All variables are ints.

PRECONDITION: \( x > 0 \)

\[
z = 0; \\
y = x;
\]

while ( \( y \% 10 == 0 \) ) {
\[
\begin{align*}
    y &= y / 10 & // integer division \\
    z &= z + 1 \\
\end{align*}
\]
}

POSTCONDITION: \{ \( x = y \cdot 10^z \land (y \% 10 \neq 0) \) \}

Example: \( x = 32000; x = 32 \cdot 10^3 \)

\( LI: \{ x = y \cdot 10^z \} \)

\begin{align*}
    \text{Base case:} & \\
    y &= x \land z = 0 \\
    \{ x = y \cdot 10^0 \} &= \{ y = x \cdot 1 \} = \{ y = x \}
\end{align*}

\( \text{Iteration } k: \)

\( \text{assume: } x = y_k \cdot 10^z \)

\[
\begin{align*}
    y_{k+1} &= \frac{y_k}{10} \\
    z_{k+1} &= z_k + 1 \\
    \{ y_{k+1} \cdot 10^{z_{k+1}} = \frac{y_k}{10} \} &= \{ y_k \cdot 10^{z_k} = x \}
\end{align*}
\]

At exit:

\[
\{ !(y \% 10 == 0) \land x = y \cdot 10^z \} \rightarrow \{ y \% 10 \neq 0 \land x = y \cdot 10^z \}\]
Termination:
One choice for D₁ = the number of trailing 0’s in y
D₁ = String.valueOf(y).length() -
    String.valueOf(y).replaceAll("0*$",""").length();

D₁ = number of trailing zeros in y, didn’t equal 0 or we would have exited loop
D₁new = number of trailing zeros in ynew
ynew = yold/10
D₁new has 1 less trailing 0. (requires yold > 0 from precondition)
D₁new < D₁old

At Exit: D₁ = 0 => no trailing zeros => y % 10 ≠ 0

Alternative: D₂ = (y % 10 == 0 ? 1 : 0) * floor(log₁₀(y))
floor(log₁₀(y)) is one less than the number of digits in y – requires y>0
i.e. only works for precondition x > 0.

D₂old = (yold % 10 == 0 ? 1 : 0) * floor(log₁₀(yold)) = floor(log₁₀(yold))  //
yold % 10 is not zero or we would exit
D₂new = (ynew % 10 == 0 ? 1 : 0) * floor(log₁₀(ynew))
    = (yold/10 % 10 == 0 ? 1 : 0) * floor(log₁₀(yold/10))
    = 1 * (floor(log₁₀(yold)) – floor(log₁₀(10))), if (yold/10 % 10 == 0)
    = D₂old – 1, if (yold/10 % 10 == 0)
    = 0, otherwise.
D₂new decreases either way.

D₂ = 0 implies exit condition:
D₂ = 0 => y % 10 ≠ 0 v floor(log₁₀(y)) = 0
    floor(log₁₀(y)) = 0  => (1 <= y < 10)
    => y % 10 ≠ 0.
Alternative: \( D_3 = \text{floor}(\log_{10}(y)) \)

\( \text{floor}(\log_{10}(y)) \) is one less than the number of digits in \( y \) – requires \( y > 0 \)
i.e. only works for precondition \( x > 0 \).

\[
\begin{align*}
D_{3\text{old}} &= \text{floor}(\log_{10}(y_{\text{old}})) \\
D_{3\text{new}} &= \text{floor}(\log_{10}(y_{\text{new}})) \\
&= \text{floor}(\log_{10}(y_{\text{old}}/10)) \\
&= \text{floor}(\log_{10}(y_{\text{old}})) - \text{floor}(\log_{10}(10)) \\
&= D_{3\text{old}} - 1
\end{align*}
\]

Therefore, \( D_{3\text{new}} < D_{3\text{old}} \).

\( D_3 = 0 \) implies exit condition:

\[
D_3 = 0 \Rightarrow \text{floor}(\log_{10}(y)) = 0 \Rightarrow (1 \leq y < 10) \Rightarrow y \ % \ 10 \neq 0.
\]

Notice that to prove termination, we need to prove that *if* \( D = 0 \), *then* the loop exits. Not necessarily the converse, i.e., it is not necessary that every time the loop terminates, \( D \) must be equal to 0. For example, if \( x = 32000 \), \( D_3 = 1 \) when the loop terminates.

As long as \( D \)'s range is a well-ordered set, and \( D \) strictly decreases at every iteration, it must eventually reach the minimum value (e.g., 0), and to prove loop termination, it suffices to prove that for that minimum value, the loop exits.
Alternative: \( D_4 = \text{floor}(\log_{10}(x)) - z \)

floor(\( \log_{10}(x) \)) is one less than the number of digits in \( x \) (or maximum possible number of zeros) – requires \( x > 0 \), i.e. it works for precondition \( x > 0 \).

