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 = 32000; $x = 32*10^3$

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

$Base\,case:$

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

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

$Iteration\,k:$

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

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

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

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

$At\,exit:$

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

Termination:
One choice for $D_1$ = the number of trailing 0's in y
$D_1$ = String.valueOf(y).length() -
String.valueOf(y).replaceAll("0*$","").length();

$D_1$ = number of trailing zeros in y, didn't equal 0 or we would have exited loop
$D_{1new}$ = number of trailing zeros in $y_{new}$
$y_{new} = y_{old}/10$
$D_{1new}$ has 1 less trailing 0. (requires $y_{old} > 0$ from precondition)
$D_{1new} < D_{1old}$

At Exit: $D_1 = 0$ => no trailing zeros => $y \% 10 \neq 0$

Alternative: $D_2 = (y \% 10 == 0 ? 1 : 0) * \text{floor}(\log_{10}(y))$
$\text{floor}(\log_{10}(y))$ is one less than the number of digits in y – requires y>0
i.e. only works for precondition x > 0.

$D_{2old} = (y_{old} \% 10 == 0 ? 1 : 0) * \text{floor}(\log_{10}(y_{old})) = \text{floor}(\log_{10}(y_{old}))$ // $y_{old} \% 10$ is not zero or we would exit

$D_{2new} = (y_{new} \% 10 == 0 ? 1 : 0) * \text{floor}(\log_{10}(y_{new}))$
$= (y_{old}/10 \% 10 == 0 ? 1 : 0) * \text{floor}(\log_{10}(y_{old}/10))$
$= 1 * (\text{floor}(\log_{10}(y_{old})) - \text{floor}(\log_{10}(10)))$, if $(y_{old}/10 \% 10 == 0)$
$= D_{2old} - 1$, if $(y_{old}/10 \% 10 == 0)$
$= 0$, otherwise.

$D_{2new}$ decreases either way.

$D_2 = 0$ implies exit condition:
$D_2 = 0 \Rightarrow y \% 10 \neq 0 \ \text{v} \ \text{floor}(\log_{10}(y)) = 0$
$\text{floor}(\log_{10}(y)) = 0 \Rightarrow (1 <= y < 10)$
$\Rightarrow y \% 10 \neq 0.$

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.

$D_{3old} = \text{floor}(\log_{10}(y_{old}))$
$D_{3new} = \text{floor}(\log_{10}(y_{new}))$
$= \text{floor}(\log_{10}(y_{old}/10))$
$= \text{floor}(\log_{10}(y_{old})) – \text{floor}(\log_{10}(10)))$
$= D_{3old} – 1$
Therefore, $D_{3new} < D_{3old}$.

$D_3 = 0$ implies exit condition:
$D_3 = 0 \Rightarrow \text{floor}(\log_{10}(y)) = 0 \Rightarrow (1 <= y < 10)$
$\Rightarrow y \% 10 \neq 0$.

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: $D_4 = floor(log_{10}(x)) - z$
$floor(log_{10}(x))$ 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.

$D_{4old} = floor(log_{10}(x))-z_{old}$
$D_{4new} = floor(log_{10}(x))-z_{new}$
$= floor(log_{10}(x))-(z_{old}+1)$
$= D_{4old} – 1$
Therefore, $D_{4new} < D_{4old}$.

$D_4 = 0$ implies exit condition:
$D_4 = floor(log_{10}(x)) – z$

If $D_4 = 0 \Rightarrow z = floor(log_{10}(x))$.
$x = y * 10^z$ (from LI)
$x = y * 10^{floor(log_{10}(x))}$
$log_{10}(x) = log_{10}(y * 10^{floor(log_{10}(x))})$
$log_{10}(x) = log_{10}(y) + log_{10}(10^{floor(log_{10}(x))})$
$log_{10}(x) = log_{10}(y) + floor(log_{10}(x))$
$log_{10}(x) – floor(log_{10}(x)) = log_{10}(y)$

- ⇨ $0 <= log_{10}(y) <1$
- ⇨ $1 <= y <10$
- ⇨ $y \% 10 \neq 0$

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:

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

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,y1) = 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

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

// reduce $y1_k$, since $y2_k > 0$ by precondition.

Similar proof for $y2_k > y1_k$.

At exit:
D = 0

- ⇨ max(y1, y2) - gcd(y1, y2) = 0
- ⇨ max(y1, y2) = gcd(y1, y2) => y1 = y2