(machine squaring) Given a natural number, write a program to find its square using only
addition, subtraction, doubling, halving, test for even, and test for zero, but not multiplication or division.

§ The question says we can double, but not multiply, so I'll take that to mean that we can multiply by \(2\) but not by anything else. The question says we can halve, but not divide, so I'll take that to mean that we can divide by \(2\) but not by anything else.

For a solution with linear time we could use
\[
a^2 = (a-1)^2 + 2xa - 1
\]

For a solution with logarithmic time, use
\[
\text{if even } a \text{ then } a^2 = 4 \times (a/2)^2 \text{ else } a^2 = 4 \times ((a-1)/2)^2 + 2xa - 1 \text{ fi}
\]

Let all variables be natural.
\[
x := a^2 \iff \text{ if } a=0 \text{ then } x:= 0
\]
\[
ext \text{ if even } a \text{ then } a:= a/2. \ x:= a^2. \ a:= a \times 2. \ x:= x \times 2x2
\]
\[
ext \text{ else } a:= (a-1)/2. \ x:= a^2. \ a:= a \times 2 + 1. \ x:= x \times 2x2 + ax2 - 1 \text{ fi}
\]

Note that in the solution, the occurrences of \(a^2\) are recursive calls. Note also that in the usual binary representation of natural numbers, \(a \times 2\) is just shift left, and both \(a/2\) (for even \(a\)) and \((a-1)/2\) (for odd \(a\)) are just shift right. The refinement can be proven in 3 cases. First case:
\[
a=0 \land (x:= 0) \quad \text{ expand assignment}
\]
\[
\equiv a=0 \land a'=a \land x'=0 \quad \text{ context}
\]
\[
\implies a=0 \land a'=a \land x'=a^2 \quad \text{ specialization}
\]
\[
\implies x:= a^2
\]
Middle case:
\[
a>0 \land \text{ even } a \land (a:= a/2. \ x:= a^2. \ a:= a \times 2. \ x:= x \times 2x2) \quad \text{ expand final assignment}
\]
\[
\equiv a>0 \land \text{ even } a \land (a:= a/2. \ x:= a^2. \ a'=a \land x'=x \times 2x2) \quad \text{ substitution law}
\]
\[
\equiv a>0 \land \text{ even } a \land (a:= a/2. \ a'=ax 2 \land x'=ax 2x2) \quad \text{ substitution law}
\]
\[
\equiv a>0 \land \text{ even } a \land a'=a/2x2 \land x'=(a/2)^2x2 \quad \text{ substitution law}
\]
\[
\equiv a>0 \land \text{ even } a \land a'=a \land x'=a^2 \quad \text{ specialization}
\]
\[
\implies x:= a^2
\]
Last case:
\[
\text{ odd } a \land (a:= (a-1)/2. \ x:= a^2. \ a:= a \times 2 + 1. \ x:= x \times 2x2 + ax2 - 1) \quad \text{ expand final assignment}
\]
\[
\equiv \text{ odd } a \land (a:= (a-1)/2. \ x:= a^2. \ a'=a \land x'=x \times 4 + ax2 - 1) \quad \text{ substitution law}
\]
\[
\equiv \text{ odd } a \land (a:= (a-1)/2. \ x:= a^2. \ a' = ax 2 + 1 \land x' = ax 4 + (ax 2 + 1) \times 2 - 1) \quad \text{ substitution law}
\]
\[
\equiv \text{ even } a \land (a:= (a-1)/2. \ a'= ax 2 + 1 \land x' = (ax 2 + ax 4 + 1) \quad \text{ substitution law}
\]
\[
\equiv \text{ odd } a \land a'=a \land x'=a^2 \quad \text{ specialization}
\]
\[
\implies x:= a^2
\]

For the timing, replace \(x:= a^2\) by \(\text {if } a=0 \text { then } t'=t \text { else } t' \leq t + 1 + \log a \text { fi} \), and put \(t:= t+1\) in front of the recursive calls. The proof is by cases. First,
\[
\text{ if } a=0 \text { then } t'=t \text { else } t' \leq t + 1 + \log a \quad \iff \quad a=0 \land x'=x \land t'=t
\]
\[
\equiv \top
\]
The second case, right side, is
\[
a \neq 0 \land \text{ even } a \land (a:= a/2. \ t:= t+1.
\]
\[
\quad \text{ if } a=0 \text { then } t'=t \text { else } t' \leq t + 1 + \log a \text { fi}.
\]
\[
a:= a \times 2. \ x:= x \times 2x2
\]
a + 0 \land even a \land if a/2 = 0 then t' = t + 1 else t' \leq t + 2 + \log (a/2) fi

\Rightarrow if a = 0 then t' = t else t' \leq t + 1 + \log a fi

which is the left side. The third case, right side, is

a + 0 \land odd a \land (a := (a-1)/2. t := t+1.

\Rightarrow if a = 0 then t' = t else t' \leq t + 1 + \log a fi

which is the left side.

Here's the best solution. Define

P \equiv y' = y + x \times n \land if x = 0 then t' = t else t' \leq t + \log x fi

Then the program is

y' = x^2 \land if x = 0 then t' = t else t' \leq t + \log x fi \iff y := 0. n := x. P

P \iff if even x \land even x \Rightarrow P else odd x \Rightarrow P fi

odd x \Rightarrow P \iff if x = 0 then ok else even x \land x > 0 \Rightarrow P fi

even x \Rightarrow P \iff n := 2 \times n. x := x/2. t := t + 1. x > 0 \Rightarrow P

x > 0 \Rightarrow P \iff if even x \land x > 0 \Rightarrow P else odd x \Rightarrow P fi