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 \textit{h} is assigned to indicate the beginning of its first occurrence.

§ It might be best to strengthen the specification to provide an indication that \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 \textit{P} and \textit{Q} as follows.
\[
P = \leftrightarrow \text{pattern} \leq \leftrightarrow \text{subject} \\
\land (\exists i: 0,.. \leftrightarrow \text{subject} - \leftrightarrow \text{pattern}, \text{subject}_{i+} = \text{pattern}) \\
\Rightarrow \text{subject}_{h..h'} = \text{pattern}
\]
\[
Q = \leftrightarrow \text{pattern} \leq \leftrightarrow \text{subject} - \textit{h} \\
\land (\exists i: h,.. \leftrightarrow \text{subject} - \leftrightarrow \text{pattern}, \text{subject}_{i+} = \text{pattern}) \\
\Rightarrow \text{subject}_{h'..h'} = \text{pattern}
\]
The refinements are as follows.
\[
P \leftarrow h := 0. \ Q
\]
\[
Q \leftarrow \text{if } h + \leftrightarrow \text{pattern} > \leftrightarrow \text{subject} \text{ then ok} \\
\text{else if } \text{subject}_{h..h'} = \text{pattern} \text{ then ok else } h := h + 1. \ Q \text{ fi fi}
\]
The proofs are as follows. First the \textit{P} refinement.
\[
h := 0. \ Q \quad \text{expand } Q, \text{ substitution law, simplify}
\]
\[
= P
\]
Now the \textit{Q} refinement.
NOT YET DONE
Finally, recursive time, which ignores the time for the string comparison.
\[\text{subject}_{h..h'} = \text{pattern}\]
NOT YET DONE

(b) using string indexing, but no other string operators.
§ The program in part (a) has only one string comparison, namely
\[\text{subject}_{h..h'} = \text{pattern}\]
To replace it with string indexing, introduce binary variable \textit{m} (for match), and natural variable \textit{n}. We need two more specifications.
\[
R = h + \leftrightarrow \text{pattern} \leq \leftrightarrow \text{subject} \Rightarrow m' = (\text{subject}_{h..h'} = \text{pattern}) \land h' = h
\]
\[
S = h \leq n \leq h + \leftrightarrow \text{pattern} \leq \leftrightarrow \text{subject} \Rightarrow m' = (\text{subject}_{n-h} = \text{pattern}) \land h' = h
\]
Now the refinements.
\[
R \leftarrow n := h. \ S
\]
\[
S \leftarrow \text{if } n = h + \leftrightarrow \text{pattern} \text{ then } m := \top \\
\text{else if } \text{subject}_{n-h} = \text{pattern} \text{ then } n := n + 1. \ S \text{ else } m := \bot \text{ fi fi}
\]
And the proofs. First the \textit{R} refinement.
\[
n := h. \ S \quad \text{expand } S, \text{ substitution law, simplify}
\]
\[
= R
\]
Now the \textit{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.
\[
P \leftarrow h := 0. \ Q
\]
\[
Q \leftarrow \text{if } h + \leftrightarrow \text{pattern} > \leftrightarrow \text{subject} \text{ then ok} \\
\text{else } R. \text{ if } m \text{ then ok else } h := h + 1. \ Q \text{ fi fi}
\]
We can optimize a little, by redefining \textit{S}, and re-refining as follows.
\[ P \iff h := 0. \ Q \]
\[ Q \iff \text{if } h + \leftrightarrow \text{pattern} > \leftrightarrow \text{subject} \text{ then ok else } n := h. \ S \text{ fi} \]
\[ S \iff \text{if } n = h + \leftrightarrow \text{pattern} \text{ then ok} \]
\[ \text{else if } \text{subject}_n = \text{pattern}_{n-h} \text{ then } n := n + 1. \ S \text{ else } h := h + 1. \ Q \text{ fi fi} \]