208 (n sort) Given a list L such that $L(\square L) = \square L$, write a program to sort L in linear time and constant space. The only change permitted to L is to swap two items.

After trying the question, scroll down to the solution.

§ The problem is P , defined as

$$P \;=\; L(\square L) = \square L \;\Rightarrow\; L' = [0;..\#L]$$

The only change permitted to L is $swap$, defined as

$$swap\ i\ j \;=\; L{:=}\ i \rightarrow L\,j \mid j \rightarrow L\,i \mid L$$

Execution time has to be linear, so that suggests starting an index variable k at 0 , and moving up by $k{:=}\ k{+}1$ until $k{=}\#L$, so that the part of the list before k is in order, and therefore the part of the list from k onward has the right items but maybe not yet in the right order.

$$P \Leftarrow k{:=}\ 0.\ Q$$
$$Q \Leftarrow \textbf{if } k{=}\#L \textbf{ then } ok$$
$$\textbf{else if } L\,k = k \textbf{ then } k{:=}\ k{+}1.\ Q$$
$$\textbf{else } swap\ (L\,k)\ k.\ Q \textbf{ fi fi}$$

To define Q , we can look at P for inspiration. Perhaps

$$Q \;=\; L(k,..\#L) = k,..\#L \;\Rightarrow\; L' = L[0;..k] \;;;\; [k;..\#L]$$

I think that will work. But I think it will be easier to prove the Q refinement if we weaken Q by strengthening its antecedent. I'm going to try

$$Q \;=\; L[0;..k] = [0;..k] \;\wedge\; L(k,..\#L) = k,..\#L \;\Rightarrow\; L' = L[0;..k] \;;;\; [k;..\#L]$$

This says: if the first part of L is done, and the last part has the right items (but not necessarily in the right order), then we complete the job by leaving the first part of L alone and putting the last part in order.

Proof of P refinement:

$k{:=}\ 0.\ Q$ — replace Q

$=\ k{:=}\ 0.\ L[0;..k] = [0;..k] \;\wedge\; L(k,..\#L) = k,..\#L \;\Rightarrow\; L' = L[0;..k] \;;;\; [k;..\#L]$ — Substitution Law

$=\ L[0;..0] = [0;..0] \;\wedge\; L(\square L) = \square L \;\Rightarrow\; L' = L[0;..0] \;;;\; [0;..\#L]$ — simplify

$=\ P$

Proof of first case of Q refinement:

$k{=}\#L \wedge ok \Rightarrow Q$ — replace ok and Q

$=\ \ k{=}\#L \wedge k'{=}k \wedge L'{=}L$
$\Rightarrow (L[0;..k] = [0;..k] \;\wedge\; L(k,..\#L) = k,..\#L \;\Rightarrow\; L' = L[0;..k] \;;;\; [k;..\#L])$ — context

$=\ \ k{=}\#L \wedge k'{=}k \wedge L'{=}L$
$\Rightarrow (L[0;..\#L] = [0;..\#L] \;\wedge\; L(\#L,..\#L) = \#L,..\#L \;\Rightarrow\; L = L[0;..\#L] \;;;\; [\#L;..\#L])$ — simplify

$=\ \top$

Proof of middle case of Q refinement:

$k{\neq}\#L \wedge Lk{=}k \wedge (k{:=}\ k{+}1.\ Q)$ — replace Q

$=\ \ k{\neq}\#L \wedge Lk{=}k$
$\wedge (k{:=}\ k{+}1.\ L[0;..k] = [0;..k] \;\wedge\; L(k,..\#L) = k,..\#L \;\Rightarrow\; L' = L[0;..k] \;;;\; [k;..\#L])$ — substitution law

$=\ \ k{\neq}\#L \;\wedge\; L\,k = k$
$\wedge (L[0;..k{+}1] = [0;..k{+}1] \wedge L(k{+}1,..\#L) = k{+}1,..\#L \;\Rightarrow\; L' = L[0;..k{+}1] \;;;\; [k{+}1;..\#L])$ — use context $Lk{=}k$ to simplify the implication

$=\ k{\neq}\#L \wedge L\,k = k \wedge (L[0;..k] = [0;..k] \;\wedge\; L(k,..\#L) = k,..\#L \;\Rightarrow\; L' = L[0;..k];;[k;..\#L])$

$=\ k{\neq}\#L \wedge Lk{=}k \wedge Q$ — specialize

$\Rightarrow \quad Q$

Proof of last case of Q refinement:

$k \ne \#L \ \land \ L\,k \ne k \ \land \ (swap\ (Lk)\ k.\ \ Q) \ \Rightarrow \ Q$ replace last Q

