(longest sorted sublist) Write a program to find the length of a longest sorted sublist of a given list, where

(a) the sublist must be consecutive items (a segment).

§ Let $L$ be the list, and let $j: \text{nat}$ be an index variable, and let $m: \text{nat}$ be a variable recording the length of the longest sorted segment ending at index $j$, and let $n: \text{nat}$ be a variable recording the length of a longest sorted segment anywhere in the interval 0..$j$. The result is the final value of $n$. Define $\text{sorted}$ so that $\text{sorted} \ h \ k$ tells whether the segment $h..k$ is sorted,

$$\text{sorted} = \langle h, k: \text{nat} \rightarrow \forall i, j: h..k. \ i \leq j \Rightarrow L i \leq L j \rangle$$

and define $\text{llss}$ so that $\text{llss} \ j$ is the length of a longest sorted segment ending at index $j$

$$\text{llss} = \langle j: 0..\#L+1 \rightarrow \uparrow i: 0..j+1. \text{if sorted} \ i \ j \ \text{then} \ j-i \ \text{else} \ \infty \rangle$$

Then the problem is $P$, defined as

$$P = n' = \uparrow i: 0..\#L+1. \text{llss} \ i$$

And one last definition:

$$Q = 0 < j < \#L \land m = \text{llss} \ j \land n = (\uparrow i: 0..j+1. \text{llss} \ i) \Rightarrow P$$

which says that if $j$ is an index (but not 0), and $m$ is the length of a longest sorted segment ending at index $j$, and $n$ is the length of a longest sorted segment ending at or before index $j$, then $n'$ is the length of a longest sorted segment. Now the solution is

$$P \iff \text{if } \#L \leq 1 \text{ then } n' = \#L \text{ else } j:= 1. \ m:= 1. \ n:= 1. \ Q \ \text{fi}$$

$$Q \iff \text{if } L(j-1) \leq L \ j \text{ then } j:= j+1. \ m:= m+1. \ n:= n \uparrow m. \text{ if } j=\#L \text{ then ok else } Q \ \text{fi}$$

$$\text{else } j:= j+1. \ m:= 1. \text{ if } \#L - j < n \text{ then ok else } Q \ \text{fi} \ \text{fi}$$

The timing is $t' \leq t + \#L$ and $t' \leq t + \#L - j$.

(b) the sublist consists of items in their order of appearance in the given list, but not necessarily consecutively.

§ This time let $\text{llss} \ n$ be the length of a longest sorted sublist of list $L[0..n]$. We can represent a sublist of $L$ by a set $S$ of indexes, which is a subset of \{0..n\}. Formally,

$$\text{llss} \ n = \uparrow S: (\langle S: \langle 0..n \rangle. \forall i, j: \sim S. i \leq j \Rightarrow L i \leq L j \rangle. \$S)$$

And this time I'll use a for-loop. Define invariant

$$A \ n \equiv s = \text{llss} \ n$$

Then

$$s' = \text{llss} \ (\#L) \iff s:= 0. \ A \ 0 \Rightarrow A'(\#L)$$

$$A \ 0 \Rightarrow A'(\#L) \iff \text{for } n:= 0..\#L \text{ do } n: 0..\#L \land A \ n \Rightarrow A'(n+1) \text{ od}$$

The first refinement is easy to prove, the second doesn't need proof, and we have yet to refine $n: 0..\#L \land A \ n \Rightarrow A'(n+1)$. As we go from $n$ to $n + 1$, the new sublists are $L[n]$ whose length is 1, and for each sorted sublist $S$ in $L[0..n]$ whose last item is less than or equal to $L n$, the list $S[0..L n]$ whose length is $\#S + 1$. To calculate that, we will form a new list $M$ such that $M k$ is the length of the longest sorted sublist whose last item is $L k$. We strengthen $A$.

$$A \ n \equiv s = \text{llss} \ n$$

$$\land \forall k:0..n. \ M k = \uparrow S: (\langle S: \langle 0..k+1 \rangle. \ k \in S \land \forall i, j: \sim S. i \leq j \Rightarrow L i \leq L j \rangle. \$S)$$

Note that $s = \uparrow \langle M[0..n]\rangle$ except when $n=0$. The remaining refinement will also use a for-loop, for which we define invariant

$$B \ m \equiv (\forall k:0..n. \ M k = \uparrow S: (\langle S: \langle 0..k+1 \rangle. \ k \in S \land \forall i, j: \sim S. i \leq j \Rightarrow L i \leq L j \rangle. \$S) \land M n = 1 \uparrow \langle k: \langle 0..m. \ L k \leq L n \rangle. \ M k + 1 \rangle)$$

Now

$$(B \ 0 \Rightarrow B' n) \land s' = s \iff \text{for } m:= 0..n \text{ do } (B \ m \Rightarrow B'(m+1)) \land s' = s \text{ od}$$
If you object that the for-loop specification \((B \ 0 \Rightarrow B' n) \land s' = s\) is not exactly in the right form, I could use frame to put it in the right form, or use the more general for-loop rule. The solution just given has running time \((#L)^2/2\). For a solution with running time bounded by \((#L) \times \log (#L)\), instead of maintaining the list \(M\) of lengths of longest sorted sublists, maintain the list of minimum last items for each length, and replace the inner loop with a binary search.