digital medium, such as a hard drive, emailed, downloaded and streamed.

FIDELITY

The fidelity of the recording is influenced by two factors.

1. SAMPLE RATE

This is the number of samples taken per second - the higher the rate the higher the fidelity.
Figure 3.8 shows the effect of sampling at different rates.

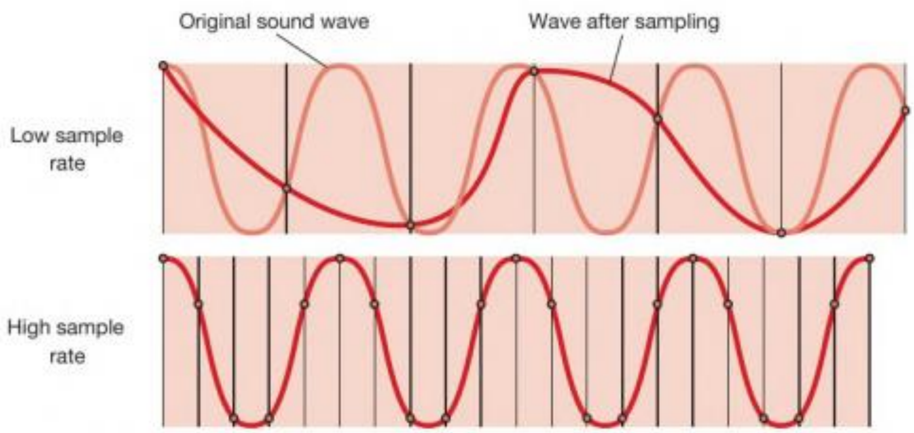
2. HOW PRECISE THE BIT DEPTH OR SAMPLING IS

One important detail is the dynamic range of the sound - the range of volumes of sound in the music.

- ■Using 8 bits allows 256 (2) gradations to be measured.
- ■16 bits allow (21) 65 536.
- ■24 bits allow (22) over 16.7 million.

Using more bits therefore allows for much smaller gradations in the volume differences.

For CD recordings a **bit depth** of 16 bits is used.



▲ **Figure 3.8** The effect of sampling at different rates

SUBJECT VOCABULARY

bit depth the number of bits used to encode the information from each sample. Increasing the number of bits used increases the amount of detail contained in each sample

The size of a digital audio sound file depends on the following:

- Sample rate per second
- Bit depth
- Duration of recording
- Number of channels - mono (one channel) or stereo (two channels).

CALCULATING FILE SIZE

WORKED EXAMPLE

Number of samples per second = 44 100

Bit depth = 16 bits

Duration = 2.5 minutes

Number of channels = 2

The size of the file is found from the following formula:

file size in bits = sample rate * bit depth * duration (in seconds) * number of channels

Therefore, the file size of the above recording is:

44 100 * 16 * 2.5 * 60 * 2 = 211 680 000 bits or 26 460 000 bytes.

Chapter 14

Data storage and compression

data storage

The binary to hexadecimal conversion of **'nibble'** units, equivalent to 4 bits, differs across different operating systems. The **decimal prefix,** used in Ubuntu, treats 'kilo' as 1000, while the **binary prefix** uses powers of 2, resulting in 'kilo' being defined as 210 or 1024, causing differences in file sizes.

SUBJECT VOCABULARY

nibble 4 bits or half a byte

decimal prefix multiplies a unit by powers of 10

binary prefix multiplies a unit by powers of 2

DECIMAL PREFIX				BINARY PREFIX			
UNIT	SYMBOL	MAGNITUDE	SIZE	UNIT	SYMBOL	MAGNITUDE	SIZE
kilobyte	KB	10^3 bytes	1000 bytes	kibibyte	KiB	2^{10} bytes	1024 bytes
megabyte	MB	10^6 bytes	1000 kilobytes	mebibyte	MiB	2^{20} bytes	1024 kibibytes
gigabyte	GB	10^9 bytes	1000 megabytes	gibibyte	GiB	2^{30} bytes	1024 mebibytes
terabyte	TB	10^{12} bytes	1000 gigabytes	trebibyte	TiB	2^{40} bytes	1024 gigibytes

▲ **Table 3.1** Decimal and binary prefixes

Therefore, every higher unit is 1000 times larger than the previous one with decimal prefix, while in binary prefix it is 1024 times larger.

In 1998, to avoid confusion, the International Electrotechnical Commission (IEC) assigned the binary prefix units with new names.

DECIMAL PREFIX	BINARY PREFIX
kilobyte (KB)	kibibyte (KiB)
megabyte (MB)	mebibyte (MiB)
gigabyte (GB)	gibibyte (GiB)
terabyte (TB)	tebibyte (TiB)

▲ **Table 3.2** The International Electrotechnical Commission assigned new names to the binary prefix units

SUBJECT VOCABULARY

compression changing the format of a data file so that the size of the file becomes smaller

bandwidth the amount of data that can be carried on a network in a given period of time

lossless compression compressing a file in such a way that it can be decompressed without any loss of data

lossy compression compression where some of the data is removed; the original file cannot be restored when the lossy file is decompressed

Data compression

Millions of image and audio files are uploaded daily, benefiting from small files for efficient data transfer and storage.

