Cambridge IGCSE · Computer Science 0478 · Notes

Data storage and compression

Read online at **pastopic.com/cambridge/computer-science-0478/notes/data-storage-and-compression**

Last checked 30 September 2026

Read first: **Text, sound and images**

What you need to know

By the end of this topic you should be able to:

1. **Say how data storage is measured**, from the smallest unit to the largest: bit, nibble, byte, kibibyte (KiB), mebibyte (MiB), gibibyte (GiB), tebibyte (TiB), pebibyte (PiB) and exbibyte (EiB). Know how many of each unit make the next one: 4 bits in a nibble, 8 bits (2 nibbles) in a byte, then 1024 of each unit in the next, for example 1024 MiB in a GiB.
2. **Calculate the file size of an image file and of a sound file** from the facts in the question: the resolution and colour depth of an image, or the sample rate, sample resolution and length of a sound. Give the answer in the unit the question asks for, and use 1024 (not 1000) between units.
3. **Explain why files are compressed**: compression makes a file smaller, so it takes less storage space, is quicker to send and needs less bandwidth to send.
4. **Explain how lossy and lossless compression work.** Lossy compression makes a file smaller by permanently removing data, for example by lowering an image's resolution or colour depth, or a sound's sample rate or sample resolution. Lossless compression makes a file smaller without permanently losing any data, for example with run-length encoding (RLE).

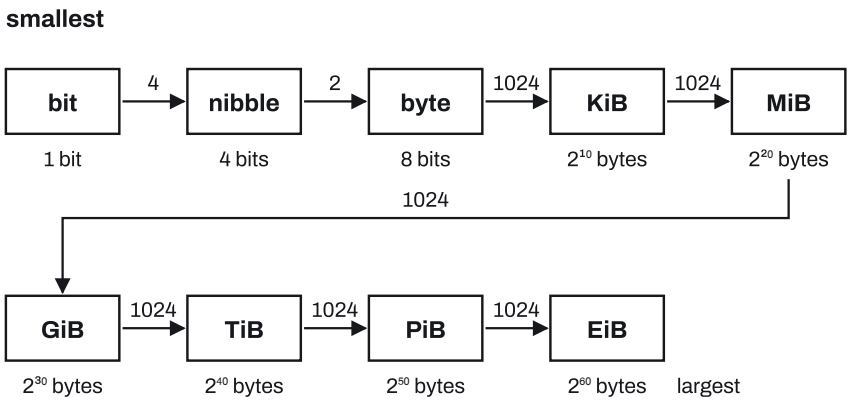
What resolution, colour depth, sample rate and sample resolution mean is in the Text, sound and images note (see Read first).

Understand it

Units of storage

Everything a computer stores is binary, so the smallest unit is one **bit**: a single 1 or 0. Bits are grouped into bigger units:

- **nibble** = 4 bits;
- **byte** = 8 bits = 2 nibbles;
- from the byte upwards, each unit is **1024** of the one below: 1024 bytes = 1 KiB, 1024 KiB = 1 MiB, 1024 MiB = 1 GiB, then TiB, PiB and EiB.



to a bigger unit: ÷ by the number on each arrow
to a smaller unit: × by the number on each arrow

Why 1024 and not 1000? Computers count in binary, and $1024 = 2^{10}$ is the power of 2 nearest to 1000. The "bi" in kibi, mebi and gibi means **bi**nary: 1 KiB is exactly 2^{10} bytes, 1 MiB is 2^{20}, 1 GiB is 2^{30} bytes. The names without "bi" (kilobyte kB, megabyte MB, gigabyte GB) are the ordinary 1000-based units, so 1 GB is 1 000 000 000 bytes but 1 GiB is 1 073 741 824 bytes. The syllabus uses KiB, MiB and so on, and every calculation uses 1024.

Converting is one rule:

- to a **bigger** unit, **divide** by 1024 for each step (by 8 from bits to bytes, by 4 from bits to nibbles);
- to a **smaller** unit, **multiply**.

