



9. Übung zur Vorlesung „Logikprogrammierung“
Sommersemester 2003

Hinweis: Die Vorlesungen am 10. und am 12.6., sowie die Übung am 11.6. fallen aus!

Aufgabe 29 (Programmieraufgabe) 5 Punkte

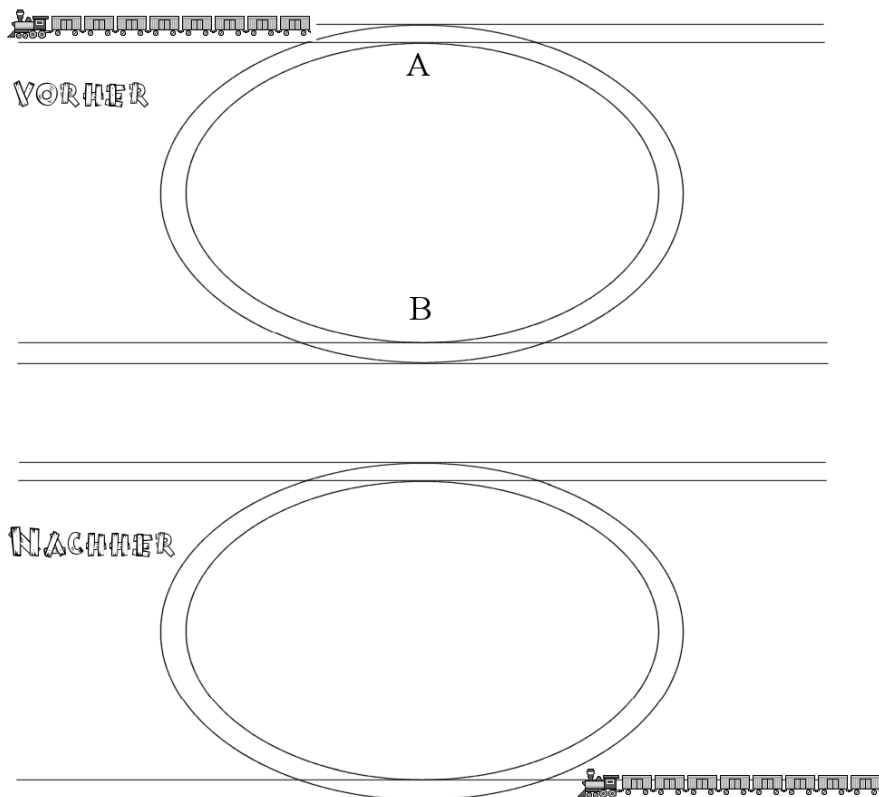
Schreiben Sie ein Prädikat `fib`, welches eine Zahl n einliest, in linearer Zeit $fibonacci(n)$ berechnet und diesen Wert ausgibt.

Aufgabe 30 (Programmieraufgabe) 5 Punkte

N Kannibalen und M Missionare (wobei $N \leq M$) erreichen auf einem gemeinsamen Fußmarsch einen Fluß. Am Flußufer liegt ein Boot, das maximal K Personen faßt. Das Boot kann den Fluß mit weniger als K Personen jedoch nicht ganz leer überqueren. Die Gruppe soll den Fluß überqueren, aber aus naheliegenden Gründen muß vermieden werden, daß sich jemals auf einem Ufer n Kannibalen und m Missionare befinden mit $0 < m < n$. Schreiben Sie ein Prolog-Programm, das dieses Übersetzungsproblem löst (falls möglich) und testen Sie es mit $N = M = 3$ und $K = 2$.

Aufgabe 31 (Programmieraufgabe) 6 Punkte

Gegeben ist die folgende Gleiskonstellation:



Ein Güterzug (Lok plus n Wagen) soll, wie gezeigt, von dem oberen auf den unteren Gleisstrang wechseln. Als Randbedingung ist zu beachten, daß auf jedem der Halbkreise maximal 4 Wagen geparkt werden können. Auf einer Weiche darf niemals geparkt werden. Lösen Sie dieses Problem in Prolog.

Aufgabe 32 (Programmieraufgabe)

6 Punkte

Definieren Sie in Prolog ein Prädikat `diamond(N)`, das für eine gegebene natürliche Zahl N auf dem Bildschirm das folgende Bild ausgibt:

```
?- diamond(3)
  1
 4  2
7  5  3
 8  6
  9
?- diamond(6)
          1
        7  2
      13  8  3
    19 14  9  4
 25 20 15 10  5
31 26 21 16 11  6
 32 27 22 17 12
    33 28 23 18
      34 29 24
        35 30
          36
```

Aufgabe 33 (Programmieraufgabe)

8 Punkte

In Aufgabe 11 haben wir einen Termsimplifikator implementiert. Dieser konnte jedoch nur Terme mit einer speziellen Struktur vereinfachen. In der Vorlesung wurde vorgestellt, wie man beliebige Terme in ihre Einzelteile zerlegen kann.

Implementieren Sie nun einen Simplifikator, der beliebige Prolog-Terme vereinfacht. Er sollte wenigstens folgende Regeln beherrschen:

- konstante Terme werden evaluiert (also z.B. $4 * 3$ wird zu 12).
- $0 * t = 0$
- $0 + t = t$
- $t * c = c * t$ ($3 * x$ wird als einfacher angesehen als $x * 3$)
- $c_1 * t + c_2 * t = (c_1 + c_2) * t$
- $x^{c_1} * x^{c_2} = x^{c_1+c_2}$

Die Menge möglicher Vereinfachungsregeln ist nach oben offen. Überlegen Sie sich selbst einige gute Vereinfachungsregeln und implementieren Sie auch diese (Kommentare nicht vergessen!).

Abgabe: Übernächste Woche in der Vorlesung am Dienstag, den 17.6.2003