0[6] Is \( t' = \infty \) implementable, where \( t \) is time? Prove or disprove.

§ Yes. Suppose \( t \) is the only variable, and its type is \( \text{xnat} \).
\[
\forall t: \text{xnat} . \exists t': \text{xnat} . \ t' = \infty \land t \geq t'
\]
\( = \)
\[
\forall t: \text{xnat} . \ \infty \geq t
\]
\( = \ T
\)

1 Let \( a \), \( b \), and \( c \) be integer variables. Express as simply as possible without using quantifiers, assignments, or dependent compositions.

(a)[6]
\[
a' = a + b + 1. \ b' = a - b - 1
\]
§
\[
a' = a + b + 1. \ b' = a - b - 1
\]
\( = \)
\[
\exists a'', b'', c'': \ a'' = a + b + 1 \land b'' = a'' - b'' - 1 \quad \text{one point for } a'' \land \text{identity for } c''
\]
\( = \)
\[
\exists b'': \ b'' = a + b + 1 - b'' - 1 \quad \text{simplify, rearrange, and identity}
\]
\( = \)
\[
\exists b'': \ b'' = a + b - b' \land T \quad \text{one point for } b''
\]
\( = \ T
\)

(b)[9]
\[
a := a + b + 1. \ b := a - b - 1
\]
§
\[
a := a + b + 1. \ a' = a \land b' = a - b - 1 \land c' = c \quad \text{substitution law}
\]
\( = \)
\[
a' = a + b + 1 \land b' = a + b + 1 - b - 1 \land c' = c \quad \text{simplify}
\]
\( = \)
\[
a' = a + b + 1 \land b' = a \land c' = c
\]

2[9] The problem is to find the remainder after natural division. Let \( n \) be a natural state variable whose initial value is the numerator and whose final value is the remainder. Let \( d: \text{nat+1} \) be the divisor (a constant). Let \( t \) be time. The problem is \( P \) defined as
\[
P \iff \ n' < d \land (\exists m: \text{nat} \cdot n = m \times d + n') \land t' \leq t + n/d
\]
And a refinement is
\[
P \iff \ \text{if } n \times d \ \text{then } o.k \ \text{else } n := n - d. \ t := t + 1. \ P \ \text{fi}
\]
Using proof by parts and cases, what exactly do you have to prove? (You are not being asked to prove; you are being asked what it is that you have to prove if you use proof by parts and cases.)

§
\[
n' < d \iff n < d \land o.k
\]
\( n' < d \iff n \geq d \land (n := n - d. \ t := t + 1. \ n' < d)
\]
\( \exists m: \text{nat} \cdot n = m \times d + n' \iff n < d \land o.k
\]
\( \exists m: \text{nat} \cdot n = m \times d + n' \iff n \geq d \land (n := n - d. \ t := t + 1. \ \exists m: \text{nat} \cdot n = m \times d + n')
\]
\( t' \leq t + n/d \iff n < d \land o.k
\]
\( t' \leq t + n/d \iff n \geq d \land (n := n - d. \ t := t + 1. \ t' \leq t + n/d)
\]

3[12] In a language with array element assignment, what is the exact precondition for \( A'(i') = 1 \) to be refined by \( (A(A(i)) := 0. \ A(i) := 1. \ i' := 2) \)?

§
\[
\forall A', i'. \ A'(i') = 1 \iff (A := A(i) \rightarrow 0 \mid A. \ A := i \rightarrow 1 \mid A. \ i := 2)
\]
\( = \)
\[
\forall A', i'. \ A'(i') = 1 \iff (A := A(i) \rightarrow 0 \mid A. \ A := i \rightarrow 1 \mid A. \ i' := 2 \land A' = A)
\]
\( = \)
\[
\forall A', i'. \ A'(i') = 1 \iff i' = 2 \land A' = i \rightarrow 1 \mid \ A(i) \rightarrow 0 \mid A
\]
\( = \)
\[
(i \rightarrow 1 \mid A(i) \rightarrow 0 \mid A) = 1
\]
\( = \)
\[
i = 2 \lor A(i) \land A = 1
\]