So 2 GiB = $2 \times 1024 = 2048$ MiB (one step down), and 1536 KiB = $1536 \div 1024 = 1.5$ MiB (one step up). Count the steps: from GiB to KiB is two steps, so multiply by 1024 twice.

To **compare** sizes written in different units, change them all to the same unit first.

File sizes

An image or sound file is just a long list of binary numbers, so its size is **how many numbers × how many bits each**.

Image: every pixel stores one colour, using as many bits as the colour depth.

$$\text{image size (bits)} = \text{width (pixels)} \times \text{height (pixels)} \times \text{colour depth (bits)}$$

Sound: every second, the sample rate says how many samples are taken, and each sample uses as many bits as the sample resolution.

$$\text{sound size (bits)} = \text{sample rate (Hz)} \times \text{sample resolution (bits)} \times \text{length (seconds)}$$

Then divide by 8 to get bytes, and by 1024 for each step up to KiB, MiB or GiB. For example, a 200 × 100 image with an 8-bit colour depth is $200 \times 100 \times 8 = 160\,000$ bits $= 20\,000$ bytes. These sums ignore the small amount of extra data (metadata) a real file also holds.

Why compress?

Compression means reducing the size of a file. A smaller file:

- takes up **less storage space** (on a device, a server or a USB drive);
- is **quicker to send**, upload, download or stream, because there are fewer bits to transmit;
- needs **less bandwidth** to transmit, and uses less of a mobile **data allowance**;
- can fit under a **file size limit**, such as the largest attachment an email service allows;
- streams with **less buffering** (less lag) for the user.

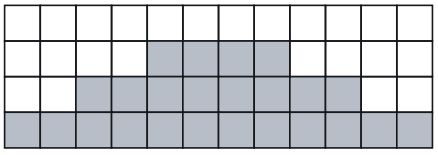
A compressed file is not "easier" to send: the steps are the same; it just arrives sooner.

Lossless compression

Lossless compression makes the file smaller **without permanently removing any data**. A compression algorithm finds data that repeats and stores it more briefly. When the file is opened, the original is **rebuilt exactly**, bit for bit.

The method named in the syllabus is **run-length encoding (RLE)**. A **run** is a group of the same value one after another. RLE stores each run once, as a pair: **how many times it repeats, and the value**. The text AAAAAABBBCCCCD (14 characters) becomes 6A3B4C1D (8 characters).

In an image, the runs are pixels of the same colour next to each other:

image (12 × 4 pixels)	**stored as runs (count, colour)**
	(12, W) (4, W) (4, G) (4, W) (2, W) (8, G) (2, W) (12, G)

W = white (unshaded), G = grey (shaded)

before: 48 pixels × 1 byte = 48 bytes

after: 8 runs × 2 bytes = 16 bytes, and every pixel can be rebuilt

The same idea fits each kind of file:

- **image**: repeating pixels (runs of the same colour) are found and stored as the colour and the number of times it repeats;
- **text**: text is turned into binary using a character set, such as ASCII or Unicode, which gives each character its own unique binary code (see Text, sound and images). Repeating characters, words or phrases are found; each is stored once in an index, and the text then just refers to its place in the index each time it occurs, with its positions;
- **sound**: repeating patterns in the music are found and indexed;
- **video**: pixels that repeat from one frame to the next are stored once, and only the **changes** between frames are recorded.

