Rules for Conversion of Real Numbers to Floating Point

1. Zero is converted to zero.
2. For a non-zero value, write it as $\pm .b_1 b_2 \cdots b_t b_{t+1} \cdots \times \beta^k$ with $b_1 \neq 0$ and at least t+1 base-β digits expressed.
3. If using truncating conversion, drop digits $b_{t+1} \cdots$. If rounding, drop digits $b_{t+1} \cdots$ ***if*** $b_{t+1} < \beta/2$. Otherwise, if $b_{t+1} \geq \beta/2$, increase digit b_t (and if this generates a carry, keep increasing digits until the carry is resolved.)
4. Check the exponent. If $k > U$, the number overflows (and has no converted value). If $k < L$, the number underflows (and most systems convert the number to zero).

Rules for Floating Point Arithmetic

1. Convert all real values in an expression to floating point.
2. In the appropriate order, do every arithmetic operation exactly and then convert the result to floating point before doing any subsequent operation. Stop if any result overflows or underflows (unless underflows are converted to zeros).

Floating Point Arithmetic Exercises

1. Assume that $\beta = 10$, t = 3, L = -3, U = 4, and that the arithmetic is *truncating*.

   A. What is *fl*(.00009)?
   B. What is *fl*(3.146)?
   C. What is *fl*(9996.)?
   E. What are *fl*((100.+.61)+.61) and *fl*(100.+(.61+.61))?
   F. What are *fl*($2.34 \times (5.67 + 8.90)$) and *fl*($(2.34 \times 5.67) + (2.34 \times 8.90)$)?

2. Repeat #2, assuming $\beta = 10$, t = 3, L = -3, U = 4, and that the arithmetic is *rounding*.

3. Determine the largest floating point number.

4. Determine the smallest positive floating point number.

5. Determine the largest floating point number smaller than one.

6. Determine the largest floating point number greater than one.

7. With $\beta = 10$, t = 3, L = -3, U = 4, and *truncating* arithmetic, discover an example where *fl*((a+b)/2) is strictly less than both *a* and *b* (where both *a* and *b* are floating point numbers). Find such an example for *rounding* arithmetic.