

*0478 supplies no formula sheet, and most of what is needed is not really a formula: it is a **place-value row, a range or a file-size calculation** that has to be recalled exactly. Each one is a mark on its own, and each is one line to learn. Write the place values down before converting anything, and put the unit on every line of the working. 0478 ; ,*

Number bases – write the place values down first

Binary place values – for one byte, 128 64 32 16 8 4 2 1 from left to right. Write this row above the digits and every conversion becomes addition.

Hexadecimal digits – 0–9 then A = 10, B = 11, C = 12, D = 13, E = 14, F = 15. Place values are ... 256 16 1.

Binary to denary – add the place values wherever there is a 1. So $0110\ 1110 = 64 + 32 + 8 + 4 + 2 = 110$.

Denary to binary – work down the place values from 128, writing 1 whenever the value fits and subtracting it. Check by adding the row back up.

Binary to hexadecimal – split into groups of four bits FROM THE RIGHT and convert each group. One byte is always two hex digits: $0110\ 1110 = 6E$.

Hexadecimal to binary – each hex digit becomes four bits, keeping leading zeros. $3F = 0011\ 1111 = 63$.

Why hexadecimal is used – it is shorter than binary and less error-prone to read or type, and it converts to binary exactly – four bits per digit. Used for MAC addresses, IPv6, HTML colour codes, assembly and memory dumps.

I Convert the denary number 202 to binary and then to hexadecimal.

$202 = 1100\ 1010$ in binary and CA in hexadecimal.

Commentary: 128 fits, remainder 74; 64 fits, remainder 10; 32 and 16 do not; 8 fits, remainder 2; 4 does not; 2 fits, remainder 0; 1 does not

Commentary: So $202 = 1100\ 1010$

Commentary: Group from the right: $1100 = C$ and $1010 = A$, so $202 = CA$

Ranges, negatives and shifts

What one byte can hold – 8 bits give $2^8 = 256$ different patterns, so an unsigned byte holds 0 to 255.

Unsigned range for n bits – 0 to $2^n - 1$. For 16 bits that is 0 to 65 535.

Two's complement range – n bits hold -2^{n-1} to $2^{n-1} - 1$, so one byte holds -128 to $+127$. The leftmost bit carries -128 , which is why the range is not symmetric.

Making a two's complement negative – write the positive value in binary, invert every bit, then add 1. So $+35 = 0010\ 0011$ becomes $1101\ 1101 = -35$.

Overflow – the result of a calculation needs more bits than the register has, so the extra bit is lost and the answer is wrong. An 8-bit register cannot hold $200 + 100$.

Logical shift left by n places – multiplies by 2^n ; zeros come in on the right and bits shifted off the left are LOST, which can change the value completely.

Logical shift right by n places – divides by 2^n and throws away the remainder; zeros come in on the left. 1001 0110 (150) shifted right once is 0100 1011 (75).

How much space data takes

Bits and bytes – 8 bits = 1 byte. A file size asked for in bytes must be divided by 8; this single step is the commonest lost mark in the topic.

Data units – 1 KiB = 1024 bytes, 1 MiB = 1024 KiB, 1 GiB = 1024 MiB; while 1 kB = 1000 bytes and 1 MB = 1000 kB. Use whichever the question uses.

Number of colours – $2^{\text{colour depth}}$. A colour depth of 1 bit gives 2 colours, 8 bits gives 256, 24 bits gives about 16.7 million.

Image file size – image resolution (width $\times$ height in pixels) $\times$ colour depth in bits. The answer is in BITS.

Sound file size – sample rate (Hz) $\times$ sample resolution (bits) $\times$ length of the recording (seconds), again in bits. Double it for stereo.

Transmission time – file size $\div$ transmission rate, with both in the same unit. A file in bytes against a link in bits per second differs by a factor of 8.

Compression – compressed size $\div$ original size, usually as a percentage. Read carefully whether the question wants the size it was reduced TO or reduced BY – a 5 MB file compressed to 2 MB is 40% of the original, a 60% saving.

| An image is 1024 by 768 pixels with a colour depth of 8 bits. Find its size in MiB.

0.75 MiB (6\,291\,456 bits, or 786\,432 bytes).

Commentary: Bits = $1024 \times 768 \times 8 = 6\,291\,456$

Commentary: Bytes = $6\,291\,456 \div 8 = 786\,432$

Commentary: KiB = $786\,432 \div 1024 = 768$

Commentary: MiB = $768 \div 1024 = 0.75$

| A 30-second mono recording is sampled at 44.1 kHz with a resolution of 16 bits. Find its size in MiB.

About 2.52 MiB.

Commentary: Bits = $44\,100 \times 16 \times 30 = 21\,168\,000$

Commentary: Bytes = $21\,168\,000 \div 8 = 2\,646\,000$

Commentary: MiB = $2\,646\,000 \div 1024 \div 1024 = 2.52$

Characters, addresses and error checking

ASCII ASCII – 7 bits per character gives 128 characters; extended ASCII uses 8 bits for 256. One character of ASCII text therefore takes one byte.

Unicode Unicode – far more bits per character, covering every writing system including Chinese. More characters, larger files – that trade-off is the marked point.

MAC address MAC – 48 bits, written as six pairs of hex digits. The first three pairs identify the manufacturer, the last three the device, and it does not change.

IP address IP – IPv4 is 32 bits, written as four denary numbers each 0 to 255; IPv6 is 128 bits, written as eight groups of four hex digits.

Parity check – a parity bit is set so the total number of 1s is even (even parity) or odd (odd parity). It detects an ODD number of flipped bits, and misses an even number – say so, because that limitation is the answer.

Checksum – a value calculated from the data block, sent with it, and recalculated on arrival. Different values mean corruption, but not where it happened.

Check digit – an extra digit calculated from the others and added to the end, as in a barcode or an ISBN. It catches mistyped and swapped digits at the point of entry.

Echo check – the receiver sends the data back and the sender compares. Simple, but it doubles the traffic and cannot tell whether the error happened on the way out or on the way back.

Automatic repeat request (ARQ) – the receiver sends an acknowledgement; if none arrives before a timeout, the sender retransmits, up to a set number of attempts.

Where these marks are lost

Leaving a file size in bits when the question asked for bytes, KiB or MiB. Write the unit on every line of the working and the final division will not be forgotten.

, bit byte

Grouping bits into fours from the LEFT when converting to hex. Group from the right and pad with zeros on the left.

Treating the leftmost bit of a two's complement byte as +128. It is –128, which is why the range stops at +127.

–128

Saying a parity check “finds errors”. It DETECTS some errors and corrects none – and it misses any even number of flipped bits.

,

Mixing 1000 and 1024 in the same question. Kilobyte and kibibyte are different units; follow whichever the question uses.

1000 1024