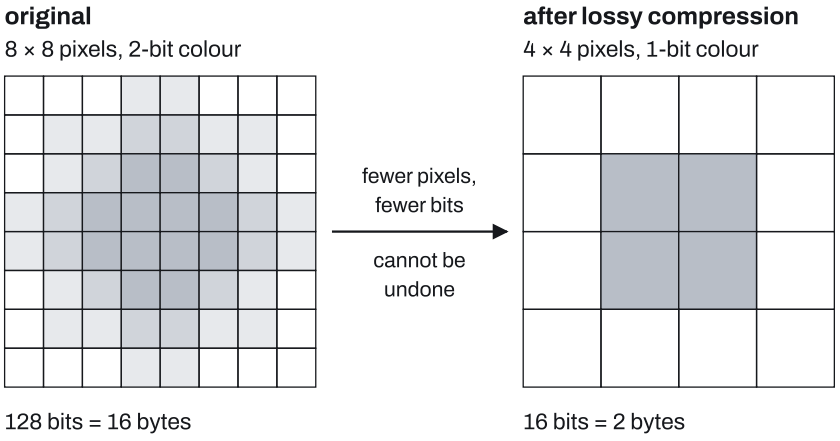
RLE only helps when there **are** runs. ABCD would become 1A1B1C1D, which is **longer**, so photographs with few repeats shrink much less than simple drawings. Lossless files are therefore usually **bigger** than lossy ones.

Use lossless when **no data may be lost**: text and program files (one wrong character spoils them), an image that will be printed or edited, or music where the highest quality matters.

Lossy compression

Lossy compression makes the file smaller by **permanently removing data**, usually data that people will barely notice (**unnecessary** data, such as shades too similar to tell apart). The original file **cannot** be rebuilt. The algorithm lowers the settings that control the file size:

- **image**: reduce the **resolution** (fewer pixels) or the **colour depth** (fewer bits per colour, so fewer colours);
- **sound**: reduce the **sample rate** (fewer samples each second) or the **sample resolution** (fewer bits per sample), and remove sounds the human ear cannot hear (**perceptual music shaping**);
- **video**: all of the image and sound changes, and reduce the **frame rate** (fewer frames each second).



Lossy compression makes files **much smaller** than lossless, so it suits files that are streamed or put on websites, where speed matters more than perfect quality. The cost is **lower quality**: an image can look blocky or blurred, and sound less like the original.

JPEG (images), MP3 (sound) and MP4 (video) are **lossy** compressed file formats.

	Lossless	Lossy
Data removed?	No, none permanently	Yes, permanently
Original rebuilt exactly?	Yes	No
Size reduction	smaller	much smaller
Quality	same as the original	lower

	Lossless	Lossy
Example method	run-length encoding (RLE)	lower resolution, colour depth, sample rate or sample resolution
Good for	text, programs, printing, editing, top-quality music	streaming, websites, thumbnails, video

Key facts and formulas

There is no formula list for this subject, so you must learn all of these.

Formula	Status
Order: bit, nibble, byte, KiB, MiB, GiB, TiB, PiB, EiB (smallest to largest)	Learn it
1 nibble = 4 bits; 1 byte = 8 bits = 2 nibbles	Learn it
1 KiB = 1024 bytes; 1 MiB = 1024 KiB; 1 GiB = 1024 MiB; 1 TiB = 1024 GiB; 1 PiB = 1024 TiB; 1 EiB = 1024 PiB	Learn it
To a bigger unit divide (by 1024 each step); to a smaller unit multiply	Learn it
Image file size (bits) $=$ width $\times$ height $\times$ colour depth	Learn it
Sound file size (bits) $=$ sample rate $\times$ sample resolution $\times$ length in seconds	Learn it
bits $\div 8$ = bytes; bytes $\div 1024$ = KiB; KiB $\div 1024$ = MiB	Learn it
Compression = reducing the size of a file	Learn it
Lossy: data removed permanently (lower resolution, colour depth, sample rate, sample resolution)	Learn it
Lossless: no data removed permanently; e.g. RLE stores each run as (count, value)	Learn it
JPEG, MP3 and MP4 are lossy compressed files	Learn it

How to do it

In the Paper 1 questions we have tagged for this topic (33 questions, 2021–2025), the types came up this often. 14 of the 33 questions mix types, so the counts add up to more than 33. Parts about other topics in the same questions (packets, storage devices, character sets) are not counted.

