229 (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).

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

After trying the question, scroll down to the solution.

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

§ Let L be the list, and let j: *nat* be an index variable, and let m: *nat* be a variable recording the length of the longest sorted segment ending at index j, and let n: *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 *sorted* so that *sorted h k* tells whether the segment $h;..k$ is sorted,

$$sorted \;=\; \langle h, k\text{: } nat\cdot\ \forall i, j\text{: } h,..k\cdot\ i{\le}j \Rightarrow Li{\le}Lj\rangle$$

and define *llss* so that *llss j* is the length of a longest sorted segment ending at index j

$$llss \;=\; \langle j\text{: } 0,..\#L{+}1\cdot\ {\Uparrow}i\text{: } 0,..j{+}1\cdot\ \textbf{if } sorted\ i\ j\ \textbf{then } j{-}i\ \textbf{else } {-}\infty\ \textbf{fi}\rangle$$

Then the problem is P, defined as

$$P \;=\; n' = {\Uparrow}i\text{: } 0,..\#L{+}1\cdot\ llss\ i$$

And one last definition:

$$Q \;=\; 0{<}j{<}\#L \;\wedge\; m = llss\ j \;\wedge\; n = ({\Uparrow}i\text{: } 0,..j{+}1\cdot\ llss\ i) \;\Rightarrow\; P$$

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

```
$P \Leftarrow$   if $\#L{\le}1$ then $n{:=}\ \#L$ else $j{:=}\ 1.\ \ m{:=}\ 1.\ \ n{:=}\ 1.\ \ Q$ fi
$Q \Leftarrow$   if $L(j{-}1) \le L\,j$
           then $j{:=}\ j{+}1.\ \ m{:=}\ m{+}1.\ \ n{:=}\ n{\uparrow}m.$ if $j{=}\#L$ then $ok$ else $Q$ fi
           else $j{:=}\ j{+}1.\ \ m{:=}\ 1.$ if $\#L{-}j{<}n$ then $ok$ else $Q$ fi fi
```

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

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

§ This time let $llss\ n$ 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,

$$llss\ n \;=\; {\Uparrow}S\text{: } (\S S\text{: } {\not\!\!\,\$}(0,..n)\cdot\ \forall i, j\text{: } {\sim}S\cdot\ i{\le}j \Rightarrow L\,i \le L\,j)\cdot\ \$S$$

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

$$A\,n \;=\; s = llss\ n$$

Then

$$s' = llss\ (\#L) \;\Leftarrow\; s{:=}\ 0.\ \ A\,0 \Rightarrow A'(\#L)$$
$$A\,0 \Rightarrow A'(\#L) \;\Leftarrow\; \textbf{for } n{:=}\ 0;..\#L\ \textbf{do } n\text{: } 0,..\#L \wedge A\,n \;\Rightarrow\; A'(n{+}1)\ \textbf{od}$$

The first refinement is easy to prove, the second doesn't need proof, and we have yet to refine $n\text{: } 0,..\#L \wedge A\,n \;\Rightarrow\; A'(n{+}1)$. 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 $S;;[L\,n]$ 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.

$$\begin{aligned} & A\,n \\ = \quad & s = llss\ n \\ & \wedge\; \forall k{:}0,..n\cdot\ M\,k = {\Uparrow}S\text{: } (\S S\text{: } {\not\!\!\,\$}(0,..k{+}1)\cdot\ k{\in}S \wedge \forall i, j\text{: } {\sim}S\cdot\ i{\le}j \Rightarrow L\,i \le L\,j)\cdot\ \$S \end{aligned}$$

Note that $s = {\Uparrow}(M[0;..n])$ except when $n{=}0$. The remaining refinement will also use a **for**-loop, for which we define invariant

$$\begin{aligned} & B\,m \\ = \quad & (\forall k{:}0,..n\cdot\ M\,k = {\Uparrow}S\text{: } (\S S\text{: } {\not\!\!\,\$}(0,..k{+}1)\cdot\ k{\in}S \wedge \forall i, j\text{: } {\sim}S\cdot\ i{\le}j \Rightarrow L\,i \le L\,j)\cdot\ \$S) \\ & \wedge\; M\,n = 1{\uparrow}({\Uparrow}k\text{: } (\S k\text{: } 0,..m\cdot\ L\,k \le L\,n)\cdot\ M\,k + 1) \end{aligned}$$

Now

$$(B\,0 \Rightarrow B'n) \wedge s'{=}s \;\Leftarrow\; \textbf{for } m{:=}\ 0;..n\ \textbf{do } (B\,m \Rightarrow B'(m{+}1)) \wedge s'{=}s\ \textbf{od}$$
$$B\,n \Rightarrow B'(n{+}1) \;\Leftarrow\; M\,n{:=}\ 1.\ \ (B\,0 \Rightarrow B'n) \wedge s'{=}s.\ \ s{:=}\ s{\uparrow}(M\,n)$$
$$(B\,m \Rightarrow B'(m{+}1)) \wedge s'{=}s \;\Leftarrow\; \textbf{if } L\,m \le L\,n\ \textbf{then } M\,n{:=}\ (M\,n){\uparrow}(M\,m{+}1)\ \textbf{else } ok\ \textbf{fi}$$

If you object that the **for**-loop specification $(B\,0 \Rightarrow B'n) \wedge s'{=}s$ 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.