That is why file **compression** is so important. Compressed files use less network **bandwidth** to upload and download. There is less Internet congestion, and it makes possible the streaming of video and audio files.

Compression algorithms are used to make the files as small as possible. There are two types of compression - **lossless** and **lossy compression**.

Lossless compression

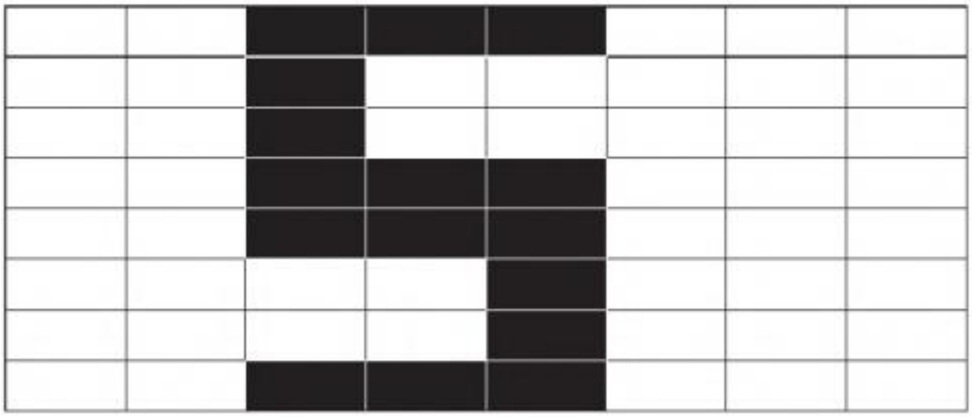
RUN-LENGTH ENCODING (RLE)

Run-length encoding is used to reduce the size of a repeating string of items. The repeating string is called a run and is represented by two bytes - the first byte represents the number of times the item of information is repeated and the second, the item of information.

SUBJECT VOCABULARY

petabyte 1 petabyte is 1000 terabytes or 1 million gigabytes

For a black and white image, like the one in Figure 3.9, compression using RLE would be effective. A colour image in which there are very short runs of different colours would not be encoded as effectively.



▲ **Figure 3.9** A bitmap diagram of the number 5

To demonstrate run-length encoding, the letters 'w' and 'b' are used to represent the white and black pixels instead of 1s and Os

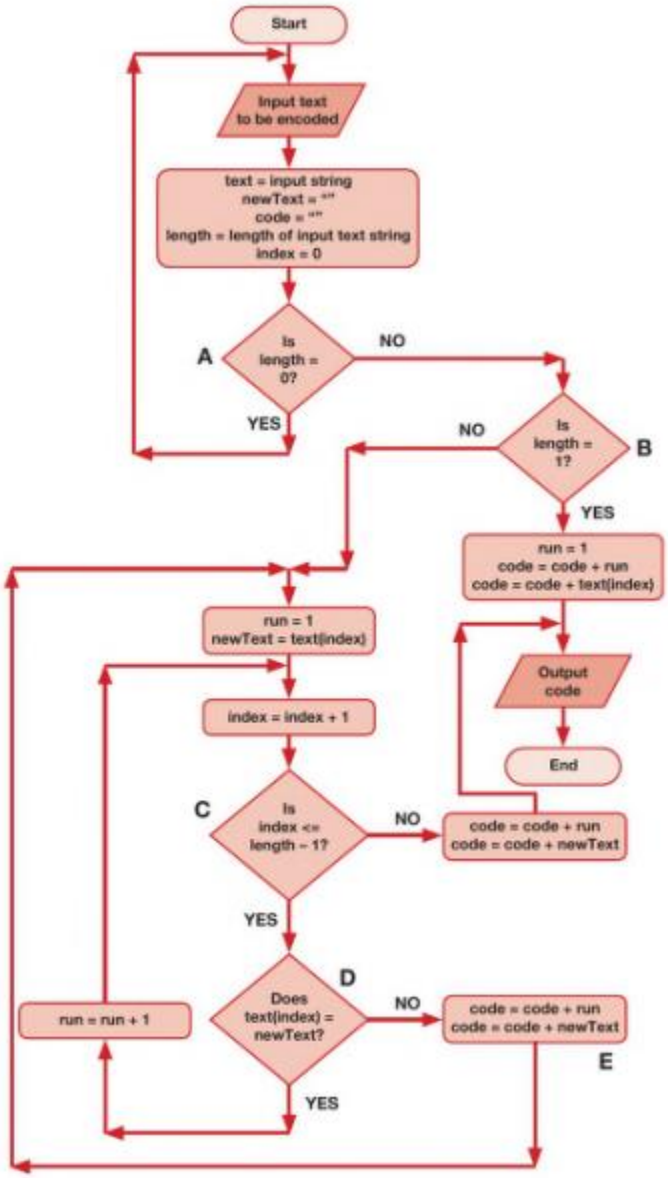
CODE	RLE VERSION	SIZE OF CODED VERSION
wwbbbwww	2w3b3w	6
wwbwwwww	2w1b5w	6
wwbwwwww	2w1b5w	6
wwbbbwww	2w3b3w	6
wwbbbwww	2w3b3w	6
wwwwbwww	4w1b3w	6
wwwwbwww	4w1b3w	6
wwbbbwww	2w3b3w	6
64 bytes		**48 bytes**

