Example: Square roots with Newton’s method
We will define in this session a function

```scala
/** Calculates the square root of parameter x */
def sqrt(x: Double): Double = ...
```

The classical way to achieve this is by successive approximations using Newton’s method.
To compute $\sqrt{x}$:

- Start with an initial estimate $y$ (let’s pick $y = 1$).
- Repeatedly improve the estimate by taking the mean of $y$ and $x/y$.

Example:

<table>
<thead>
<tr>
<th>Estimation</th>
<th>Quotient</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$2 / 1 = 2$</td>
<td>1.5</td>
</tr>
<tr>
<td>1.5</td>
<td>$2 / 1.5 = 1.333$</td>
<td>1.4167</td>
</tr>
<tr>
<td>1.4167</td>
<td>$2 / 1.4167 = 1.4118$</td>
<td>1.4142</td>
</tr>
<tr>
<td>1.4142</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
First, define a function which computes one iteration step

```scala
def sqrtIter(guess: Double, x: Double): Double = 
  if (isGoodEnough(guess, x)) guess 
  else sqrtIter(improve(guess, x), x)
```

Note that `sqrtIter` is *recursive*, its right-hand side calls itself.

Recursive functions need an explicit return type in Scala.

For non-recursive functions, the return type is optional
Second, define a function `improve` to improve an estimate and a test to check for termination:

```scala
def improve(guess: Double, x: Double) =
  (guess + x / guess) / 2

def isGoodEnough(guess: Double, x: Double) =
  abs(guess * guess - x) < 0.001
```
Third, define the sqrt function:

```scala
def sqrt(x: Double) = sqrtIter(1.0, x)
```
1. The `isGoodEnough` test is not very precise for small numbers and can lead to non-termination for very large numbers. Explain why.

2. Design a different version of `isGoodEnough` that does not have these problems.

3. Test your version with some very very small and large numbers, e.g.

   0.001
   0.1e-20
   1.0e20
   1.0e50