Given a natural number and a list of natural numbers, write a program to determine if every natural number up to the given number is an item in the list.

Let the natural number be \( n \), and let the list be \( L \). We will look through \( L \) from beginning to end, remembering which numbers in \( 0, \ldots, n \) have been seen. Let \( B: [n^*\text{bin}] \) be a list variable to say what has been found. To start, \( B = [n^*\bot] \) saying that no numbers in \( 0, \ldots, n \) have been found yet. In the end, if \( B' = [n^*\top] \) then all \( n \) numbers are present in list \( L \). The specification is \( S \), defined as

\[
S = \forall i: 0, \ldots, n \cdot B'i = \exists j: 0, \ldots, n\cdot L j = i
\]

Introduce natural variable \( k \) and define

\[
R = 0 \leq k \leq \#L \land (\forall i: 0, \ldots, n \cdot B i = \exists j: 0, \ldots, k \cdot L j = i) \implies S
\]

Now refine

\[
S \iff B: [n^*\bot], k: 0, R
\]

\[
R \iff \text{if } k = \#L \text{ then } \text{ok} \else \text{if } L k < n \text{ then } B: L k \rightarrow \top \mid B \text{ else } \text{ok} \cdot k := k + 1 \cdot R \fi
\]

Proof of the \( S \) refinement:

\[
B:=[n^*\bot], k:=0, R
\]

\[
= B:=[n^*\bot], k:=0, 0 \leq k \leq \#L \land (\forall i: 0, \ldots, n \cdot B i = \exists j: 0, \ldots, k \cdot L j = i) \implies S
\]

substitution law twice. Note: \( S \) does not mention \( B \) nor \( k \)

\[
= 0 \leq 0 \leq \#L \land (\forall i: 0, \ldots, n \cdot [n^*\bot] i = \exists j: 0, \ldots, \bot \cdot L j = i) \implies S
\]

\[
= 0 \leq 0 \leq \#L \land (\forall i: 0, \ldots, n \cdot \bot = \exists j: 0, \ldots, \bot \cdot L j = i) \implies S
\]

\[
= \top \land \top \implies S
\]

\[
= S
\]

Proof of the \( R \) refinement, by cases. First case

\[
k = \#L \land \text{ok} \implies R
\]

\[
= k = \#L \land k' = k \land B' = B \implies (0 \leq k \leq \#L \land (\forall i: 0, \ldots, n \cdot B i = \exists j: 0, \ldots, k \cdot L j = i) \implies S)
\]

\[
= k = \#L \land k' = k \land B' = B \land 0 \leq k \leq \#L \land (\forall i: 0, \ldots, n \cdot B i = \exists j: 0, \ldots, k \cdot L j = i)
\]

\[
\implies (\forall i: 0, \ldots, n \cdot B'i = \exists j: 0, \ldots, \#L \cdot L j = i)
\]

\[
= \top
\]

For the second case of the \( R \) refinement, I start with a subexpression of the right side.

\[
\text{if } L k < n \text{ then } B: L k \rightarrow \top \mid B \text{ else } \text{ok} \cdot k := k + 1 \cdot R \fi
\]

\[
= \text{if } L k < n \text{ then } B: L k \rightarrow \top \mid B \text{ else } \text{ok} \cdot k := k + 1 \cdot R \fi \text{ expand } R \text{ twice}
\]

\[
= \text{if } L k < n \text{ then } 0 \leq k + 1 \leq \#L \land (\forall i: 0, \ldots, n \cdot (L k \rightarrow \top \mid B) i = \exists j: 0, \ldots, k + 1 \cdot L j = i) \implies S
\]

\[
= \text{else } 0 \leq k + 1 \leq \#L \land (\forall i: 0, \ldots, n \cdot B i = \exists j: 0, \ldots, k + 1 \cdot L j = i) \implies S \fi
\]

\[
= \text{UNFINISHED}
\]

Now the second case of the \( R \) refinement, using the previous simplification:

\[
k = \#L \land (\text{if } L k < n \text{ then } B: L k \rightarrow \top \mid B \text{ else } \text{ok} \cdot k := k + 1 \cdot R) \implies R
\]

\[
= \text{UNFINISHED}
\]

\[
= \top
\]

The recursive time is \( \#L \).