Exercise 4.1:
Show with the help of the Prolog tree how the cut is used in the following program,
\[
\begin{align*}
  r(a). \\
  r(b). \\
  q(a) &\leftarrow r(X), !, p(a). \\
  q(f(X)) &\leftarrow r(X). \\
  p(X) &\leftarrow r(X). \\
  p(f(X)) &\leftarrow q(X), !, r(X). \\
  p(g(X)) &\leftarrow r(X).
\end{align*}
\]
and where the query \(?- p(X).\) is taken. What would happen without the cut?

Exercise 4.2:
Consider the following program together with the query \(?- r(X).\)
\[
\begin{align*}
  q(b). \\
  r(a). \\
  s(b). \\
  p(X) &\leftarrow q(X), s(X), !. \\
  p(X) &\leftarrow r(X). \\
  r(X) &\leftarrow s(X). \\
  r(X) &\leftarrow p(X), !, q(X).
\end{align*}
\]
a) Show with the help of the Prolog tree how the cut is used, i.e. indicate explicitly, if branches are eliminated from the tree.

b) Give the output in the order of the computation.
Exercise 4.3:
The built-in predicate `fail/0`, fails when Prolog encounters it as a goal. Thus, it can be viewed as an instruction for backtracking. On the other hand, the cut predicate `!`, blocks backtracking.

Define the predicate `neg/1` which allows you to express *negation as failure*.