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

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

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

Proof:
\[
m := -\infty, i := 0, Q \quad \text{substitution law, twice}
\]
\[
= m' = -\infty \uparrow (\uparrow L[0;..#L]) \land t' = t + #L - 0 \quad \text{simplify}
\]
\[
= Q
\]

Now to refine $Q$.
\[
Q \iff \text{if } i = \#L \text{ then } ok \text{ else } m := m(\uparrow L i), i := i + 1, t := t + 1, Q \text{ fi}
\]

Proof, by cases. First case:
\[
i = \#L \land ok \quad \text{expand } ok, \text{ and then use context to complicate } m \text{ and } t
\]
\[
\implies i = \#L \land m' = m(\uparrow L[i;..#L]) \land t' = t + #L - i \quad \text{specialize}
\]

Last case:
\[
i + \#L \land (m := m(\uparrow L i), i := i + 1, t := t + 1, Q)
\]
\[
= i + \#L \land m' = m(\uparrow L i)(\uparrow L[i+1;..#L]) \land t' = t + 1 + #L - (i + 1) \quad \text{simplify time}
\]
\[
= i + \#L \land m' = m(\uparrow L i)(\uparrow L[i+1;..#L]) \land t' = t + #L - i \quad \text{move } L i \text{ inside } \uparrow
\]
\[
= i + \#L \land m' = m(\uparrow L[i;..#L]) \land t' = t + #L - i \quad \text{specialize}
\]
\[
\implies Q
\]

For a for-loop solution, define
\[
F i \equiv m' = m(\uparrow L[i;..#L]) \land t' = t + #L - i
\]

Now we solve the problem as follows:
\[
R \iff m := -\infty, F 0
\]

Proof:
\[
m := -\infty, F 0 \quad \text{expand } F, \text{ then substitution law}
\]
\[
= m' = (-\infty) \uparrow (\uparrow L[0;..#L]) \land t' = t + #L - 0 \quad \text{simplify}
\]
\[
= R
\]

The remaining problem $F 0$ is the right form to solve with a for-loop.
\[
F 0 \iff \text{for } i := 0;..#L \text{ do } m := m(\uparrow L i), t := t + 1 \text{ od}
\]

We must prove the two refinements that this abbreviates. First
\[
0 \leq i < \#L \land (m := m(\uparrow L i), t := t + 1, F(i + 1)) \quad \text{expand } F(j + 1)
\]
\[
= 0 \leq i < \#L \land (m := m(\uparrow L i), t := t + 1, m' = m(\uparrow L[i+1;..#L]) \land t' = t + #L - (i + 1)) \quad \text{substitution law twice}
\]
\[
= 0 \leq i < \#L \land m' = m(\uparrow L(i+1;..#L)) \land t' = t + 1 + #L - (i + 1) \quad \text{simplify twice}
\]
\[
= 0 \leq i < \#L \land m' = m(\uparrow L[i;..#L]) \land t' = t + #L - i \quad \text{specialize}
\]
\[
\implies F i
\]

Last
\[
F(\#L) \equiv m' = m(\uparrow L[#L;..#L]) \land t' = t + #L - #L
\]
\[
= m' = m(-\infty) \land t' = t \quad \text{ok}
\]

Alternatively, we could have used the invariant form of for-loop law, but without the timing. Define
\[
A i \equiv m = \uparrow L[0;..i]
\]
Then

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

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.

§ also see book Subsection 8.0.1