Aufgabe 1

Wir haben in der Vorlesung gelernt was eine Substitution ist und welche Substitutionen zulässig sind. "Zulässiges Substitutieren" kann induktiv wie folgt definiert werden:

- $v[x := A] = v$ für Variablen $x \neq v$
- $x[x := A] = A$
- $c[x := A] = c$ für Konstanten c
- $(A\ B)[x := C] = (A[x := C]\ B[x := C])$
- $(\lambda y.A)[x := B] = \lambda y.(A[x := B])$ falls $y \neq x$ und $y \notin FV(B)$
- $(\lambda x.A)[x := B] = \lambda x.A$
- $(\lambda y.A)[x := B] = \lambda z.((A[y := z])[x := B])$ mit einer "frischen Variablen" z, wenn $x \neq y$ und $y \in FV(B)$.

Wir wollen das im Unterricht begonnene "λ Programm" um eine Funktion

```
substitute:  Term -> string -> Term -> Term,
```

die im ersten Argument (vom Typ Term) das zweite Argument (Variable als String gegeben) durch das dritte Argument (wieder vom Typ Term) substituiert. Es soll also

```
substitute t1 var t2 = t1[var := t2]
```

gelten. Es soll dabei darauf geachtet werden, dass "zulässig" substituiert wird. Um dies zu erreichen, erledigen Sie der Reihe nach folgende Punkte:

- Implementieren Sie eine Funktion

  ```
  isFreeVar :  string -> Term -> bool,
  ```

  die entscheidet ob eine Variable frei in einem Term vorkommt. Benützen Sie dazu die Funktion FV/`freeVar` aus der Vorlesung.

- Implementieren Sie eine Funktion

  ```
  freshVar :  Term -> string,
  ```

  die zu einem gegebenen Term einen String erzeugt, der nicht als Name im Term vorkommt.

- Benutzen Sie nun die beiden Funktonen `freshVar` und `isFreeVar` um die Funktion `substitute` gemäss der "induktiven Anleitung" am Anfang zu implementieren.