True or false? (unless otherwise specified)

1. Let $(\bullet)_2$ denote binary representation. It holds that
$$(0.1011)_2 + (0.0101)_2 = 1.$$
2. Let $(\bullet)_7$ denote base 7 representation. It holds that
$$(1000)_7 \times (2)_7 = (2000)_7.$$
3. Let $p \in \mathbf{N}$. The set $\big\{(b_0.b_1b_2\dots b_{p-1})_2 : b_i \in \{0,1\}\big\} \subset \mathbf{R}$ contains 2^p distinct real numbers.
4. The machine epsilon of a floating point format is the smallest strictly positive number that can be represented exactly in the format.
5. Let ε denote the machine epsilon for the `Float64` format. Any $x \in \mathbf{R}$ such that $-\varepsilon < x < \varepsilon$ cannot be represented in the `Float64` format.
6. Machine multiplication is commutative, meaning that $a \mathbin{\widehat{*}} b = b \mathbin{\widehat{*}} a$ for any `Float64` point numbers a and b.
7. If `x` is a `Float16` and `y` is a `Float64` number, then the result of `x + y` is a `Float64` number.
8. The real number `0.0` can be represented exactly in the `Float32` format.
9. It holds that
$$(0.\overline{011})_2 = \frac{3}{4}.$$
10. In Julia, `Float64(x) == Float32(x)` is `true` if `x` is a rational number.
11. The value of the machine epsilon for `Float64` format is the same as for the `Float32` format.
12. The spacing (in absolute value) between successive double-precision (`Float64`) floating point numbers is always equal to the machine epsilon.
13. **All** the natural numbers can be represented exactly in the double precision floating point format `Float64`.
14. Machine addition in the `Float64` format is associative but not commutative.
15. In Julia, `Float64(.4) == Float32(.4)` evaluates to `true`.
16. **(Bonus)** In Julia `10 + eps() == 1` evaluates to `true`.
Explain briefly why:

17. **(Bonus)** In Julia `sqrt(1 + eps()) == 1 + eps()` evaluates to `false`.
Explain briefly why:

Answer key

1. **True.** $(0.1011)_2 = 1/2 + 1/8 + 1/16 = 11/16 = 0.6875$ and $(0.0101)_2 = 1/4 + 1/16 = 5/16 = 0.3125$, whose sum is 1.

2. **True.** In base 7, multiplying by $(2)_7$ multiplies the value by 2: $(1000)_7 = 7^3$, and $2 \cdot 7^3 = (2000)_7$.

3. **False.** There are *two* choices for b_0 in addition to the p fractional bits, so the set contains 2^{p+1} distinct real numbers, not 2^p.

4. **False.** The machine epsilon is the distance between 1 and the next representable number. The smallest strictly positive number in the format can be much smaller (a subnormal number).

5. **False.** For example, the numbers in the subnormal range near 2^{-1074} are much smaller in magnitude than $\varepsilon \approx 10^{-16}$ and are all representable in the `Float64` format.

6. **True.** The rounding function is symmetric in its arguments, so machine multiplication is commutative.

7. **True.** `Float16` and `Float64` are promoted to `Float64` before the addition, so the result is a `Float64`.

8. **True.** The number `0.0` (with zero significand and any exponent) is representable in every format, in particular in `Float32`.

9. **False.** Let $S = (0.\overline{011})_2$. The repeating block is 011, whose value is 3/8, so $S = (3/8) \cdot (1 + 1/8 + 1/64 + \cdots) = (3/8)/(1 - 1/8) = 3/7$. The number equals 3/7, not 3/4.

10. **False.** Most rational numbers (e.g. $x = 0.1$) are not exactly representable, and the nearest `Float64` and `Float32` values generally differ.

11. **False.** `eps(Float64) = 2^{-52}` while `eps(Float32) = 2^{-23}`; the two values differ.

12. **False.** The spacing between consecutive numbers grows with the exponent: it equals ε only near 1, and is smaller (or larger) elsewhere.

13. **False.** Integers larger than 2^{53} cannot all be represented exactly, because the `Float64` mantissa has only 53 bits.

14. **False.** Machine addition in `Float64` is commutative but *not* associative (rounding can make the two groupings differ).

15. **False.** The decimal number 0.4 is not exactly representable in binary, and its nearest `Float64` and `Float32` values differ, so the comparison evaluates to `false`.

16. **False.** `eps()` is much smaller than 10, so `10 + eps()` rounds back to `10.0`, which is not equal to 1.

17. **True.** The true value of $\sqrt{1+\varepsilon}$ lies just below the midpoint $1+\varepsilon/2$ between two consecutive representable numbers (the spacing near 1 is ε), so it rounds (by round-to-nearest-even) to `1.0`, while `1 + eps()` equals $1 + \varepsilon$. Hence the two values differ and the comparison is `false`.