▲ **Table 3.3** Run-length encoding of the image of the number 5 in Figure 3.9

Coding the run length algorithm

An RLE algorithm traverses a string and counts the number of times each character is repeated. It starts by checking if the input string has any letters and if it only has one. If there are multiple characters, a loop is used to examine each one. If the character at index 1 is equal to 0, the run length is incremented, and a new run is started for the next character.

▶ Figure 3.11 A flowchart of a possible RLE algorithm



Lossy compression

Bitmap images

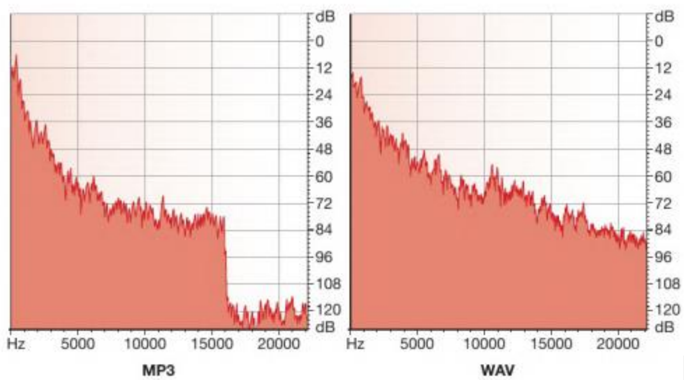
Digital images, with millions of pixels and 24-bit depth, can be large files with tiny color differences. A lossy compression algorithm analyzes data and compares color values, rewriting the file using fewer bits. The Joint Photographic Experts Group's compression technique produces JPEG files, reducing file size 17 times but maintaining little difference in detail and clarity.

▲ Digitally compressing an image makes little difference to our eyes but huge savings in file size

Audio files

Uncompressed audio files, typically in Waveform Audio (WAV) format, contain sampled sound data. These files are compressed and encoded as MPEG-1 audio layer 3 files, or MP3 files, by removing redundant data. A 30-MB WAV file can be compressed to a 3 MB MP3 equivalent, with a dip in range at high frequencies.

▶ Figure 3.12 The difference in ranges between an MP3 and a WAV sound sample can clearly be seen



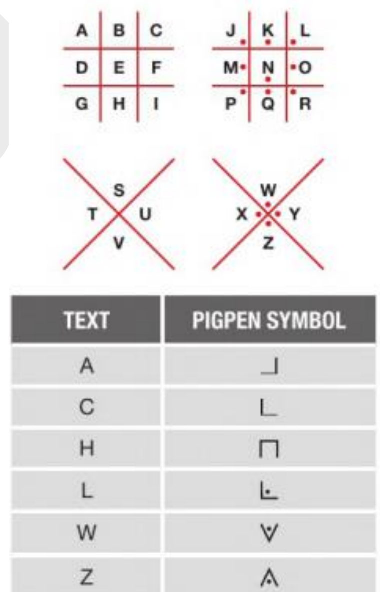
Chapter 15

Encryption

Modern methods of data encryption

Asymmetric encryption

is a method of encrypting and decrypting data using two keys: a public key known to everyone, and a private key known only to the sender. It is commonly used for sending encrypted messages as the recipient's public key can be obtained from a public directory.



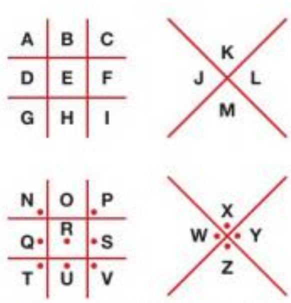
▲ Figure 3.13 The pigpen cipher

Symmetric encryption

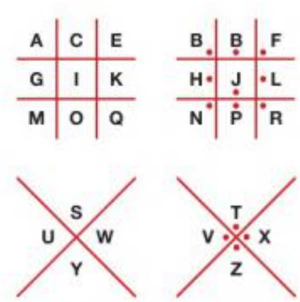
Symmetric encryption, used in HTTPS connections, involves a client and server establishing a secure connection. The client generates a key, sends it to the server, decrypts it, and uses the same key for secure transmissions.