\( z \) is the accumulator of zeros in \( x \).

\[
\begin{align*}
D_{4\text{old}} &= \text{floor}(\log_{10}(x)) - z_{\text{old}} \\
D_{4\text{new}} &= \text{floor}(\log_{10}(x)) - z_{\text{new}} \\
&= \text{floor}(\log_{10}(x)) - (z_{\text{old}} + 1) \\
&= D_{4\text{old}} - 1
\end{align*}
\]

Therefore, \( D_{4\text{new}} < D_{4\text{old}} \).

\( D_4 = 0 \) implies exit condition:
\( D_4 = \text{floor}(\log_{10}(x)) - z \)

If \( D_4 = 0 \Rightarrow z = \text{floor}(\log_{10}(x)) \).

\[
\begin{align*}
x &= y \times 10^z \text{ (from LI)} \\
x &= y \times 10^{\text{floor}(\log_{10}(x))} \\
\log_{10}(x) &= \log_{10}(y \times 10^{\text{floor}(\log_{10}(x))}) \\
\log_{10}(x) &= \log_{10}(y) + \log_{10}(10^{\text{floor}(\log_{10}(x))}) \\
\log_{10}(x) &= \log_{10}(y) + \text{floor}(\log_{10}(x)) \\
\log_{10}(x) - \text{floor}(\log_{10}(x)) &= \log_{10}(y) \\
\Rightarrow 0 &\leq \log_{10}(y) < 1 \\
\Rightarrow 1 &\leq y < 10 \\
\Rightarrow y \pmod{10} &\neq 0
\end{align*}
\]
gcd is the greatest common divisor of two positive integers, i.e. the largest integer number that evenly divides both numbers.

**PRECONDITION:** \{ x_1 > 0 \land x_2 > 0 \}

y_1 = x_1;  
y_2 = x_2;

while ( y_1 != y_2 ) {
    if ( y_1 > y_2 ) {
        y_1 = y_1 - y_2;
    } else {
        y_2 = y_2 - y_1;
    }
}

**POSTCONDITION:** \{ y_1 = \text{gcd}( x_1, x_2 ) \}

Some gcd facts:

\text{gcd}(x,x) = x

\text{gcd}(x,y) = \text{gcd}(x-y, y)

Proof:

x=ad, y=bd

x-y = ad - bd = (a-b)d \implies d \text{ is a divisor of } x-y, \text{ as well as } x \text{ and } y

At exit, we want y_1 = \text{gcd}(x,y) \land y_1 = y_2 \text{ (exit condition)}

since \text{gcd}(y_1,y_1) = y_1 \text{ and at exit } y_2=y_1, \text{ a good guess might be}

LI: \text{gcd}(y_1, y_2) = \text{gcd}(x_1, x_2)

Let’s see if it works
Initial step:
y_1 = x_1, y_2 = x_2;
gcd(y_1, y_2) = gcd(x_1, x_2)

Iteration k+1:

\textit{assume : } gcd(y_{1_k}, y_{2_k}) = gcd(x_1, x_2)
y_{1_k} < y_{2_k}
y_{1_{k+1}} = y_{1_k} - y_{2_k}
gcd(y_{1_{k+1}}, y_{2_{k+1}}) = gcd(y_{1_{k}} - y_{2_{k}}, y_{2_{k}}) = gcd(y_{1_{k}}, y_{2_{k}}) = gcd(x_1, x_2)

Similar proof for y_2 > y_1. If y_1 = y_2, we exit loop.

At Exit:
!(y_1 \neq y_2) \land gcd(y_1, y_2) = gcd(x_1, x_2)
\Rightarrow (y_1 = y_2) \land gcd(y_1, y_2) = gcd(x_1, x_2)
\Rightarrow gcd(y_1, y_1) = gcd(x_1, x_2)
\Rightarrow y_1 = gcd(x_1, x_2)

Termination:
At each iteration, we choose max(y_1, y_2). At the end, y_1 = y_2 = gcd(x_1, x_2).
A reasonable choice for D might be:
D = max(y_1, y_2) - gcd(y_1, y_2)
The minimum is 0 and it should decrease at each iteration.
Minimum occurs when y_1 = y_2

\begin{align*}
D_k &= max(y_{1_k}, y_{2_k}) - gcd(y_{1_k}, y_{2_k}) \\
D_{k+1} &= max(y_{1_{k+1}}, y_{2_{k+1}}) - gcd(y_{1_{k+1}}, y_{2_{k+1}}) \\
y_{1_k} &> y_{2_k} \\
D_{k+1} &= max(y_{1_k} - y_{2_k}, y_{2_k}) - gcd(y_{1_k} - y_{2_k}, y_{2_k}) < max(y_{1_k}, y_{2_k}) - gcd(y_{1_k}, y_{2_k}) \\
\therefore D_{k+1} &< D_k
\end{align*}
// reduce y_{1_k}, since y_{2_k} > 0 by precondition.
Similar proof for $y_{2k} > y_{1k}$.

At exit:
$D = 0$

$\Rightarrow \max(y_1, y_2) - \gcd(y_1, y_2) = 0$

$\Rightarrow \max(y_1, y_2) = \gcd(y_1, y_2) \Rightarrow y_1 = y_2$