Q1) Linear vs. Non-linear multimedia + examples

- **Linear**: Fixed, no user control over order/timing. *Examples*: TV broadcast, cinema movie, radio.
- **Non-linear**: Interactive; user controls path/sequence. *Examples*: websites/hypermedia, video games, e-learning modules, YouTube with seek/menu.

Q2) Basic stages of multimedia production (define briefly)

1. **Concept/Goal** – audience, message, scope.
2. **Pre-production** – script, storyboard, asset list, budget, schedule.
3. **Production** – capture/create media (video, audio, graphics, 3D), screen design.
4. **Post-production** – editing, compositing, color/audio mix, effects.
5. **Authoring/Integration** – assemble in tool (e.g., HTML5/app), interactivity, navigation.
6. **Testing/QA** – functionality, usability, performance, compatibility.
7. **Distribution/Deployment** – web/app store/streaming, packaging.
8. **Maintenance/Updates** – fixes, content refresh.

Q3) Bitmap vs. Vector images

- **Bitmap (raster)**: Grid of pixels; resolution-dependent; enlarging causes pixelation. *Best for*: photos. *Formats*: PNG, JPG, BMP.
- **Vector**: Shapes/paths & math; resolution-independent; scales cleanly. *Best for*: logos, icons, diagrams. *Formats*: SVG, AI, EPS.

Q3 again) Lossless vs. Lossy compression

- **Lossless**: No information lost; exact original can be restored. *Examples*: PNG, FLAC, ZIP.
- **Lossy**: Discards perceptually less important data; smaller files, not perfectly reversible. *Examples*: JPEG, MP3, AAC.

Q4) Image with aspect ratio 6:2, total pixels = 480,000, grayscale

- Aspect 6:2 = **3:1** → let width = 3x, height = x.
  3x2=480,000⇒x=4003x^2 = 480{,}000 \Rightarrow x = 4003x2=480,000⇒x=400.
  Dimensions: 1200×4001200 \times 4001200×400 pixels.
- **Size** (grayscale ≈ 8 bits/pixel = 1 byte/pixel):
  480,000480{,}000480,000 bytes ≈ **480 KB** (decimal) ≈ **468.75 KiB** (binary).

Q5) Median-cut algorithm (what & steps)

- **Use**: Color quantization—reduce an image to *k* representative colors (palette).
- **Steps**:
  1. Collect all pixels in RGB space.
  2. Find the channel (R/G/B) with the largest range in the current set ("box").
  3. Sort pixels by that channel and **split at the median** → two boxes.
  4. Repeat splitting boxes until you have the required number of boxes (palette size).
  5. For each box, take the average (or median) color as the palette entry.
  6. Map each pixel to the nearest palette color.

Q6) Interlaced vs. Non-interlaced scanning

- **Interlaced**: Draws odd and even fields alternately (half lines each pass). Lower bandwidth, possible flicker/comb artifacts.
- **Non-interlaced (progressive)**: Draws all lines in order each frame. Higher quality/motion clarity; more bandwidth.

Q7) Main steps of MPEG compression (high level)

- **Prediction**: Motion estimation/compensation (P/B frames) vs. intra (I-frames) within a GOP.

- **Transform**: Block DCT on residuals (and on intra blocks).
- **Quantization**: Coefficient quantization (controls bitrate vs. quality).
- **Scan**: Zig-zag to group low-frequency coefficients.
- **Entropy coding**: Run-length + VLC/ Huffman (or CABAC in newer standards).
- **Rate control**: Bit allocation across frames/blocks.

Q8) Audio storage (time = 10 s)

Formula: **bytes = sampling_rate × bit_depth × channels × duration / 8**

a) 22.05 kHz, 8-bit, mono:
= 22,050 × 8 × 1 × 10 / 8 = **220,500 bytes** ≈ **215.33 KiB** (≈ 220.5 KB).

b) 44.1 kHz, 8-bit, mono:
= 44,100 × 8 × 1 × 10 / 8 = **441,000 bytes** ≈ **430.66 KiB** (≈ 441 KB).

c) 44.1 kHz, 16-bit, **stereo**:
= 44,100 × 16 × 2 × 10 / 8 = **1,764,000 bytes** ≈ **1.68 MiB** (≈ 1.764 MB).

d) 11.025 kHz, 16-bit, **stereo**:
= 11,025 × 16 × 2 × 10 / 8 = **441,000 bytes** ≈ **430.66 KiB** (≈ 441 KB).

Perfect 👌 I'll give you the explanation **first in English**, then the **Arabic translation** right below it.

🟦 English

Run Length Encoding (RLE) Explanation

- **Step 1:** Start reading each row from left to right.
- **Step 2:** Count the number of consecutive 0s (white pixels).
- **Step 3:** Then count the number of consecutive 1s (black pixels).
- **Step 4:** Alternate counts between white and black until the end of the row.
- **Step 5:** If the row **starts with 1**, you must write 0 first (to indicate that zero white pixels appear before the first black pixel).

Example Rows

1. Row: 011000010000110
   RLE → **1, 2, 4, 1, 4, 2, 1**
2. Row: 100000111000001
   RLE → **0, 1, 5, 3, 5, 1**
3. Row: 000001111100000
   RLE → **5, 5, 5**

🟩 Arabic (ترجمة)

شرح ضغط Run Length Encoding (RLE)

- **الخطوة 1:** اقرأ كل صف من اليسار لليمين.
- **الخطوة 2:** عد عدد الـ 0 (بكسلات بيضاء) المتتالية.
- **الخطوة 3:** بعد ذلك عد عدد الـ 1 (بكسلات سوداء) المتتالية.
- **الخطوة 4:** استمر في التبديل بين العد للأبيض والأسود حتى نهاية الصف.
- **الخطوة 5:** إذا كان الصف **يبدأ بـ 1**، يجب كتابة 0 أولاً (للدلالة أنه لا يوجد بكسلات بيضاء قبل أول بكسل أسود).

أمثلة على الصفوف

1. الصف: 011000010000110
   الناتج RLE → **1, 2, 4, 1, 4, 2, 1**
2. الصف: 100000111000001
   الناتج RLE → **0, 1, 5, 3, 5, 1**
3. الصف: 000001111100000
   الناتج RLE → **5, 5, 5**

هل تحبني أضغطلك **كل الصفوف في الصورة** بنفس الطريقة (أكتب الجدول كامل بالـ RLE لكل row)؟