Exercises such as this one help you develop your ability to reason about floatingpoint
operations from a programmer's perspective. Make sure you understand
each of the answers.
A. x == (int)(double) x
Yes, since double has greater precision and range than int.
B. x == (int)(float) x
No. For example, when x is *TMax*.
C. d == (double)(float) d

No. For example, when d is 1e40, we will get $+\infty$ on the right.
D. f == (float)(double) f
Yes, since double has greater precision and range than float.
E. f == -(-f)
Yes, since a floating-point number is negated by simply inverting its sign bit.

F. 1.0/2 == 1/2.0
Yes, the numerators and denominators will both be converted to floatingpoint
representations before the division is performed.
G. d*d >= 0.0

Yes, although it may overflow to $+\infty$.
H. (f+d)-f == d
No. For example, when f is 1.0e20 and d is 1.0, the expression f+d will be
rounded to 1.0e20, and so the expression on the left-hand side will evaluate
to 0.0, while the right-hand side will be 1.0.