Question type	Questions
Units of storage and conversions	13
Lossy or lossless: identify, choose and justify	13
Why compress: purpose and benefits	9
Calculate an image or sound file size	6
Describe how lossless compression works	5
Describe how lossy compression reduces a file	4

1. Units of storage and conversions (13 questions)

Short 1-mark parts. Learn the chain in the first diagram, then:

1. Work out **how many steps** apart the two units are.
2. Going **up** (to a bigger unit), **divide**; going **down**, **multiply**, by 1024 for each step.

Variations you will meet:

- **Name a unit**: the smallest unit (bit), the unit equal to 4 bits (nibble), 8 bits (byte), 1024 GiB (TiB); the largest of a list (the one furthest along the chain). "Bot" or "memory unit" are not units.
- **Counts**: bits in a byte (8), nibbles in a byte (**2**, not 4), nibbles in 2 bytes (4), bits in a KiB ($1024 \times 8 = 8192$).
- **One step**: 4 GiB = 4096 MiB; 512 KiB = 0.5 MiB; 1 EiB = 1024 PiB; 4096 MiB = 4 GiB.
- **More than one step**: 5 TiB = $5 \times 1024 = 5120$ GiB, and 2560 MiB = $2560 \div 1024 = 2.5$ GiB. From GiB to TiB is one step, so 4 GiB = $4 \div 1024 = 0.0039$ TiB (to 4 d.p.).
- **"How many TiB are in 3 PiB?"** is **one** step: $3 \times 1024 = 3072$. Multiplying by 1024×1024 is a common slip.
- **Compare sizes** in different units: change them all to one unit first. Older papers used kB, MB and GB and allowed 1000 or 1024; the current syllabus always uses the "bi" units and 1024.

2. Lossy or lossless: identify, choose and justify (13 questions)

Identify the type from a description or a file:

- "permanently reduces the colour depth and resolution" or "reduces the sample rate and sample resolution" → **lossy**;
- "no data is permanently removed" or "recognises repeating patterns and indexes them" → **lossless**;

- a JPEG, MP3 or MP4 file → **lossy compressed** file (not "not compressed");
- "give two types of compression" → lossy and lossless.

Give the **type** (lossy or lossless) when asked for a type, not a file format such as MP3.

Choose and justify. Name the type, then give a reason **tied to the scenario**:

1. **Lossless**, when the file must keep its quality: the photos will be printed, the image is sold in high quality, the musician wants to edit the recording later, the text must not change. Reason: no data is permanently removed, so the original can be rebuilt and the quality stays the same; lossy would permanently remove data and lower the quality (blurring or pixelation).
2. **Lossy**, when size and speed matter more than perfect quality: videos on a website, thumbnails, streaming. Reason: lossy makes the file much smaller, so it takes less storage on the web server, loads or downloads faster, buffers less and uses less bandwidth; the video does not need to be top quality.

Drawbacks you may be asked for:

- of **lossy**: quality is reduced; data is lost for good, so the original file cannot be rebuilt;
- of **lossless** (compared with lossy): the file is **larger**, so it takes more storage space on the web server, takes longer to upload, download or stream (buffering), needs more bandwidth and uses more data allowance.

"Justify" wants **why this type suits this case**, not a description of how the method works.

3. Why compress: purpose and benefits (9 questions)

- **"State what is meant by (file) compression"**: reducing the size of a file. Don't just name lossy or lossless: that is an example, not a meaning.
- **"Give the name of the process that reduces a file's size"**: compression.
- **Benefits** (give as many as the marks): less storage space; quicker to send, upload or download; less bandwidth needed; less data allowance used; fits an email's attachment size limit; less buffering when streaming.
- **Use the scenario and don't repeat the question.** If it says "so that it has a shorter transmission time", give a different benefit, such as less storage space. For an emailed image, say why it helps **the email**: attachment limits, faster to send and receive, less space on the email server.
- **"State the effect of compression on the file size"**: it reduces the file size.

