Solutions

Each answer shows the steps, then the **result**.

2.1

- $16 + 8 + 1 =$ **25**

2.2

- $37 = 32 + 4 + 1$: `100101`

2.3

- $2^6 =$ **64** values
- largest: $64 - 1 =$ **63**

2.4

- $3 \times 8 =$ **24** bits

3.1 A

- $16 + 4 + 2 = 22$

3.2 D

- 7 bits hold 0 to $2^7 - 1 = 127$, so 128 overflows

3.3 B

- `1111` is 15 and `10000` is 16, so 14, 15, 16

3.4

(a)

- $2^5 - 1 =$ **31**
- `11111`

(b)

- 35 is larger than 31, the largest 5-bit value
- an **overflow error**: the value cannot be stored correctly

(c)

- readings are **samples** taken once a minute and rounded to whole numbers, so changes between samples and fractions of a degree are **lost**
- any one: take samples **more often**; store each reading with **more bits** / decimal places

3.5 A

- 4 bytes is 32 bits, so the largest value is about 2^{31} (signed) or 2^{32}; 2^{40} is too large for X but fits in Y

3.6

(a)

- **128**
- positions are numbered from 0 at the right, so the leftmost of 8 bits is position 7, and $2^7 = 128$

(b)

- the same 0s and 1s can stand for sound, so the details of how the sound is stored are hidden and the programmer works with numbers