ANSWERS

F13.5 Number and data questions

Exam preparation · GCSE · OCR J277 1.2.4, AQA 8525 3.3.2, Edexcel 1CP2 2.1.3 · about 15 min

What this lesson is about

Conversions, units and file sizes at speed, with the checks that catch mistakes.

- Show your working
- The five conversions
- The checks that catch slips
- Units and file sizes
- Character codes

Questions

6 marks in all

1. Convert 200 to 8-bit binary. [1 mark]

> **Answer:** 11001000. 128 + 64 + 8.

2. Convert 10110110 to hexadecimal. [1 mark]

> **Answer:** B6. 1011 is B, 0110 is 6.

3. How many different values can 8 bits hold? [1 mark]

> **Answer:** 256. One more than the largest, which is 255.

4. A 64 by 48 image at 8 bits a pixel. How many bits? [1 mark]

> **Answer:** 24576. 64 × 48 × 8.

5. What is the ASCII code for capital A? [1 mark]

> **Answer:** 65. Lower-case a is 97, which is 32 more.

6. Your binary answer has 7 digits. What should you check? [1 mark]

- ☐ **A** An 8-bit answer needs exactly 8 digits, so add the leading zero
- ☐ **B** Nothing: 7 is fine
- ☐ **C** Convert it to hex instead
- ☐ **D** Double the number

> **Answer:** A. Counting the digits catches most slips.

The task: the conversion drill

For each number in `numbers`, print `<denary> = <8-bit binary> = <2-digit hex>`, working each one out yourself rather than using `bin()`, `hex()` or `format()`. Then for the image in `image` (width, height, bits per pixel), print `image: <n> bits` and `image: <n> kB`, and for `sound` (rate, seconds, bit depth) print `sound: <n> bits` and `sound: <n> kB`. Give the kB to 1 decimal place.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

numbers = [5, 77, 200, 255]
image = (64, 48, 8)        # width, height, bits per pixel
sound = (8000, 5, 8)       # samples a second, seconds, bits a sample
```

The hint students can ask for: For the binary, work down the place values from 128, taking each one that fits. For the hex, the first digit is how many sixteens and the second is what is left, looked up in a string of digits. A size in bits is the three numbers multiplied; in kB it is that divided by eight and then by a thousand.

A solution

```
from bugbot import *
connect()
numbers = [5, 77, 200, 255]
image = (64, 48, 8)
sound = (8000, 5, 8)
DIGITS = "0123456789ABCDEF"

def to_binary(n):
    bits = ""
    for value in [128, 64, 32, 16, 8, 4, 2, 1]:
        if n >= value:
            bits = bits + "1"
            n = n - value
        else:
            bits = bits + "0"
    return bits

def to_hex(n):
    return DIGITS[n // 16] + DIGITS[n % 16]

for n in numbers:
    print(f"{n} = {to_binary(n)} = {to_hex(n)}")
bits = image[0] * image[1] * image[2]
print("image:", bits, "bits")
print(f"image: {bits / 8 / 1000:.1f} kB")
bits = sound[0] * sound[1] * sound[2]
print("sound:", bits, "bits")
print(f"sound: {bits / 8 / 1000:.1f} kB")
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.