$= \quad k \ne \#L \ \land \ L\,k \ne k \ \land \ (swap\ (Lk)\ k.\ \ Q)$

$\Rightarrow \ (L[0;..k] = [0;..k] \ \land \ L(k,..\#L) = k,..\#L \ \ \Rightarrow \ \ L' = L[0;..k] \ ;;\ [k;..\#L])$ portation

$= \quad k \ne \#L \land L\,k \ne k \land (swap\ (Lk)\ k.\ \ Q) \land \ L[0;..k] = [0;..k] \ \land \ L(k,..\#L) = k,..\#L$

$\Rightarrow \ \ L' = L[0;..k] \ ;;\ [k;..\#L]$

To prove this implication, I'll go from the antecedent on the top line to the consequent on the bottom line.

$k \ne \#L \land L\,k \ne k \land (swap\ (Lk)\ k.\ \ Q) \land \ L[0;..k] = [0;..k] \ \land \ L(k,..\#L) = k,..\#L$

replace $swap$ and Q

$= \quad k \ne \#L \ \land \ L\,k \ne k \land \ L[0;..k] = [0;..k] \ \land \ L(k,..\#L) = k,..\#L$

$\land \ (\quad L{:=}\ Lk{\to}Lk \mid k{\to}L(Lk) \mid L.$

$L[0;..k] = [0;..k] \ \land L(k,..\#L) = k,..\#L \ \ \Rightarrow \ \ L' = L[0;..k] \ ;;\ [k;..\#L] \ \ )$

substitution law

$= \quad k \ne \#L \ \land \ L\,k \ne k \land \ L[0;..k] = [0;..k] \ \land \ L(k,..\#L) = k,..\#L$

$\land \ (\qquad (L\,k \to L\,k \mid k \to L(L\,k) \mid L)[0;..k] = [0;..k]$

$\land \quad (L\,k \to L\,k \mid k \to L(L\,k) \mid L)(k,..\#(L\,k \to L\,k \mid k \to L(L\,k) \mid L))$

$= \ k,..\#(L\,k \to L\,k \mid k \to L(L\,k) \mid L)$

$\Rightarrow \ \ L' = (L\,k \to L\,k \mid k \to L(L\,k))[0;..k] \ ;;\ [k;..\#(L\,k \to Lk \mid k \to L(L\,k) \mid L)] \quad)$

swap does not affect length

$= \quad k \ne \#L \ \land \ L\,k \ne k \land \ L[0;..k] = [0;..k] \ \land \ L(k,..\#L) = k,..\#L$

$\land \ (\qquad (L\,k \to L\,k \mid k \to L(L\,k) \mid L)[0;..k] = [0;..k]$

$\land \ \ (L\,k \to L\,k \mid k{\to}L(L\,k) \mid L)(k,..\#L) = k,..\#L$

$\Rightarrow \ \ L' = (L\,k \to L\,k \mid k \to L(L\,k))[0;..k] \ ;;\ [k;..\#L] \quad)$

This next step is more complicated and less formal than I would like. In the top line it says $L[0;..k] = [0;..k]$, and since each item in the list occurs once, the items less than k are used up at indexes less than k . The top line also says $Lk \ne k$, therefore $L\,k > k$. So the swap is swapping the item at k with an item at an index greater than k . The swap does not affect the first part of the list $L[0;..k]$. The swap affects the last part of the list, but it does not change the bunch of items in the last part of the list $L(0,..k)$. So the top line, used as context, allows us to simplify the bottom three lines.

$= \quad k \ne \#L \ \land \ L\,k \ne k \land \ L[0;..k] = [0;..k] \ \land \ L(k,..\#L) = k,..\#L$

$\land \ (\qquad L[0;..k] = [0;..k]$

$\land \ \ L(k,..\#L) = k,..\#L$

$\Rightarrow \ \ L' = L[0;..k] \ ;;\ [k;..\#L] \quad)$ discharge

$= \quad k \ne \#L \ \land \ L\,k \ne k \land \ L[0;..k] = [0;..k] \ \land \ L(k,..\#L) = k,..\#L$

$\land \ L' = L[0;..k] \ ;;\ [k;..\#L]$ specialize

$\Rightarrow \quad L' = L[0;..k] \ ;;\ [k;..\#L]$

And that completes the last case of the Q refinement.

