Exercise 1 (Fixed-point Combinator)

- Use a fixed-point combinator to compute the length of lists on the encoding given in the last tutorial.
- Find an easier solution for the encoding from the last homework.

Exercise 2 (β-reduction on de Bruijn Preserves Substitution)

We consider an alternative representation of λ-terms that is due to de Bruijn. In this representation, λ-terms are defined according to the following grammar:

\[ d ::= i \in \mathbb{N} \mid d_1 \; d_2 \mid \lambda \; d \]

Define substitution and β-reduction on de Bruijn terms.
Now restate Lemma 1.2.5 for de Bruijn terms and prove it:

\[ s \rightarrow_\beta s' \implies s[u/x] \rightarrow_\beta s'[u/x] \]
Homework 3 (Multiplication)

Define multiplication using fix and prove its correctness. You can assume that you are given a predecessor function \( \text{pred} \) such that:

- \( \text{pred} \ 0 \rightarrow^* 0 \)
- \( \text{pred} \ (\text{succ} \ n) \rightarrow^* n \)

Homework 4 (Efficient Substitution on de Bruijn)

We define a new lifting operator \(-\uparrow^\_\-\):

\[
i \uparrow^l_i \equiv \begin{cases} 
  i, & \text{if } i < l \\
  i + n, & \text{if } i \geq l 
\end{cases}
\]

\[(d_1 \ d_2) \uparrow^l_i \equiv d_1 \uparrow^l_i \ d_2 \uparrow^l_i \]

\[(\lambda \ d) \uparrow^l_i \equiv \lambda \ d \uparrow^{l+1}_i \]

Use \(-\uparrow^\_\-) to define a more efficient version of substitution for de Bruijn terms that only applies lifting in the case that a variable is actually replaced by a term. Prove that \( t[s/0] \) yields the same result for both, your new version and the version from the tutorial. *Hint:* Find a suitable generalization first.

Homework 5 (Expanding Lets)

We have a language with let-expressions, i.e.:

\[
t ::= v \mid t \ t \mid \text{let } v = t \text{ in } t
\]

Write a program which expands all \text{let} -expressions. The \text{let}-semantics are:

\[
(\text{let } v = t_1 \text{ in } t_2) = (\lambda v. t_2) \ t_1
\]

If you want to use a language different from ML, Ocaml, Haskell, Java, and Python, please talk to the tutor first.