All variables are ints.

PRECONDITION: x > 0

```
z = 0;
y = x;

while ( y % 10 == 0 ) {
  y = y / 10   // integer division
  z = z + 1
}
```

POSTCONDITION: { x == y * 10^z && (y % 10 != 0) }

Example: x = 1100; x = 11*10^2

$LI: \{x == y*10^z\}$

$Base\,case:$

$y == x \,\&\&\, z == 0$

$\{x == y*10^0\} == \{y == x*1\} == \{y == x\}$

$Iteration\,k:$

$assume: x == y_k * 10^{z_k}$

$y_{k+1} = \frac{y_k}{10}$

$z_{k+1} = z_k + 1$

$\{y_{k+1} * 10^{z_{k+1}} == \frac{y_k}{10} * 10^{z_k+1}\} == \{y_k * 10^{z_k} == x\}$

$At\,exit:$

$\{!(y\%10 == 0) \,\&\&\, x == y*10^z\} \rightarrow \{y\%10 \neq 0 \,\&\&\, x == y*10^z\}$

Termination:

One choice for D = the number of trailing 0's in y

D = String.valueOf(y).length() -
String.valueOf(y).replaceAll("0*$","").length();

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

D = number of trailing zeros in y, didn’t equal 0 or we would have exited loop
D_new = number of trailing zeros in y_new
y_new = y_old/10
D_new has 1 less trailing 0.
D_new < D_old

D = (y % 10 == 0 ? 1 : 0) * int(log10(y)) = int(log10(y))   // D is not zero or we would exit
D_new = (y_new % 10 == 0 ? 1 : 0) * int(log10(y_new))
            = (y/10 % 10 == 0 ? 1 : 0) * int(log10(y/10))
            = 1 * (int(log10(y)) – int(log10(10)))
            = D – 1 if first term was 1, otherwise D_new = 0 and D decreases either way.

At Exit:
D = 0 => no trailing zeros => y % 10 != 0

Using the other decrement function
D = 0 => y % 10 != 0

gcd is the greatest common divisor of two positive integers, i.e. the largest integer number that evenly divides both numbers.

```
PRECONDITION: { x1 > 0 && x2 > 0 }

y1 = x1
y2 = x2

while ( y1 != y2 ) {
  if ( y1 > y2 ) {
    y1 = y1 - y2
  }
  else {
    y2 = y2 - y1
  }
}

POSTCONDITION: { y1 == gcd( x1, x2 ) }
```

Some gcd facts:

gcd(x,x)=x

gcd(x,y) = gcd(x-y, y)

Proof:

x=ad, y=bd

x-y = ad – bd = (a-b)d => d is a divisor of x-y, as well as x and y

At exit, we want y1==gcd(x,y) && y1 == y2 (exit condition)

since gcd(y1,y1) == y1 and at exit y2==y1, a good guess might be

LI: gcd(y1, y2) = gcd(x1, x2)

Let’s see if it works

Initial step:
y1 = x1, y2 = x2;
gcd(y1, y2) = gcd(x1, x2)

Iteration k+1:

$$
\begin{aligned}
&assume: \gcd(y1_k, y2_k) = \gcd(x1, x2) \\
&y1_k < y2_k \\
&y1_{k+1} = y1_k - y2_k \\
&\gcd(y1_{k+1}, y2_{k+1}) = \gcd(y1_k - y2_k, y2_k) = \gcd(y1_k, y2_k) = \gcd(x1, x2)
\end{aligned}
$$

Similar proof for y2> y1. If y1==y2, we exit loop.

At Exit:
!(y1 != y2) && gcd(y1,y2)==gcd(x1,x2)
=> (y1 == y2) && gcd(y1, y2) == gcd(x1,x2)
=> gcd(y1,y2) == gcd(x1, x2)
=> y1 = gcd(x1,x2)

Termination:
At each iteration, we choose max(y1, y2). At the end, y1 == y2 == gcd(x1,x2).
A reasonable choice for D might be:
D = max(y1,y2) – gcd(y1,y2)
The minimum is 0 and it should decrease at each iteration.
Minimum occurs when y1 == y2

$$
\begin{aligned}
&D_k = \max(y1_k, y2_k) - \gcd(y1_k, y2_k) \\
&D_{k+1} = \max(y1_{k+1}, y2_{k+1}) - \gcd(y1_{k+1}, y2_{k+1}) \\
&y1_k > y2_k \\
&D_{k+1} = \max(y1_k - y2_k, y2_k) - \gcd(y1_k - y2_k, y2_k) < \max(y1_k, y2_k) - \gcd(y1_k, y2_k) \\
&\therefore D_{k+1} < D_k
\end{aligned}
$$

  // reduce larger y

Similar proof for y2>y1.

At exit:
D == 0
max(y1, y2) - gcd(y1, y2) ==0
max(y1, y2) == gcd(y1, y2) => y1 == y2