4. Calculate an image or sound file size (6 questions)

Image:

1. Pixels = width × height.
2. Bits = pixels × colour depth. If the colour depth is in **bytes** (such as "2 bytes"), multiply by it to get bytes straight away.
3. Bytes = bits ÷ 8, then ÷ 1024 for each step up to the unit asked for.

Sound:

1. Bits = sample rate × sample resolution × length in seconds (turn minutes into seconds first).
2. Bytes = bits ÷ 8, then ÷ 1024 for each step up.

Variations you will meet:

- **Several files**: find one, then multiply by how many there are.
- **"How many files fit?"** Change the space into the same unit as one file, divide, and **round down** to a whole number of files. For example, 100 × 100 images at 16-bit colour are $100 \times 100 \times 16 \div 8 = 20\,000$ bytes each, and 5 MiB is $5 \times 1024 \times 1024 = 5\,242\,880$ bytes, so $5\,242\,880 \div 20\,000 = 262.1$, that is **262** images.
- **The answer's unit is fixed** by the question ("Answer ... KiB"). Write the unit.

Write each stage: the working earns most of the marks.

5. Describe how lossless compression works (5 questions)

Build a chain of separate points (each is a mark):

1. A **compression algorithm** is used,
2. such as **run-length encoding (RLE)**.
3. **Repeating data is found**: for an image, runs of pixels of the same colour; for text, repeated characters, words or phrases; for sound, repeated patterns; for video, pixels that repeat between frames.
4. Each is **stored once**, with **the number of times it repeats** (and, for text, it is indexed with its positions).
5. **No data is permanently removed**, so the original file can be rebuilt exactly.

Fit the points to the file in the question. For an image, write about repeating pixels and colours, not repeated words; for text, words and characters, not pixels. Giving a file type (PNG) or "put it in a zip folder" does not name a method. Never mention removing sounds people can't hear: that is lossy.

6. Describe how lossy compression reduces a file (4 questions)

1. A **compression algorithm** is used.
2. It **permanently removes data**, usually **unnecessary** data that people will barely notice; the original cannot be rebuilt.
3. Then name the changes that fit the file:
   - image: reduce the resolution (number of pixels); reduce the colour depth (bits per pixel, so the range of colours);
   - sound: reduce the sample rate; reduce the sample resolution; remove sounds the human ear cannot hear (perceptual music shaping);
   - video: any of those, and reduce the frame rate or remove repeating frames.

"Give two reductions lossy compression makes to the images, other than file size": the resolution, the colour depth, the quality (detail), the metadata.

Worked examples

The bold codes show where marks are earned, as in Cambridge mark schemes. The number is how many marks.

- **M1** method mark: for a correct method, even if the numbers slip.
- **A1** accuracy mark: for a correct answer; it needs the M mark before it.
- **B1** mark for a correct result on its own, with no method needed.

Example 1

A student's phone shows file sizes in different units.

(a) Give the size of a 3 MiB photo in kibibytes (KiB). **[1]**

(b) A backup is 6144 GiB. Give its size in tebibytes (TiB). **[1]**

(c) State how many bytes are equal to 20 nibbles. **[1]**

(d) Three files are 2 GiB, 2100 MiB and 2 000 000 KiB. Identify the largest file. Show your working. **[2]**

(a) One step down, so multiply: $3 \times 1024 = $ **3072 KiB**. **B1**

(b) One step up, so divide: $6144 \div 1024 = $ **6 TiB**. **B1**

(c) 2 nibbles make a byte: $20 \div 2 = $ **10 bytes**. **B1**

(d) Change all three to MiB: $2 \times 1024 = 2048$ MiB; 2100 MiB; $2\,000\,000 \div 1024 = 1953.125$ MiB. **M1** The largest is **2100 MiB**. **A1**

Example 2

A digital camera takes photos with a resolution of 640 × 480 pixels and a colour depth of 24 bits.

