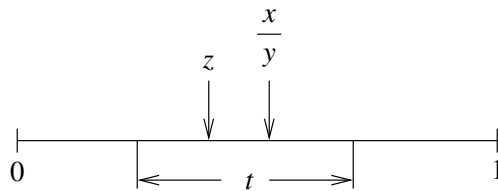


Find the *quotient* z of real numbers $0 \leq x < y \leq 1$ to within tolerance t (without using division, of course).

$$\{0 \leq x < y \leq 1\} \text{ S } \{z \leq x/y < z + t\}$$



Programming strategy:

```

{0 ≤ x < y ≤ 1}
begin
  make a guess;
  while ¬good-enough
    {this is a reasonable guess}
  do
    improve the guess
  end
  {z ≤ x/y < z + t}

```

Introduce a variable d to represent the *known accuracy* of the current guess.

$$\text{INV} \equiv z \leq x/y < z + d$$

Establish the invariant.

```
{0 ≤ x < y ≤ 1}
begin
z := 0;
d := 1;
while d > t
  {z < x/y < z + d}
  do
    improve z and d
  end
{z ≤ x/y < z + t}
```

Let's try a *binary search* (!?) If $z + \frac{1}{2}d > x/y$, then $z \leq x/y < z + \frac{1}{2}d$; otherwise, $z + \frac{1}{2}z \leq \frac{z}{y} < z + d$. Either way, we know the quotient to within $\frac{1}{2}d$, so

```
{0 ≤ x < y ≤ 1}
begin
  z := 0;
  d := 1;
  while d > t do
    {z < x/y < z + d}
    begin
      if z +  $\frac{1}{2}d > x/y$  then z := z else z := z +  $\frac{1}{2}d$ ;
      d :=  $\frac{1}{2}d$ 
    end
  end
end
{z ≤ x/y < z + t}
```

[Add either a **skip** statement or a one-branch **if** to the programming language if the assignment $z := z$ bothers you.] Fix the “cheat” in the **if**-test. Get rid of the division by multiplying through by y .

```
{0 ≤ x < y ≤ 1}
begin
z := 0;
d := 1;
while d > t do
  {z < x/y < z + d}
  begin
    if zy +  $\frac{1}{2}dy$  > x then skip
      else z := z +  $\frac{1}{1}d$ ;
      d :=  $\frac{1}{2}d$ 
    end
  end
end
{z ≤ x/y < z + t}
```

The test seems costly. Introduce “*trailer variables*” u and v to hold get rid of the multiplications, subject to invariants $u = zy$ and $v = \frac{1}{2}dy$. Now we are obligated to maintain these invariants, depending on the test:

Case A: $u + v > x$:

$$d' = \frac{1}{2}d$$

$$z' = z$$

$$u' = z' \cdot y = z \cdot y = u$$

$$v' = \frac{1}{2}(d') \cdot y = \frac{1}{2}(\frac{1}{2}d)y = \frac{1}{2}v$$

Case B: $u + v \leq x$:

$$d' = \frac{1}{2}d$$

$$z' = z + \frac{1}{2}d = z + d'$$

$$u' = z' \cdot y = (z + \frac{1}{2}d)y = zy + \frac{1}{2}dy = u + v$$

$$v' = \frac{1}{2}(d') \cdot y = \frac{1}{2}(\frac{1}{2}d)y = \frac{1}{2}v$$

```

{0 ≤ x < y ≤ 1}
begin
z := 0;
d := 1;
u := 0;
v :=  $\frac{1}{2}y$ ; ] Establish the invariant
while d > t do
    {z < x/y < z + d ∧ u = zy ∧ v =  $\frac{1}{2}dy$ }
    begin
        d :=  $\frac{1}{2}d$ ;
        if u + v > x then skip
            else begin z := z + d; u := u + v; end;
        v :=  $\frac{1}{2}v$ 
    end
end
{z ≤ x/y < z + t}

```

This is called *Wensley's algorithm* [Wensley 58] for real number division, specifically computing the fractional parts of floating point representations.

Since it reduces integer division to addition and divide-by-two, it is a candidate for use in a typical processor, which will have instructions for these operations.

However, this algorithm is not ideally suited for implementation in hardware because addition takes too long ($\mathcal{O}(n)$ for n -bit operands). Later, we will see a series of optimizations that improve on Wensley's algorithm that are used in floating point hardware.