Recursive time is bounded by $2{\times}\#L$. Counting just $swap$s, the time is bounded by $\#L$. To prove time bounds, it is helpful to define

$f\,i \ = \ ¢§j{:}\ i,..\#L\cdot\ L\,j \ne j$

Then the timing specifications are A and B , defined as

$A \ = \ t' \le t + \#L + f\,0$

$B \ = \ t' \le t + \#L - k + f\,k$

With time, the refinements are

$$A \Leftarrow k:=0.\ B$$
$$B \Leftarrow \textbf{if } k=\#L \textbf{ then } ok$$
$$\textbf{else if } Lk=k \textbf{ then } k:=k+1.\ t:=t+1.\ B$$
$$\textbf{else } swap\ (Lk)\ k.\ t:=t+1.\ B \textbf{ fi fi}$$

Proof of A refinement:

	$k:=0.\ B$	replace B
$=$	$k:=0.\ t' \le t + \#L - k + f\,k$	Substitution Law
$=$	$t' \le t + \#L - 0 + f\,0$	
$=$	A	

Proof of first case of B refinement:

	$k=\#L \wedge ok \Rightarrow B$	replace ok and B
$=$	$k=\#L \wedge k'=k \wedge L'=L \wedge t'=t \Rightarrow t' \le t + \#L - k + f\,k$	context
$=$	$k=\#L \wedge k'=k \wedge L'=L \wedge t'=t \Rightarrow t \le t + \#L - \#L + f\,(\#L)$	simplify and apply
$=$	$k=\#L \wedge k'=k \wedge L'=L \wedge t'=t \Rightarrow 0 \le ¢\S j\text{: } \#L,..\#L\cdot\ L\,j \ne j$	simplify
$=$	$k=\#L \wedge k'=k \wedge L'=L \wedge t'=t \Rightarrow 0 \le 0$	simplify and base
$=$	$\top$	

Proof of middle case of B refinement:

	$k \ne \#L \wedge L\,k = k \wedge (k:=k+1.\ t:=t+1.\ B)$	replace B
$=$	$k \ne \#L \wedge L\,k = k \wedge (k:=k+1.\ t:=t+1.\ t' \le t + \#L - k + f\,k)$	substitution law
$=$	$k \ne \#L \wedge L\,k = k \wedge t' \le t + 1 + \#L - k - 1 + f\,(k+1)$	simplify
$=$	$k \ne \#L \wedge L\,k = k \wedge t' \le t + \#L - k + f\,(k+1)$	context $L\,k = k$ implies $f\,k = f(k+1)$
$=$	$k \ne \#L \wedge L\,k = k \wedge t' \le t + \#L - k + f\,k$	specialize
$\Rightarrow$	B	

Proof of last case of B refinement:

$$k \ne \#L \wedge L\,k \ne k \wedge (swap\ (L\,k)\ k.\ t:=t+1.\ B) \qquad \text{replace } swap \text{ and } B$$

$$= \quad k \ne \#L \wedge L\,k \ne k \wedge (L:= L\,k \rightarrow L\,k \mid k \rightarrow L(L\,k) \mid L.\ t:=t+1.\ t' \le t + \#L - k + f\,k)$$

The next step looks like it should be the Substitution Law. But f is defined in terms of L. So we have to apply f first.

$$= \quad k \ne \#L \wedge Lk \ne k \wedge (\ L:= L\,k \rightarrow L\,k \mid k \rightarrow L(L\,k) \mid L.\ t:=t+1.\ t' \le t + \#L - k + ¢\S j\text{: } k,..\#L\cdot\ L\,j \ne j\ )$$

Now use the Substitution Law

$$= \quad k \ne \#L \wedge L\,k \ne k \wedge t' \le t + 1 + \#(L\,k \rightarrow L\,k \mid k \rightarrow L(L\,k) \mid L) - k + ¢\S j\text{: } k,..\#(L\,k \rightarrow L\,k \mid k \rightarrow L(L\,k) \mid L)\cdot\ (L\,k \rightarrow L\,k \mid k \rightarrow L(L\,k) \mid L)j \ne j$$

swap does not affect length

$$= \quad k \ne \#L \wedge L\,k \ne k \wedge t' \le t + 1 + \#L - k + ¢\S j\text{: } k,..\#L\cdot\ (L\,k \rightarrow L\,k \mid k \rightarrow L(L\,k) \mid L)j \ne j$$

swap reduces the number of out-of-place items by 1 or 2

$$\Rightarrow \quad k \ne \#L \wedge L\,k \ne k \wedge t' \le t + 1 + \#L - k + ¢(\S j\text{: } k,..\#L\cdot\ L\,j \ne j) - 1$$

$$= \quad k \ne \#L \wedge L\,k \ne k \wedge t' \le t + \#L - k + f\,k \qquad \text{specialize}$$

$$\Rightarrow \quad B$$

And that completes the last case of the B refinement.