The pigpen cipher

The pigpen cipher is a simple cipher that replaces each letter with a symbol, corresponding to the portion of the grid containing the letter. It is easy to solve unless different grids are designed, but alternative grids are shown in Figs 3.14 and 3.15 for more sophistication.



▲ Figure 3.14 Variation 1 of the pigpen cipher



▲ Figure 3.15 Variation 2 of the pigpen cipher

The Caesar cipher

The Caesar cipher is a simple encryption method named after Julius Caesar, where letters are shifted a set number of places, with positive shifts to the right and negative shifts to the left.

PLAIN TEXT (INPUT)	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
CIPHER TEXT (OUTPUT)	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B

▲ Table 3.4 The Caesar cipher with a key of –2

ALGORITHM FOR THE CAESAR CIPHER

This is a perfect example to practise your computational thinking skills. You are going to write a program to encrypt and decrypt a message using a specific key using the Caesar cipher method.

Here is an explanation of the problem:

1 Find out whether the message has to be encrypted or decrypted. 2 If the message is to be encrypted, then
3 ask the user to enter the message.
4 Ask the user for the key.
5 Analyse each character in the message and if it is a letter calculate the letter the 'key' number of places to the right if the shift is positive or to the left if it is negative.
6 If it is a punctuation mark, then do not change it.
7 Build up a new message using these new letters.
8 Output the new message.

If the message is to be decrypted, repeat the above, but reverse the shift.

Here is a flowchart of a possible solution. (figure 3.16)

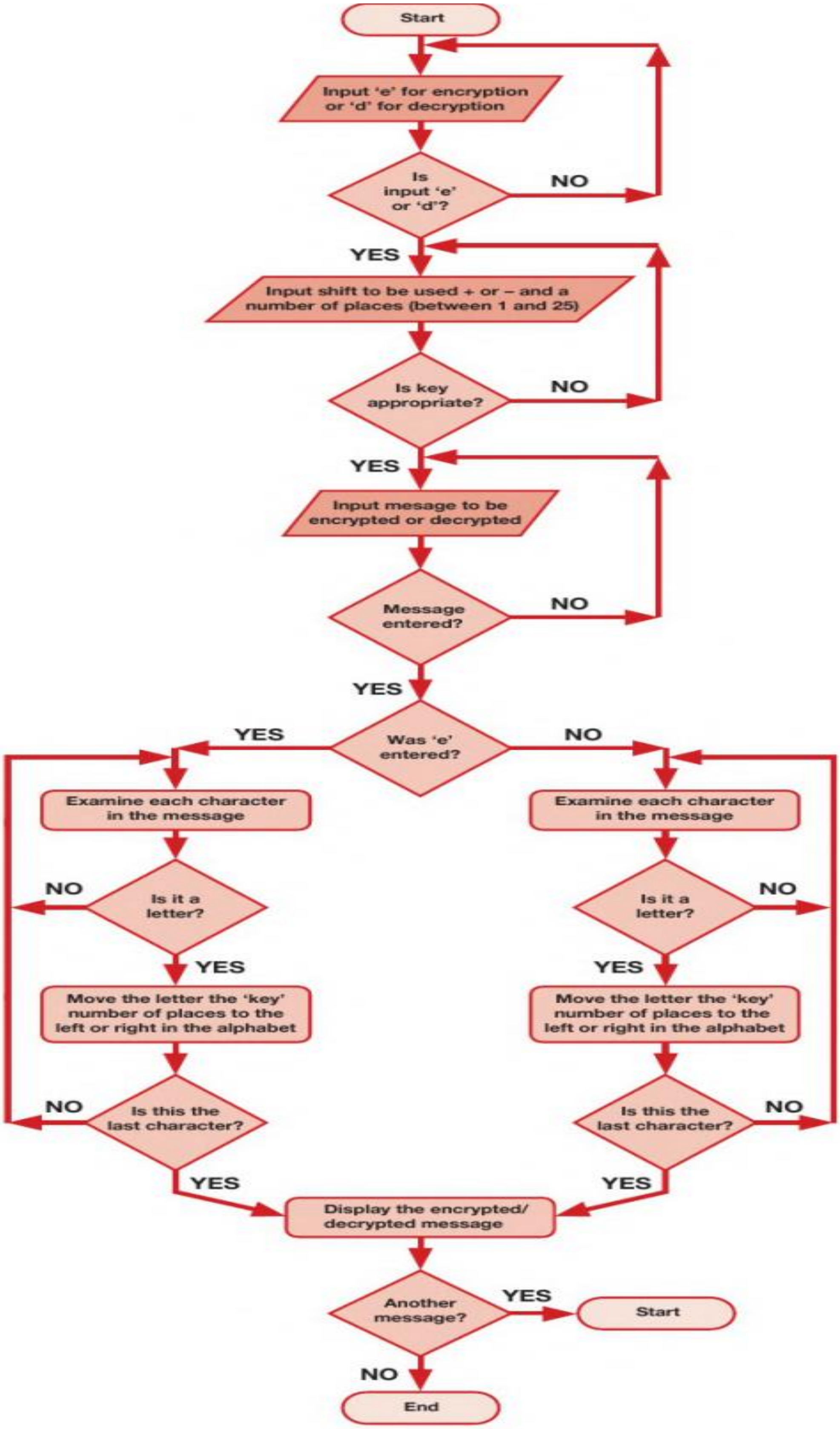
All this seems straightforward, but there are some more sub-problems.

