Write a program to find the maximum item in a list.

See book Subsection 8.0.1. See also the solution below, after trying the question.
Let the list be \( L \) (a constant), and I assume its items are numbers. Let \( m \) be a number variable; its final value will be the answer. Let \( i \) be a natural variable to index \( L \). Let \( t \) be time measured recursively. The problem is \( R \), where

\[
R \equiv m' = \uparrow L \land t' = t + \#L
\]

Define \( Q \equiv m' = m \uparrow \uparrow L[i;..\#L] \land t' = t + \#L - i \)

Then

\[
R \iff m := -\infty. \ i := 0. \ Q
\]

Proof:

\[
m := -\infty. \ i := 0. \ Q \quad \text{substitution law, twice}
\]

\[
m' = -\infty \uparrow \uparrow L[0;..\#L] \land t' = t + \#L - 0
\]

\[
Q
\]

Now to refine \( Q \).

\[
Q \iff \text{if } i = \#L \text{ then ok else } m := m \uparrow i. \ i := i + 1. \ t := t + 1. \ Q \text{ fi}
\]

Proof, by cases. First case:

\[
i = \#L \land \text{ok}
\]

expand \( \text{ok} \), and then use context to complicate \( m \) and \( t \)

\[
i = \#L \land m' = m \uparrow \uparrow L[i;..\#L] \land t' = t + \#L - i
\]

specialize

\[
Q
\]

Last case:

\[
i + \#L \land (m := m \uparrow i. \ i := i + 1. \ t := t + 1. \ Q)
\]

substitution, 3 times

\[
i + \#L \land m' = m \uparrow \uparrow L[i;..\#L] \land t' = t + 1 + \#L - (i + 1)
\]

simplify time

\[
i + \#L \land m' = m \uparrow \uparrow L[i;..\#L] \land t' = t + \#L - i
\]

move \( L \) \( i \) inside \( \uparrow \)

\[
Q
\]

For a \textbf{for}-loop solution, define

\[
F \ i \equiv m' = m \uparrow \uparrow L[i;..\#L] \land t' = t + \#L - i
\]

Now we solve the problem as follows:

\[
R \iff m := -\infty. \ F \ 0
\]

Proof:

\[
m := -\infty. \ F \ 0 \quad \text{expand } F, \text{ then substitution law}
\]

\[
m' = -\infty \uparrow \uparrow L[0;..\#L] \land t' = t + \#L - 0
\]

\[
Q
\]

The remaining problem \( F \ 0 \) is the right form to solve with a \textbf{for}-loop.

\[
F \ 0 \iff \text{for } i := 0;..\#L \text{ do } m := m \uparrow \uparrow L[i. \ t := t + 1 \text{ od}
\]

We must prove the two refinements that this abbreviates. First

\[
0 \leq i < \#L \land (m := m \uparrow L[i. \ t := t + 1. \ F(i + 1))\]

expand \( F(j + 1) \)

\[
0 \leq i < \#L \land (m := m \uparrow L[i. \ t := t + 1. \ m' = m \uparrow \uparrow L[i + 1;..\#L] \land t' = t + \#L - (i + 1))
\]

substitution law twice

\[
0 \leq i < \#L \land m' = m \uparrow \uparrow L[i+1;..\#L] \land t' = t + 1 + \#L - (i + 1)
\]

simplify

\[
0 \leq i < \#L \land m' = m \uparrow \uparrow L[i;..\#L] \land t' = t + \#L - i
\]

specialize

\[
Q
\]

Last

\[
F(\#L)
\]

\[
m' = m \uparrow \uparrow L[\#L;..\#L] \land t' = t + \#L - \#L
\]

\[
m' = m \uparrow -\infty \land t' = t
\]

\[
ok
\]

Alternatively, we could have used the invariant form of \textbf{for}-loop law, but without the timing. Define

\[
A \ i \equiv m \uparrow L[0;..i]
\]

Then
\[ R \iff m := -\infty. A 0 \Rightarrow A'(\#L) \]
\[ A 0 \Rightarrow A'(\#L) \iff \text{for } i := 0;..\#L \text{ do } A i \Rightarrow A'(i+1) \text{ od} \]
\[ A i \Rightarrow A'(i+1) \iff m := m \uparrow L i \]

The first and last of these must be proven (the middle one is a gift), and the proofs are a lot like the proofs we have just done.