HW3.1. Floating Point Theory

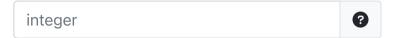
Feel free to check out the guide that we have prepared to help you in this problem.

For the following question, we will be following the IEEE-754 Floating Point standard *but with different numbers of bits:* with 6 bits of exponent, an exponent bias of -31, and 25 bits of mantissa. Due to PrairieLearn restrictions, please submit your answer as the full number.

Floating point representations present a trade-off between range and accuracy. Due to the limited significant digits that you can represent, there will be a point where integers cannot be exactly represented anymore.

Q1.1: What's the smallest non-infinite positive integer (whole number) this representation CANNOT represent?

Hint: Think about how your floating point value changes when you change the mantissa by 1.



The floating point standard optimizes the representation of numbers by not including the implied 1 for the 'normal' binary representations.

Q1.2: What power of 2 is the smallest representable positive normalized number? Submit the exponent only.



Floating point also allows for representations of numbers even smaller than the smallest normalized number. Denormal numbers utilize an implied 0, instead of an implied 1.

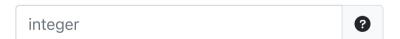
Q1.3: What power of 2 is the smallest representable positive value? Submit the exponent only.



Floating point also has support for special values that can be returned as the result of invalid operations.

Q1.4: How many NaNs can be represented with this floating point system?

Hint: Recall how NaNs are represented following the floating point standard.

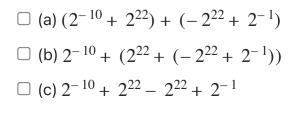


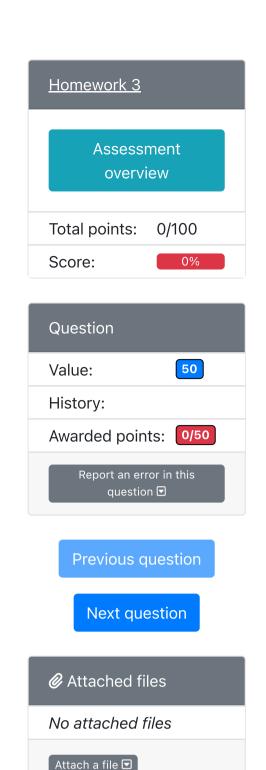
Q1.5: What's the most negative possible denormalized number representable in this system? Write the value in IEEE-754 hexadecimal notation (with uppercase letters), including the "0x" prefix.



The next few questions explore some quirkier aspects of floating-point numbers.

Q2.1: Addition and subtraction on unsigned and two's complement numbers is both commutative and associative. (You can arbitrarily re-order a series of additions and subtractions and get the same result.) In contrast, floating point addition is **not** associative, because between each substep, the result loses a small amount of precision. Given the floating point encoding from earlier, which of the following sequence of floating point additions and subtractions returns the correct result?





Attach text 모

 \Box (d) $2^{-10} + (2^{22} - 2^{22}) + 2^{-1}$

Select all possible options that apply.

Q3.1: An IEEE-754 **single-precision** floating point number is 32 bits, consisting of 1 sign bit, 8 exponent bits, and 23 mantissa bits. Can this floating-point encoding represent the space of all integers a 32-bit, two's complement number can represent?

- (a) No
- (b) Yes

Save & Grade 20 attempts left

Save only

Additional attempts available with new variants 🔞