■Your program will need to check that the user input - the key and the message has actually been entered.

■The algorithm will have to start at the first character and move to the last (i.e. it must traverse the string), therefore your program will have to find the length of the message.

■Some of the letters may be entered in upper case and some in lower case. Your program should be able to cope with this and encode or decode them.

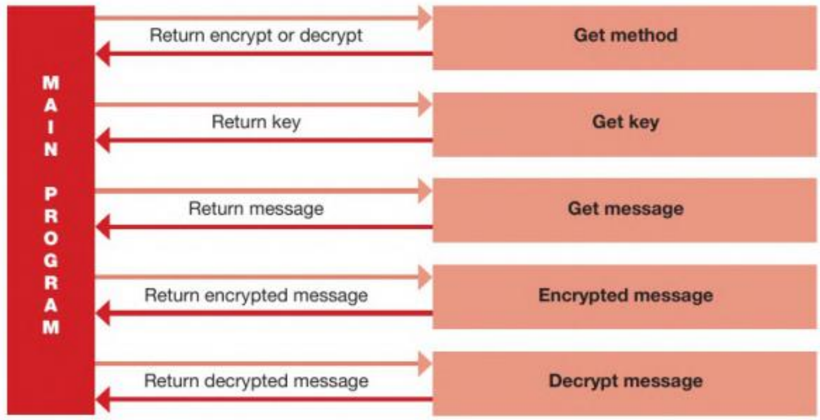
■Your program will have to leave any characters such as spaces, commas and full stops as they are.


Start
Input 'e' for encryption or 'd' for decryption
Is input 'e' or 'd'?
NO
YES
Input shift to be used + or – and a number of places (between 1 and 25)
Is key appropriate?
NO
YES
Input mesage to be encrypted or decrypted
Message entered?
NO
YES
Was 'e' entered?
YES
NO
Examine each character in the message
Is it a letter?
NO
YES
Move the letter the 'key' number of places to the left or right in the alphabet
Is this the last character?
NO
YES
Examine each character in the message
Is it a letter?
NO
YES
Move the letter the 'key' number of places to the left or right in the alphabet
Is this the last character?
NO
YES
Display the encrypted/ decrypted message
Another message?
YES
Start
NO
End

STRUCTURED PROGRAMMING

Creating the algorithm and coding the program will allow you to use structured programming using subprograms, and especially functions, which are called from the main program.

▶ **Figure 3.17** An example of structured programming



Here is the main program in pseudocode – the subprograms are not included.

`SET run TO 1`	
`WHILE run = 1 DO`	#A WHILE loop is started. It will continue until the variable 'run' does not contain the value 1.
`    SET method TO findMethod()`	#This calls the function for the user to ask for encryption or decryption.
`    SET key TO findKey()`	#This calls the function for the user to enter the key.
`    SET message TO getMessage()`	#This calls the function for the user to enter the message.
`    IF method == "e" THEN`	
`        SET newMessage TO encrypt(message, key)`	#Functions to encrypt or decrypt are called.
`    ELSE`	
`        SET newMessage TO decrypt(message, key)`	
`    END IF`	
`    SEND newMessage TO DISPLAY`	#The encrypted or decrypted message is displayed.
`    SEND "Please enter x to quit or any other key to continue" TO DISPLAY`	
`    RECEIVE response FROM (CHARACTER) KEYBOARD`	
`    IF response = "x" then`	#The user is asked if they want to encrypt or decrypt again. If not, the variable 'run' is changed from 1 and the loop will stop.
`        SET run TO "x"`	
`    ELSE`	
`        SET run TO 1`	
`    END IF`	
`END WHILE`	

The Virgine cipher

The Virgine cipher uses polyalphabetic substitution, shifting each alphabet cyclically to the left, creating a table with 26 rows matching one of the 26 possible Caesar ciphers.

▶ **Figure 3.18** The Vigenère cipher

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

ENCRYPTION

We will encrypt, 'This is a message'. The message is referred to as the **plaintext**.

PLAINTEXT	t	h	i	s	i	s	a	m	e	s	s	a	g	e

▲ **Table 3.5** Plaintext table for 'This is a message'

The **keystream** for encryption is developed by repeating the **keyword** until it matches the plaintext, ensuring each message is encrypted differently, similar to the Caesar cipher but with limitless options.

