A number x is called a *fixed point* of a function f if $f(x) = x$. For some functions f we can locate a fixed point by beginning with an initial guess and then applying f repeatedly

$$x, f(x), f(f(x)), f(f(f(x))), \dots$$

until the value does not change anymore (or the change is within a small tolerance).

1. Finding a fixed point

Write in Scala a recursive function fixedPoint that given a function f and an initial guess tries to find a fixed point for f using the technique explained above. Its signature will be:

```
def fixedPoint(f: (Double) Double, guess: Double): Double;
```

Two successive values x and y will be considered as close enough if $|(x - y)/x|$ is less than the tolerance (you could wrap this test into a function isCloseEnough).

Experiment on some examples (for instance on affine functions when the gradient is smaller than 1 like $x \mapsto 0.5x + 2$).

2. Square root

Here is a specification of the $sqrt$ function:

$$\begin{aligned} sqrt(x) &= \text{the } y \text{ such that } y * y = x \\ &= \text{the } y \text{ such that } y = x/y \end{aligned}$$

This suggests that $sqrt(x)$ can be computed by fixed point iteration. Write such a function and try it.

3. Average damping

You should have noticed that the function does not converge.

Instrument the fixed point function with a print statement which keeps track of the current guess value and observe what actually happens.

To make the iteration converge we can average successive values. This is exactly what we get if we pass the following function to fixedPoint:

$$x \mapsto (x + f(x))/2$$

instead of the function f we had initialy in mind.

This technique of *average dampening* is so general that it can be wrapped in a function averageDamp. Its signature would be:

```
def averageDamp(f: (Double) Double): (Double) Double;
```

Write such a function and modify the function sqrt to use it.

4. Cube root

Write a function for computing cube roots using fixedPoint and averageDamp.