Introduce variable $n : \text{nat}$, and a stack. Define predicate $P : \text{text} \rightarrow \text{bin}$ so that $P s$ means that $s$ has its brackets properly paired and nested. Here are its axioms.

$P ""$
$P "x"
$P a = P ("( ; a ; ")")$
$P a = P ("[ ; a ; "]")$
$P a \land P z \Rightarrow P (a ; z)$

Now we need two specifications.

$R = (\text{isempty} \land n' \iff t \equiv P t)$
$Q = \text{(defined later)}$

Here are the refinements.

$R \iff \text{mkempty. } n := 0. \ Q$

$Q \iff \if n \equiv t \then \text{ok}
\else \if t_n = "x" \then n := n + 1. \ Q
\else \if t_n = "( \then \text{push } "\). \ n := n + 1. \ Q
\else \if t_n = "[ \then \text{push } "]. \ n := n + 1. \ Q
\else \if \text{isempty} \then \text{ok}
\else \if t_n = \text{top} \then \text{pop. } n := n + 1. \ Q
\else \text{ok} \fi \fi \fi \fi \fi \fi

I have used a stack, and for the purpose of executing the program, the stack can be implemented any way that is correct. But for the purpose of defining specification $Q$, I implement it as follows. Let $s$ be a text-valued implementer's variable.

$\text{mkempty} = s := \text{nil}$
$\text{isempty} = \iff s = 0$
$\text{push} = \langle c : \text{char} \mid s := c ; s \rangle$
$\text{top} = s_0$
$\text{pop} = s := s_1 \cdots \leftarrow s$

Now I can define specification $Q$.

$Q \equiv P (t_{0..n} ; s) \Rightarrow R$

And finally we can prove the refinements. UNFINISHED