Let the variables be \( x \) and \( y \), both natural, and let \( n \) be a natural constant. Then

\[
\begin{align*}
  x' &= n^3 \iff x := n. \ x' &= x \times n. \ x' &= x \times n \\
  x' &= x + n \iff y := x. \ x := 0. \ x' &= x + y \times n \\
  x' &= x + y \times n \iff \text{if } y = 0 \text{ then ok else } x := x + n. \ y := y - 1. \ x' &= x + y \times n \text{ fi}
\end{align*}
\]

Adding recursive time,

\[
\begin{align*}
  x' &= n^3 \land t' = t + n^2 + n \iff x := n. \ x' &= x \times n \land t' = t + x. \ x' &= x \times n \land t' = t + x \\
  x' &= x + n \land t' = t + x \iff y := x. \ x := 0. \ x' &= x + y \times n \land t' = t + y \\
  x' &= x + y \times n \land t' = t + y \iff \text{if } y = 0 \text{ then ok else } x := x + n. \ y := y - 1. \ t := t + 1. \ x' &= x + y \times n \land t' = t + y \text{ fi}
\end{align*}
\]

Here's a linear solution in which \( n \) is a natural variable. We can try to find \( n^3 \) in terms of \((n - 1)^3\). We find

\[
n^3 = (n - 1)^3 + 3 \times n^2 - 3 \times n + 1
\]

The problem is the occurrence of \( n^2 \). But maybe we can find it the same way, in terms of \((n - 1)^2\) using the identity

\[
n^2 = (n - 1)^2 + 2 \times n - 1
\]

So we need a variable \( x \) for the cubes and a variable \( y \) for the squares.

\[
\begin{align*}
  x' &= n^3 \iff x' &= n^3 \land y' &= n^2 \\
  x' &= n^3 \land y' &= n^2 \iff \text{if } n = 0 \text{ then } x := 0. \ y := 0 \text{ else } n := n - 1. \ x' &= n^3 \land y' &= n^2 \\
  \end{align*}
\]

We cannot complete that refinement due to a little problem: in order to get the new values of \( x \) and \( y \), we need not only the values of \( x \) and \( y \) just produced by the recursive call, but also the original value of \( n \), which was not saved. So we revise.

\[
\begin{align*}
  x' &= n^3 \iff x' &= n^3 \land y' &= n^2 \land n' &= n \\
  x' &= n^3 \land y' &= n^2 \land n' &= n \iff \text{if } n = 0 \text{ then } x := 0. \ y := 0 \\
  \text{else } n := n - 1. \ x' &= n^3 \land y' &= n^2 \land n' &= n. \ n := n + 1. \\
  y := y + n + n - 1. \ x := x + y + y + y + n - n - n + 1 \text{ fi}
\end{align*}
\]

After we decrease \( n \), the recursive call promises to leave it alone, and then we increase it back to its original value, which fulfills the promise. With recursive time,

\[
\begin{align*}
  x' &= n^3 \land t' = t + n \iff x' &= n^3 \land y' &= n^2 \land n' &= n \land t' = t + n \\
  x' &= n^3 \land y' &= n^2 \land n' &= n \land t' = t + n \iff \text{if } n = 0 \text{ then } x := 0. \ y := 0 \\
  \text{else } n := n - 1. \ t := t + 1. \ x' &= n^3 \land y' &= n^2 \land n' &= n \land t' = t + n. \ n := n + 1. \\
  y := y + n + n - 1. \ x := x + y + y + y + n - n - n + 1 \text{ fi}
\end{align*}
\]

Now here's a solution using a for-loop according to the invariant rule in Chapter 5. We haven't got many operations to work with. We can try to accumulate a sum, as follows.

\[
x' &= n^3 \iff x := 0. \text{ for } k := 0; \ldots; n \text{ do } x := x + y \text{ od}
\]

We need conditions \( I_k \) such that

\[
\begin{align*}
  I_0 & \iff x := 0 \\
  k : 0; \ldots; n \land I_k & \Rightarrow I_{k + 1} \iff x := x + ? \\
  x' &= n^3 \iff I_n
\end{align*}
\]

