# Power function

Here is a static method for computing $x^n$ for $x$ real (double in Java) and $n$ non-negative integer.

```
class Power {

    static double power(double x, int n) {
        double r = 1.0;
        double t = x;
        while (n > 0) {
            if (n % 2 == 1) r = r * t;
            t = t * t;
            n = n / 2;
        }
        return r;
    }

}
```

1. Run the tool without any annotations. What are you asked to prove? Use the krakatoa pragma for removing arithmetic overflow checks. Comment about interpretation of floating-point numbers.

2. Add annotations for proving termination. Hint: you might add a lemma about division by 2.

3. Add an assertion in the code to show that $n$ is zero just before the return statement and prove it. Hint: add also a loop invariant.

4. Introduce an axiomatic block for declaring the mathematical function `lpower` over reals, with integer exponents, and use this function to specify the expected behavior of method `power`. Check the syntax of your annotations using krakatoa.

5. Add a loop invariant suitable for proving the behavior. Hints: you have to add the necessary axioms in the axiomatic block for `lpower`. Try to prove the post-condition first. Depending on the back-end prover, you might need to add extras lemmas about integer multiplication.