PLAINTEXT	t	h	i	s	i	s	a	m	e	s	s	a	g	e
KEYSTREAM	c	o	d	e	c	o	d	e	c	o	d	e	c	o

▲ **Table 3.6** Using 'code' as the keyword to develop a keystream

To encrypt each letter of the plaintext, the intersection of the row with the plaintext letter and the column with the keystream letter is found. The first letter of the plaintext is 't' and for the keystream it is 'c'. The row starting with 't' and the column starting with 'c' intersect at the cell containing the letter 'v'.

SUBJECT VOCABULARY

plaintext the text that is to be encrypted

keystream the characters that are combined with the plaintext to produce an encrypted message

keyword the text that is chosen to generate the keystream

▶ **Table 3.7** Using the Vigenère grid to encode the first letter

	A	B	C	D
A	A	B	C	D
B	B	C	D	E
C	C	D	E	F
D	D	E	F	G
E	E	F	G	H
F	F	G	H	I
G	G	H	I	J
H	H	I	J	K
I	I	J	K	L
K	J	K	L	M
K	K	L	M	N
L	L	M	N	O
M	M	N	O	P
N	N	O	P	Q
O	O	P	Q	R
P	P	Q	R	S
Q	Q	R	S	T
R	R	S	T	U
S	S	T	U	V
T	T	U	V	W
U	U	V	W	X

The user finds the first letter of the keystream, 'C', and decrypts the ciphertext by looking down the column to 'V', then to 'T', and repeats.

THE RAIL FENCE CIPHER

This is a simple cipher which transposes the characters following a simple rule based on a key.

'This is a message' would be encoded in the following way using a key of 3. The letters are set out on three rows or tracks following a zigzag diagonal path.

▶ **Table 3.10** The rail fence cipher with a key of 3

t				i				e				g	
	h		s		s		m		s		a		e
		i				a				s			

To encrypt the message, we read each letter from left to right across each row in order, starting with the top row, which would be t→→e → g. Therefore, the encrypted ciphertext would be:
tieghsssmsaeias
With a key of 4, it would be:
Taghsmaeliesss

▶ Table 3.11 The rail fence cipher with a key of 4

t						a						g	
	h				s		m				a		e
		i		i				e		s			
			s						s				

To decrypt a message a grid is created with as many rows as the key.

The message 'TWLIIHSEBOGNWNENRSWNTEDAIIH' has been encrypted with a key of 4. Therefore a grid with four rows is drawn and a dash placed in the cells in a diagonal path.

–						–						–						–						–			
	–				–		–				–		–				–		–				–		–		
		–		–				–		–				–		–				–		–				–	
			–						–						–						–						–

▲ Table 3.12 A rail fence cipher grid with a key of 4 ready to decrypt the message

The letters, starting from the left, replace the rows, starting with the top row.

T						W						L						I						I			
	-				-		-				-		-				-		-				-		-		
		-		-				-		-				-		-				-		-				-	
			-						-						-						-						-
T						W						L						I						I			
	H				S		E				B		O				G		N				W		N		
		-		-				-		-				-		-				-		-				-	
			-						-						-						-						-
T						W						L						I						I			
	H				S		E				B		O				G		N				W		N		
		E		N				R		S				W		N				T		E				D	
			-						-						-						-						-
T						W						L						I						I			
	H				S		E				B		O				G		N				W		N		
		E		N				R		S				W		N				T		E				D	
			A						I						I						H						-

▲ Table 3.13 The stages of filling in the letters of the ciphertext to decrypt the message

Revision questions

1)[2023 may:2]
Computers manipulate binary patterns. People interpret those patterns.
(a) Complete the table by adding the hexadecimal notation for each of the denary values.

Denary	Hexadecimal
8	
12	

[2]

(b) Identify the expression to give the number of unique binary patterns that can be [1]
stored in six bits.

A. 6^2
B. 6 X 2
C. 6^2 - 1
D. 6^2

(c) Binary patterns are manipulated by shifts.

(i) Give the result of applying a logical shift right by two to the binary
pattern 0101 1100 [1]

(ii) Give the result of applying an arithmetic shift right by three to the binary
pattern 1100 0101 [1]

(d) Binary patterns can be interpreted as signed or unsigned integers.

(i) Convert the denary unsigned integer 60 to 8-bit binary. [2]

(ii) Here is a binary bit pattern for a signed integer in sign and magnitude format.
1001 0110
Convert the binary bit pattern to denary.
Be sure to include a sign symbol in your answer. [2]

(iii) Negating a signed integer means changing its sign without changing its value. [2]
The negation of +16 is -16. The negation of -24 is +24.
Here is the binary bit pattern for a signed integer in two's complement format.
1110 0101
Convert the binary pattern to its negation in two's complement.

