Let \( \textit{subject} \) and \( \textit{pattern} \) be two texts. Write a program to do the following. If \( \textit{pattern} \) occurs somewhere within \( \textit{subject} \), natural variable \( h \) is assigned to indicate the beginning of its first occurrence.

It might be best to strengthen the specification to provide an indication if \( \textit{pattern} \) does not occur anywhere in \( \textit{subject} \), but I'll stick with the question as asked.

(a) using any string operators given in Section 2.3.

Define specifications \( P \) and \( Q \) as follows.

\[
P = \leftrightarrow\textit{pattern} \leq \leftrightarrow\textit{subject} \\
\wedge (\exists i: 0,..,\leftrightarrow\textit{subject} \leq \leftrightarrow\textit{pattern}) \\
\Rightarrow \textit{subject} \downarrow h_{i..i+h} \leftrightarrow \textit{pattern} = \textit{pattern}
\]

\[
Q = \leftrightarrow\textit{pattern} \leq \leftrightarrow\textit{subject} - h \\
\wedge (\exists i: h,..,\leftrightarrow\textit{subject} \leq \leftrightarrow\textit{pattern}) \\
\Rightarrow \textit{subject} \downarrow h_{i..i+h} \leftrightarrow \textit{pattern} = \textit{pattern}
\]

The refinements are as follows.

\[
P \iff h := 0. \\
Q \iff \text{if } h + \leftrightarrow\textit{pattern} > \leftrightarrow\textit{subject} \text{ then ok} \\
\text{else if } \textit{subject} \downarrow h_{i..i+h} \leftrightarrow \textit{pattern} = \textit{pattern} \text{ then ok else } h := h + 1. \\
Q \text{ fi fi}
\]

The proofs are as follows. First the \( P \) refinement.

\[
h := 0. \\
Q \text{ expand } Q \text{ , substitution law, simplify} \Rightarrow P
\]

Now the \( Q \) refinement.

NOT YET DONE

Finally, recursive time, which ignores the time for the string comparison

\[
\textit{subject} \downarrow h_{i..i+h} \leftrightarrow \textit{pattern} = \textit{pattern}
\]

NOT YET DONE

(b) using string indexing, but no other string operators.

The program in part (a) has only one string comparison, namely

\[
\textit{subject} \downarrow h_{i..i+h} \leftrightarrow \textit{pattern} = \textit{pattern}
\]

To replace it with string indexing, introduce binary variable \( m \) (for match), and natural variable \( n \). We need two more specifications.

\[
R = h + \leftrightarrow\textit{pattern} \leq \leftrightarrow\textit{subject} \Rightarrow m = (\textit{subject} \downarrow h_{i..i+h} \leftrightarrow \textit{pattern} = \textit{pattern}) \wedge h' = h
\]

\[
S = h \leq n \leq h + \leftrightarrow\textit{pattern} \leq \leftrightarrow\textit{subject} \Rightarrow m = (\textit{subject} \downarrow n_{i..i+h} \leftrightarrow \textit{pattern} = \textit{pattern}) \wedge h' = h
\]

Now the refinements.

\[
R \iff n := h. \\
S \iff \text{if } n = h + \leftrightarrow\textit{pattern} \text{ then } m := \top \\
\text{else if } \textit{subject} \downarrow n_{i..i+h} \text{ then } n := n + 1. \\
\text{else } m := \bot \text{ fi fi}
\]

And the proofs. First the \( R \) refinement.

\[
n := h. \\
S \text{ expand } S \text{ , substitution law, simplify} \Rightarrow R
\]

Now the \( S \) refinement.

NOT YET DONE

Finally, recursive time, which counts the time for the string comparison.

NOT YET DONE

Now we put it all together, as follows.

\[
P \iff h := 0. \\
Q \iff \text{if } h + \leftrightarrow\textit{pattern} > \leftrightarrow\textit{subject} \text{ then ok} \\
\text{else } R. \text{ if } m \text{ then ok else } h := h + 1. \\
Q \text{ fi fi}
\]

We can optimize a little, by redefining \( S \), and re-refining as follows.
\[ P \leftarrow h := 0. \]
\[ Q \leftarrow \text{if } h + \leftrightarrow \text{pattern} > \leftrightarrow \text{subject} \text{ then } \text{ok} \text{ else } n := h. \ S \text{ fi} \]
\[ S \leftarrow \text{if } n = h + \leftrightarrow \text{pattern} \text{ then } \text{ok} \]
\[ \text{else if } \text{subject}_n = \text{pattern}_{n-h} \text{ then } n := n+1. \ S \text{ else } h := h+1. \ Q \text{ fi fi} \]