Induktionsvorlage

Diese Beweise werden mit dem Tool cyp[1] getestet.

```
Lemma: P(xs)

Proof by induction on List xs

Case []
  To show: P(xs)

  Proof
      -- show P([])
  QED

Case x:xs
  To show: P(x:xs)
  IH: P(xs)

  Proof
      -- show P(x:xs) using IH P(xs)
  QED

QED
```

Syntaxhinweise:

- Als Gleichheitszeichen muss `.=.` verwendet werden
- Jeden Schritt in eigene Zeile
- Gleichheitszeichen + Regel am Anfang
- `--` leitet einen Kommentar ein
- Die Beweisschritte (in einem `Case`) müssen eingerückt werden
- Groß-/Kleinschreibung ist relevant
- In jedem Induktionsfall muss die zu beweisende Aussage explizit aufgeführt werden
- Benutzte Induktionshypothesen müssen explizit aufgeführt werden

[1] `https://github.com/noschinl/cyp`

Beispiel

- Jeder Schritt muss mit `by Regel` gerechtfertigt werden
- Die `Regel` kann sein:
  - `def f` für eine Funktion `f`
  - `IH` für die Induktionshypothese
  - Der Name eines beliebigen Axioms. Diese werden explizit im Übungsplatz genannt.

```
Lemma: length(xs ++ ys) .=. length xs + length ys

Proof by induction on List xs

Case []

    To show: length ([] ++ ys) .=. length [] + length ys

    Proof

                            length ([] ++ ys)
        (by def ++)         .=. length ys

                            length [] + length ys
        (by def length)     .=. 0 + length ys
        (by arith)          .=. length ys

    QED

Case x:xs

    To show: length ((x:xs) ++ ys) .=. length (x:xs) + length ys
    IH: length (xs ++ ys) .=. length xs + length ys

    Proof

                            length ((x : xs) ++ ys)
        (by def ++)         .=. length (x : (xs ++ ys))
        (by def length)     .=. 1 + length (xs ++ ys)
        (by IH)             .=. 1 + (length xs + length ys)
        (by arith)          .=. 1 + length xs + length ys

                            length (x : xs) + length ys
        (by def length)     .=. 1 + length xs + length ys

    QED

QED
```