Die Peter'sche Funktion p ist definiert durch die Rekursionsgleichungen

$$\begin{aligned} p(0,y) &= y' \\ p(x',0) &= p(x,1) \\ p(x',y') &= p(x,p(x',y)). \end{aligned}$$

Im Prolog kann man nicht direkt Funktionen definieren, sondern nur Prädikate. Zur Darstellung einer Funktion definiert man ihren Graphen als Prädikat. So schreibt man etwa in Prolog `p(X,Y,Z)` für $p(x,y) = z$. Variablen müssen ja in Prolog groß geschrieben werden. Im Programm `peter.pl` habe ich die Prologvariable `X1` für x' und `Y1` für y' verwendet. Das Programm `peter.pl` zeigt nur das Endergebnis der Berechnung, nicht die einzelnen Berechnungsschritte.

Das Programm `peter_red.pl` zeigt die einzelnen Berechnungsschritte an, zum Beispiel für die Berechnung

$$\begin{aligned} p(1,2) = p(0,p(1,1)) &= p(0,p(0,p(1,0))) \\ &= p(0,p(0,p(0,1))) = p(0,p(0,2)) = p(0,3) = 4. \end{aligned}$$

Hierzu müssen wir die dabei auftretenden Terme, z.B. den Term $p(0,p(0,p(1,0)))$ selbst als Prologterm `p(0,p(0,p(1,0)))` darstellen, nicht wie bei `peter.pl` ein 3-stelliges Prädikat `p` definieren. Und wir müssen uns anschauen, wie man die Terme der Berechnung nacheinander erhält, nämlich durch Anwendung der Rekursionsgleichungen von links nach rechts, bis als Ergebnis eine Zahl herauskommt. Wir betrachten also die Rekursionsgleichungen als Ersetzungsregeln:

$$\begin{aligned} p(0,y) &\to y' \\ p(x',0) &\to p(x,1) \\ p(x',y') &\to p(x,p(x',y)). \end{aligned}$$

Wenn wir in einer dieser Regeln für x und y konkrete Zahlen einsetzen, schreiben wir $s \to t$, wobei s der Term auf der linken Seite und t der Term auf der rechten Seite ist. Es gilt zum Beispiel $p(0,3) \to 4$ und $p(3,0) \to p(2,1)$ und $p(2,4) \to p(1,p(2,3))$. Wenn wir nun $p(1,2)$ berechnen wollen, reduzieren wir den Term $p(1,2)$ schrittweise zu einer Zahl, also zu einem nicht weiter reduzierbaren Term (einer *Normalform*). Wir schreiben dann einen langen Pfeil $\longrightarrow$ statt dem Gleichheitszeichen $=$, um die Richtung der Reduktion anzudeuten:

$$\begin{aligned} p(1,2) \longrightarrow p(0,p(1,1)) &\longrightarrow p(0,p(0,p(1,0))) \\ &\longrightarrow p(0,p(0,p(0,1))) \longrightarrow p(0,p(0,2)) \longrightarrow p(0,3) \longrightarrow 4. \end{aligned}$$

Dabei wenden wir jeweils eine Regel auf einen Teilterm des zu reduzierenden Terms an. Beispiel: $p(0,p(0,p(1,0))) \longrightarrow p(0,p(0,p(0,1)))$, weil $p(1,0) \to p(0,1)$. Wir sagen dann, der Term $p(0,p(0,p(1,0)))$ reduziere sich in einem Schritt zum Term $p(0,p(0,p(0,1)))$. Wenn ein Term s in null oder mehr Schritten zu einem Term t reduziert werden kann, schreiben wir $s \longrightarrow^* t$.

Bei der Implementierung als Prologprogramm wäre es naheliegend, $\to$ durch `->` und $\longrightarrow$ durch `-->` darzustellen. Da die aber beide in Prolog schon eine andere Bedeutung haben, verwenden wir Doppelpfeile: `=>` für $\to$, `==>` für $\longrightarrow$ und `==>*` für $\longrightarrow^*$. Im Programm sind `=>`, `==>` und `==>*` als zweistellige Prädikate definiert und als Infixoperatoren deklariert, d.h. wir dürfen z.B. `S=>T` statt `=>(S,T)` schreiben. Beachten Sie, dass für alle auftretenden Terme der Form $p(s,t)$ das s eine Zahl ist, nicht selbst ein mittels p zusammengesetzter Term!