From the first and last criteria, it seems clear we need

\[
I_k \equiv x = k^3
\]

Now we can find the question mark.

\[
\begin{align*}
  k : 0; \ldots; n \land I_k & \Rightarrow I_{k + 1} \\
  & \equiv k : 0; \ldots; n \land x = k^3 \Rightarrow x' = (k + 1)^3 \\
  & \equiv k : 0; \ldots; n \land x = k^3 \Rightarrow x' = k^3 + 3 \times k^2 + 3 \times k + 1 \\
  & \iff x := x + 3 \times k + 3 \times k + 1
\end{align*}
\]

Unfortunately, we don't have squaring or multiplication. So let's just say \( x := x + y \) and strengthen \( I \) to
Here's a solution with a less obscure loop specification. Let
\[
I_k \iff x = k^3 \land y = 3xk^2 + 3xk + 1
\]
Now we must revise the initialization
\[
I_0 \iff x := 0. \ y := 1
\]
and recalculate the loop body
\[
k := 0..n \land I_k \Rightarrow I'(k+1)
\]
drop the \( k : 0..n \); it isn't helping
\[
\iff x = k^3 \land y = 3xk^2 + 3xk + 1 \Rightarrow x' = (k+1)^3 \land y' = 3x(k+1)^2 + 3x(k+1) + 1
\]
\[
\iff x' = x + y \land y' = y + 6xk + 6
\]
\[
= \iff x := x + y. \ y := y + k + k + k + k + k + k + 6
\]
and we're done, but it's a little inelegant to add up \( 6k \) so let's say \( y := y + z \) and strengthen \( I \) to
\[
I_k \iff x = k^3 \land y = 3xk^2 + 3xk + 1 \land z = 6xk + 6
\]
Now we must revise the initialization
\[
I_0 \iff x := 0. \ y := 1. \ z := 6
\]
and recalculate the loop body
\[
k := 0..n \land I_k \Rightarrow I'(k+1)
\]
\[
\iff x = k^3 \land y = 3xk^2 + 3xk + 1 \land z = 6xk + 6
\]
\[
\Rightarrow x' = (k+1)^3 \land y' = 3x(k+1)^2 + 3x(k+1) + 1 \land z' = 6x(k+1) + 6
\]
\[
\iff x' = x + y \land y' = y + 6xk + 6 \land z' = z + 6
\]
\[
= \iff x := x + y. \ y := y + z. \ z := z + 6
\]
and we're done again. Altogether,
\[
x' = n^3 \iff x := 0. \ y := 1. \ z := 6. \ \text{for } k := 0..n \ \text{do } x := x + y. \ y := y + z. \ z := z + 6 \ \text{od}
\]
We never need a for-loop, so here's the same solution without one. Let
\[
Q = \forall k : \text{nat} \ x = k^3 \land y = 3xk^2 + 3xk + 1 \land z = 6xk + 6 \Rightarrow x' = (k+n)^3
\]
or, more obscurely and less convenient for proof,
\[
Q = y = 3xk^{2/3} + 3xk^{1/3} + 1 \land z = 6xk^{1/3} + 6 \Rightarrow x' = (x^{1/3} + n)^3
\]
Then
\[
x' = n^3 \land t' = t + n \iff x := 0. \ y := 1. \ z := 6. \ Q \land t' = t + n
\]
\[
Q \land t' = t + n \iff \begin{cases} \text{if } n = 0 \text{ then } \text{ ok} \\ \text{else } x := x + y. \ y := y + z. \ n := n - 1. \ t := t + 1. \ Q \land t' = t + n \end{cases}
\]
Here's a solution with a less obscure loop specification. Let
\[
R = x' = n^3 \land y' = 3nx^2 + 3xn + 1 \land z' = 6xn + 6 \land t' = t + n
\]
Then
\[
x' = n^3 \land t' = t + n \iff R
\]
\[
R \iff \begin{cases} \text{if } n = 0 \text{ then } x := 0. \ y := 1. \ z := 6 \\ \text{else } n := n - 1. \ t := t + 1. \ R. \ x := x + y. \ y := y + z. \ z := z + 6 \end{cases}
\]