Floating point kerfuffle

A tech startup that specialised in making coffee machines with built-in AI assistants receives a bug report. The machine's “smart pour” feature, which dispenses liquid based on voice input (e.g. “pour me 0.1 litres”), keeps dispensing 0.099609375 litres instead of 0.1 litres.

The engineers trace the issue to the machine's use of a custom 10-bit floating point format (called MiniFloat) for internal calculations, defined as follows:

- S: 1 sign bit
- E: 4 exponent bits (with a bias of 8)
- B: 5 significand bits

So the value of a number is: $(-1)^{S} \times B \times 2^{E-8}$

To illustrate, the bit pattern 1110001101 gives

$$(-1)^{1} \times 0.1101 \times 2^{1100-1000}$$

$$= \; -1 \times 0.0110 \times 2^{4} = -\frac{3}{8} \times 16 = -6$$

(a) Given that the binary representation of 0.1 is 0.000110011…, approximate 0.1 as a MiniFloat number.

Answer: $0100110011 \; = \; (-1)^{0} \times 1.0011 \times 2^{0011}$

(b) Why is there a discrepancy between the requested amount poured and the actual amount poured?

Answer: binary cannot exactly represent 0.1, especially with limited bits

(c) Which of the following decimals is representable in MiniFloat format?

- 0.2
- 0.375
- 0.8
- 0.12