Let $L$ be a list variable assigned a non-empty list. Reassign it so that any run of two or more identical items is collapsed to a single item.

Let $i$ be a natural variable used to index $L$. The program is

\[
P \iff i := 1. \quad Q
\]

\[
Q \iff \text{ if } i = \#L \text{ then } ok
\]

\[
\text{ else if } L[i] = L[i-1] \text{ then } L := ((0;..i) ; (i+1;..\#L))
\]

\[
\text{ else } i := i + 1. \quad Q \quad \text{fi}
\]

Now we need to define specifications $P$ and $Q$, and then prove the two refinements.

Part of the specification says that $L$ and $L'$ have the same items in them.

$L(0,..\#L) = L'(0,..\#L')$

Another part of the specification is that in $L'$ no two adjacent items are equal.

$\neg \exists j: 1,..\#L'. \; L[j] = L'[j-1]$

The rest of the specification is that the items of $L'$ are in the same order as in $L$. I don't know how to formalize that. So I'll prove what I can.

Let's start with

\[
P = Q = L(0,..\#L) = L'(0,..\#L')
\]

Here's the proof of the first refinement, starting with the right side.

\[
i := 1. \quad Q
\]

\[
= i := 1. \quad L(0,..\#L) = L'(0,..\#L')
\]

\[
= L(0,..\#L) = L'(0,..\#L')
\]

\[
= P
\]

Now the last refinement, by cases. First case:

\[
i = \#L \wedge ok
\]

\[
= i = \#L \wedge L' = L \wedge i' = i
\]

\[
\Rightarrow Q
\]

Middle case:

\[
i \neq \#L \wedge L[i] = L[i-1] \wedge (L := ((0;..i) ; (i+1;..\#L))). \quad Q
\]

\[
= \text{UNFINISHED}
\]

\[
\Rightarrow Q
\]

Last case:

\[
i \neq \#L \wedge L[i] \neq L[i-1] \wedge (i := i+1). \quad Q
\]

\[
= i \neq \#L \wedge L[i] \neq L[i-1] \wedge L(0,..\#L) = L'(0,..\#L')
\]

\[
= \text{substitution law}
\]

\[
i \neq \#L \wedge L[i] \neq L[i-1] \wedge L(0,..\#L) = L'(0,..\#L')
\]

\[
\Rightarrow Q
\]

Now redefine

\[
P = \neg \exists j: 1,..\#L'. \; L'[j] = L'[j-1]
\]

\[
Q = \neg \exists j: i,..\#L'. \; L'[j] = L'[j-1]
\]

Here's the proof of the first refinement, starting with the right side.

\[
i := 1. \quad Q
\]

\[
= i := 1. \quad \neg \exists j: i,..\#L'. \; L'[j] = L'[j-1]
\]

\[
= \neg \exists j: 1,..\#L'. \; L'[j] = L'[j-1]
\]

\[
= P
\]

Now the last refinement, by cases. First case:

\[
i = \#L \wedge ok
\]

\[
= i = \#L \wedge L' = L \wedge i' = i
\]

\[
\Rightarrow Q
\]

Middle case:

\[
i \neq \#L \wedge L[i] = L[i-1] \wedge (L := ((0;..i) ; (i+1;..\#L))). \quad Q
\]
Last case:
\[ i \neq \#L \land L \cdot i \neq L(i-1) \land (i := i + 1). \ Q \]

The recursive time is \( \#L - 1 \). Redefine

\[ P \leftarrow t' = t + \#L - 1 \]
\[ Q \leftarrow t' = t + \#L - i \]

and insert the time increment

\[ P \leftarrow i := 1. \ Q \]
\[ Q \leftarrow \text{if } i = \#L \text{ then } \text{ok} \]
\[ \text{else if } L \cdot i = L(i-1) \text{ then } L := L((0;..i) ; (i+1;..\#L)) \]
\[ \text{else } i := i + 1. \ t := t + 1. \ Q \text{ fi} \]