(a) Calculate the file size of one photo in kibibytes (KiB). Show all your working. **[3]**

(b) The photos are stored on a 1 GiB memory card. Calculate how many photos can be stored. **[2]**

(c) The photos are compressed with lossy compression that reduces the colour depth to 8 bits. Give the new file size of one photo in KiB, and one drawback of this change. **[2]**

(a) $640 \times 480 = 307\,200$ pixels; $307\,200 \times 24 = 7\,372\,800$ bits. **M1**

$7\,372\,800 \div 8 = 921\,600$ bytes. **M1** $921\,600 \div 1024 =$ **900 KiB**. **A1**

(b) $1\ \text{GiB} = 1024 \times 1024 = 1\,048\,576$ KiB. **M1** $1\,048\,576 \div 900 = 1165.08$, so **1165 photos** (a part photo can't be stored). **A1**

(c) A third as many bits per pixel, so $900 \div 3 =$ **300 KiB**. **B1** The photos can show fewer colours, so the quality is lower, and the removed data cannot be got back. **B1**

Example 3

A band records a 2-minute song with a sample rate of 32 000 Hz and a sample resolution of 16 bits.

(a) Calculate the file size of the recording in kibibytes (KiB). Show all your working. **[3]**

(b) The band will edit the recording later. Identify the most suitable type of compression and justify your choice. **[3]**

(c) Describe how the type of compression you chose reduces the file size. **[3]**

(a) 2 minutes = 120 seconds. $32\,000 \times 16 \times 120 = 61\,440\,000$ bits. **M1**

$61\,440\,000 \div 8 = 7\,680\,000$ bytes **M1**, and $7\,680\,000 \div 1024 =$ **7500 KiB** (about 7.32 MiB). **A1**

(b) Lossless. **B1** No data is permanently removed, so the band keeps the original recording to edit. **B1** Lossy would permanently remove data and lower the sound quality. **B1**

(c) A compression algorithm such as run-length encoding is used. **B1** Repeated patterns in the sound data are found **B1** and stored once with the number of times they repeat (indexed), so the file is smaller but can be rebuilt exactly. **B1**

Example 4

One row of an image is 20 pixels long: 7 white, then 3 black, then 10 white. Each pixel uses 8 bits.

(a) Show how this row is stored using run-length encoding. **[2]**

(b) Each run is stored as 1 byte for the count and 1 byte for the colour. Calculate how many bytes are saved. **[2]**

(c) Explain why RLE would not reduce the size of a row that goes white, black, white, black, ... for 8 pixels. **[2]**

(a) Each run is stored as (count, colour): **(7, white) (3, black) (10, white)**. **B1** for finding the runs, **B1** for count and colour in each pair.

(b) Before: 20 pixels × 1 byte = 20 bytes. After: 3 runs × 2 bytes = 6 bytes. **M1** Saved: $20 - 6 =$ **14 bytes**. **A1**

(c) There are no repeating pixels, so every run has a count of 1. **B1** That makes 8 runs × 2 bytes = 16 bytes, which is **larger** than the 8 bytes of the original row. **B1**