(e) Construct an expression to convert 13 kilobytes to kibibytes.
You **do not** need to do the calculation. [2]
2). [2023 may:3]

(b) Images are stored as files before being displayed or printed.

(i) Give **one** measurement of image resolution. [1]

(ii) Increasing the colour depth of an image leads to an increase in the image file size.
Describe the reason for the increase in file size. [2]

(e) Here is part of a file that contains electric meter readings.
04631 04984 05103 05163 05271 05383 05487 05722
Explain the effect on the file of applying a run-length encoding algorithm to this data file. [2]

3). [2022 may:2]
Computers use binary to represent and store data.

(a) The denary number 78 is the ASCII code for the character N.

(i) Convert the denary number 78 to 8-bit binary. [2]

(ii) Identify the number of characters that can be represented using standard ASCII. [1]

A. 64
B. 128
C. 256
D. 512

(iii) Explain one reason for using Unicode rather than ASCII to encode languages other than English. [2]

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

(b) Convert the denary number -43 to 8-bit binary using sign and magnitude representation. [2]

(c) Complete the table by adding these two 8-bit binary integers. [2]

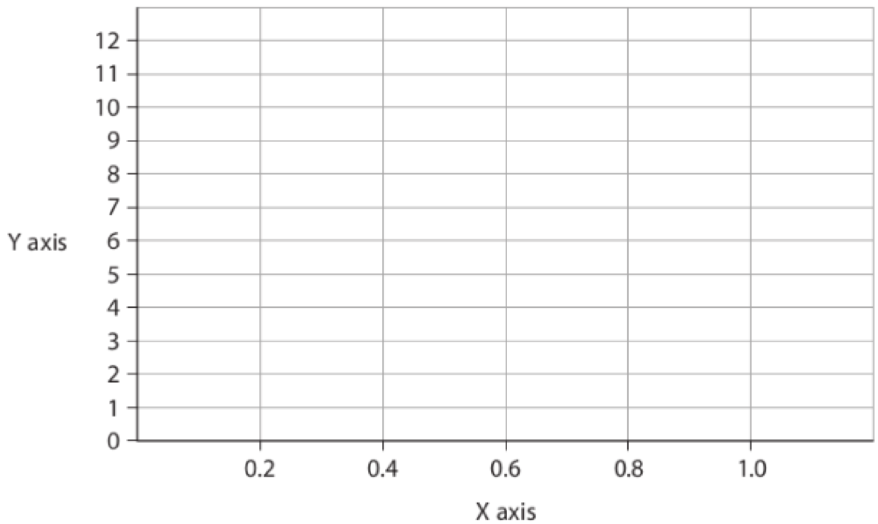
(d) A bitmap image is made up of pixels.

(i) An image has five colours.
Complete the table by adding a unique binary pattern for each colour.
Each pattern must use the same minimum colour depth. [2]

Colour	Binary pattern
Green	
Black	
White	
Red	
Blue	

(ii) Another image is 3579 pixels high and 6128 pixels wide.
The image is stored with a 32-bit colour depth.
The metadata for the image is 732 bytes.
Construct an expression to show how the file size, in megabytes, is calculated. You do **not** need to do the calculation. [4]

4). [2022 may:3]
Alyssa is a music producer.
(a) Figure 1 shows the denary values of five samples of an analogue sound using a sample interval of 0.2 seconds.



Sample number	Denary value
1	1
2	10
3	12
4	5
5	3

Figure 1

(i) Complete this graph using the sample information from Figure 1 to show the digital sound wave. [3]

(ii) Give a suitable label for the X axis. [1]

(iii) Give a suitable label for the Y axis. [1]

5). [2021 Nov:1]
Computers use binary to represent and store data.

(a) Binary and hexadecimal notation can be used to represent numbers.

(i) Convert the denary number 77 to 8-bit binary. [2]

(ii) Convert the denary number -126 to 8-bit binary using two's complement. [2]

(iii) Convert the binary pattern 11000110 to hexadecimal.[2]

(b) Images are stored as binary data.
(i) A bitmap image is 400 pixels wide by 200 pixels high.
It has a colour depth of 12 bits.
Construct an expression to show how the image size, in bytes, is calculated.
Do not calculate the answer. [2]

(ii) Figure 1 is an image of an icon, set out in an 8 x 8-pixel grid.
Figure 1
A run-length encoding compression algorithm uses:
- W for white pixels
- B for black pixels.
Give the result of compressing the first 4 lines of the icon using the algorithm.
Line 1
Line 2
Line 3
Line 4

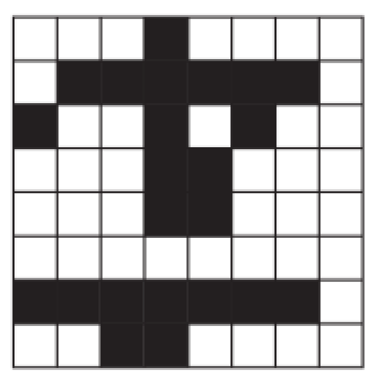


Figure 1

(iii) Run-length encoding is a type of lossless compression. [1]
State what is meant by the term **lossless** in this context.

