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

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

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

Then

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

Proof:

\[
m := -\infty . \ i := 0 . \ Q
= m' = \max -\infty (\max L[0;..\#L]) \land t' = t + \#L - 0
= Q
\]

Now to refine \( Q \).

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

Proof, by cases. First case:

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

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

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

specialize

\[
Q
\]

Last case:

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

substitution, 3 times

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

\[
\text{max} \text{ is associative; and simplify time max}
\]

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

move \( L_i \) inside \( \max \)

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

specialize

\[
Q
\]

For a for-loop solution, define

\[
F i k \equiv m' = \max m (\max L[i;..k]) \land t' = t + k - i
\]

Now we solve the problem as follows:

\[
R \iff m := -\infty . \ F 0 (\#L)
\]

Proof:

\[
m := -\infty . \ F 0 (\#L)
= m' = \max (\max L[0;..\#L]) \land t' = t + \#L - 0
= R
\]

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

\[
F 0 (\#L) \iff \text{for } j := 0;..\#L \text{ do } m := \max m (L_j) . \ t := t + 1 \text{ od}
\]

We must prove the three refinements that this abbreviates. First

\[
F j j \iff 0 \leq j \leq \#L \land \text{ok}
\]

expand \( F \) and \( \text{ok} \)

\[
m' = \max m (\max L[j;..j]) \land t' = t + j - j
\]

\[
\text{MAX} \text{ of an empty segment}
\]

\[
m' = \max m (\max L[0;..\#L]) \land t' = t
\]

\[
m' = m \land t' = t
\]

\[
\text{specialization}
\]

\[
T
\]

Next

\[
F j (j+1) \iff 0 \leq j \leq \#L \land (m := \max m (L_j) . \ t := t + 1)
\]

expand \( F \) and the last assignment and use substitution law

\[
m' = \max m (\max L[j;..j+1]) \land t' = t + j + 1 - j
\]

\[
\text{one-item segment}
\]

\[
m' = \max m (L_j) \land t' = t + 1
\]

\[
\text{specialization}
\]

\[
T
\]
Last, starting with its right side,

\[ 0 \leq i < j < k \leq L \land (F i j \cdot F j k) \quad \text{expand last two } F \text{'s} \]

\[ = \quad 0 \leq i < j < k \leq L \]
\[ \land (m' = \max m (MAX L[i..j]) \land t' = t + j - i) \]
\[ \land (m'' = \max m (MAX L[j..k]) \land t'' = t + k - j) \quad \text{expand dependent composition} \]
\[ = \quad 0 \leq i < j < k \leq L \]
\[ \land (\exists m'', t''. \quad m'' = \max m (MAX L[i..j]) \land t'' = t + j - i) \]
\[ \land m' = \max m'' (MAX L[i..k]) \land t' = t'' + k - j) \quad \text{one-point} \]
\[ = \quad 0 \leq i < j < k \leq L \land m' = \max m (MAX L[i..j]) (MAX L[j..k]) \land t' = t + k - i \quad \text{max is associative} \]
\[ = \quad 0 \leq i < j < k \leq L \land m' = \max m (MAX L[i..j]) (MAX L[j..k]) \land t' = t + k - i \quad \text{MAX law} \]
\[ \Rightarrow \quad F i k \quad \text{specialization} \]

Alternatively, we could have used the invariant form of for-loop law. Define

\[ I_j = m = MAX L[0..j] \]

Then

\[ R \longleftarrow m := -\infty. \quad I_0 \Rightarrow I' (#L) \]
\[ I_0 \Rightarrow I' (#L) \longleftarrow \quad \text{for } j := 0..L \text{ do } \quad I_j \Rightarrow I' (j+1) \text{ od} \]
\[ I_j \Rightarrow I' (j+1) \longleftarrow \quad m := \max m (L, j) \]

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. In either case, the recursive timing is \( t' = t + #L \).