Common mistakes

- **Using 1000 with the "bi" units.** 4 GiB is 4096 MiB, not 4000 MiB. KiB, MiB and GiB are binary units and always need 1024 (March 2023 report).
- **Multiplying by 1024 too many times.** 2 PiB to TiB is **one** step: $2 \times 1024 = 2048$. Many candidates multiplied 1024 by 1024 (June 2025 report). Count the steps first.
- **Four nibbles in a byte.** A nibble is 4 bits, so a byte has **2** nibbles; "4" was the most common wrong answer (June 2025 report).
- **Giving an example instead of a meaning or type.** "What is compression?" wants "reducing the size of a file", not "lossy compression" (March 2022 report). "Which type of compression?" wants lossy or lossless, not MP3 (June 2023 report). A zip folder or a file type is not a compression method (March 2025 report).
- **"Easier to send."** A compressed file is **quicker** to send, not easier (March 2022 report).
- **Mixing up lossless and lossy.** Saying lossless "removes no data" and then "removes sounds the ear can't hear" contradicts itself: removing sounds is lossy (June 2021 report). Descriptions of lossless compression often read like lossy (June 2023 report).
- **Ignoring the file type.** Explaining RLE with repeated words when the question is about an image (or pixels when it's about text) loses marks (June 2023 and June 2025 reports). Match the repeated data to the file.
- **"Unwanted" data.** Lossy removes **unnecessary** data (detail people will not notice), not "unwanted" data: the algorithm can't know what the user wants (March 2021 report).
- **Describing instead of justifying.** "Why lossless?" needs a reason for this scenario (quality, editing, printing), not how lossless works (November 2021 report).
- **Repeating the question's own reason.** If the question already gives "shorter transmission time", that answer scores nothing (June 2025 report).
- **Stopping at bits.** A file size in KiB needs ÷ 8 **and** ÷ 1024.

How to get the marks

- **"State" / "Give" / "Identify"**: one precise word or fact: "nibble", "2048", "lossless", "less storage space".
- **Calculations**: "Show all your working" means each stage earns a mark (pixels, × colour depth, ÷ 8, ÷ 1024). Put the final answer in the unit asked for, and use 1024.
- **"Describe how lossless (or lossy) compression works"**: separate points in order: algorithm → example method → what is found or removed → how it is stored → whether data is lost. Each point is a mark, so a 4-mark answer needs four different points.
- **"Explain why … lossless instead of lossy"**: name the reason for this user (quality kept, can edit, will be printed) **and** what lossy would do (permanently lose data, lower quality).
- **"Justify your choice"**: the choice gets one mark; the marks for the justification need reasons, not a description of the method.
- **Benefits and drawbacks**: give different ideas, not the same one twice ("less storage" and "less space on the server" are one idea). Link them to the context: a website, streaming, email.

Quick check

1. How many bits are in 3 KiB?
2. Give 5 PiB in TiB, and 2048 GiB in TiB.
3. An image is 800 × 600 pixels with a 16-bit colour depth. Calculate its file size in KiB.
4. A sound clip is 20 seconds long, recorded at 8000 Hz with an 8-bit sample resolution. Calculate its file size in KiB.
5. A row of 16 pixels is GGGGGWWWWWWWWGGG (G = grey, W = white). Write it using run-length encoding.
6. An online shop compresses its product photos with lossy compression before putting them on its website. Give two benefits of this and one drawback.

Answers

1. $3 \times 1024 = 3072$ bytes; $3072 \times 8 =$ **24 576 bits**.
2. $5 \times 1024 =$ **5120 TiB**; $2048 \div 1024 =$ **2 TiB**.
3. $800 \times 600 \times 16 = 7\,680\,000$ bits; $\div 8 = 960\,000$ bytes; $\div 1024 =$ **937.5 KiB**.
4. $8000 \times 8 \times 20 = 1\,280\,000$ bits; $\div 8 = 160\,000$ bytes; $\div 1024 =$ **156.25 KiB**.
5. **(5, G) (8, W) (3, G)**, or 5G8W3G: three runs instead of 16 pixels.
6. Benefits (any two): the pages load faster for customers; the photos take less storage space on the web server; less bandwidth and data allowance are used. Drawback: the photos are lower quality (data is permanently removed), and the originals can't be rebuilt from them.

Past questions to try

- **0478/12/F/M/23 Q3**: four conversions between bytes, nibbles, KiB, MiB, GiB, PiB and EiB.
- **0478/13/O/N/22 Q3**: how many thumbnail images fit in the storage space.
- **0478/12/F/M/25 Q2**: why an artist chose lossless compression, and run-length encoding described.
- **0478/12/O/N/21 Q5**: how lossy compression shrinks videos, and the drawbacks of lossless instead.