(c) Raoul wants to record a song and store it as a digital sound file.
Figure 2 shows an overview of the process of recording a song and storing it as a digital sound file.

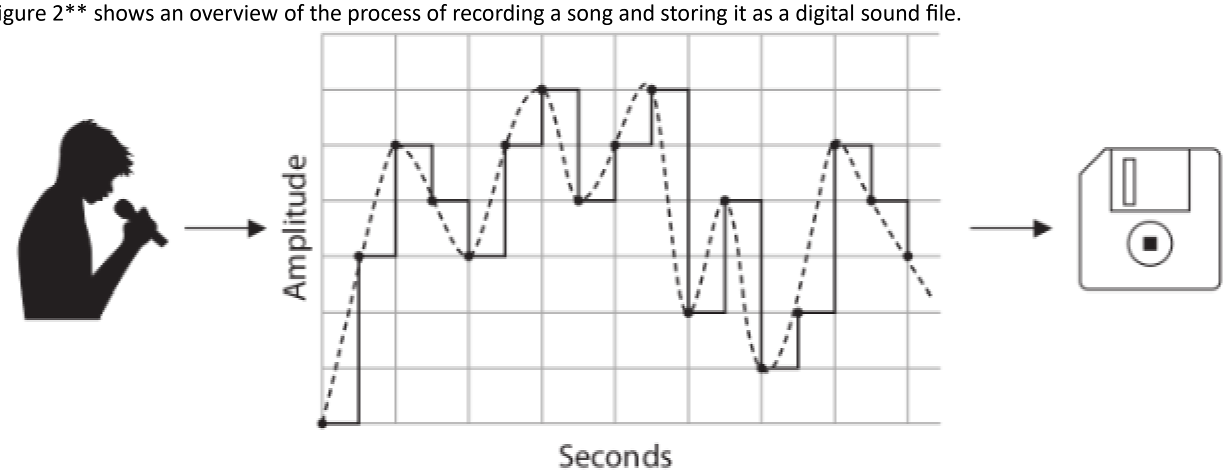


Figure 2

(i) Describe the steps taken to convert the analogue sound to a digital sound file. [3]

(ii) Raoul uses a lossy algorithm to compress the file.
Give **one** reason why the sound quality of the compressed file may not be as good as the uncompressed version. [1]

6). [2021 June:1]
Binary digits (bits) are grouped together to represent different types of data.
(a) Hexadecimal notation is sometimes used to represent patterns of binary digits.
Identify **one** reason why programmers use hexadecimal notation. [1]
A. Easier for a computer to understand.
B. Easier for humans to read.
C. Takes up less computer memory.
D. Quicker than binary to execute.

(b) Convert the bit pattern 0101 1010 to hexadecimal. [2]
(c) Identify the number of binary patterns that can be represented by 8 bits. [1]
A. 128
B. 256
C. 512
D. 1024

(d) Complete the table by adding these two 8-bit binary integers. (2)

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

(e) Give the result of performing a logical left shift of 1 place on the binary integer 0100. (1)

(f) Give the number of bits per character used by standard ASCII (1)

(g) **Figure 1** shows a black and white bitmap image.
The pixels in row 5 are represented by the binary pattern 1001 1001

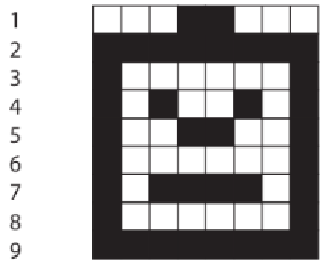


Figure 1

(i) State what is meant by the term **pixel**. (1)

(ii) Construct an expression to show the number of pixels in the image. (1)

(iii) The image is changed so that any pixel can be one of 16 different colours.
State the minimum number of bits that would be needed to represent one pixel. (1)

6). [2021 June:2]
(a) Files may be compressed using lossless or lossy algorithms.

(i) Music files are distributed over the internet. [2]
Give **two** reasons why lossy compression is used when distributing music files over the internet.

(ii) Identify which **one** of these file formats uses lossy compression.
A. BMP
B. DOC
C. JPEG
D. PNG (1)

(iii) Here is a string of data.
CCCWWWCCWWWWWWWCCC
Give the result of compressing the string using a run-length encoding algorithm. (1)

(g) A video file is to be transmitted over the internet.
* The network transmission speed is 54 Mbps.
* The file size is 6 gigabytes (GB).
Construct an expression to show how the transmission time, in seconds, is calculated.
You **do not** must do the calculation. (4)