1. Rekursive Vergleiche – Braunkohle

Das in der Übung vorgestellte „Vergleiche auf Braunkohle“ Beispiel hat die Schwäche, dass es zwar ausgibt wenn ein Datensatz größer gleich ist, aber nicht das entsprechende Jahr dazu. Schreiben Sie das Programm so um, dass das Jahr auf dem Bildschirm ausgegeben wird.

Hinweis: Das Prädikat `write/1` kann genutzt werden um ein Atom auf dem Bildschirm auszugeben.

So gibt `write(something)` das Atom `something` aus. Ebenso gibt `X=something, write(X).` den Inhalt der instanziierten Variable `X` auf dem Bildschirm aus, nämlich wiederum `something`.

2. Zusammenfügen von Listen

Erstellen Sie (einfache) englische Sätze der Form

„Subjekt Verb Adjektiv Artikel Objekt“

. Wobei sowohl Adjektiv als auch Artikel optional sein sollen.

(Beispiel: I learn rapidly prolog)

Benutzen Sie folgende Wörter:

Subjekte I, you, we

Verben eat, throw, plant

Adjektive lazy, ravenous

Artikel a, the

Objekte apple, plum, kumquad

Bilden Sie aus diesen Wörtern alle möglichen Sätze mit Hilfe eines Prologprogramms.

Hinweis: Gehen Sie wie bei der Autoteileaufgabe aus dem Übungsskript vor. Bilden Sie eine Wortbasis und spezifizieren Sie dann grammatische Strukturen.

3. Fakultät

Schreiben Sie ein Programm, das die Fakultät berechnet. Die rekursive Definition der Fakultät für $n \in \mathbb{N}_0$ lautet:

$$n! = \begin{cases} 1, & n = 0 \\ n \cdot (n-1)!, & n > 0 \end{cases}$$

4. Finde den Fehler

Finden sie in folgender Datenbasis alle sechs Fehler. Es soll nur eine Ausgabe generiert werden!

```
1 /* Programm soll eine Folge y=2*x aller natürlichen Zahlen (inklusive 0) bis zu einer beim
      ↪ Start angegebenen Zahl berechnen und sie in einer Liste ablegen */
2 /* Die Werte in der Liste werden in absteigender Reihenfolge abgelegt! */

4 folge(Zahl, Folge) :- Zahl < 1, Folge=[].

6 folge(Zahl, [Erg, Y]) :-
7    Vor = Zahl -1,
8    folge(Vor, Erg),
9    Y = Zahl * 2.
```

5. sichere Sicherheit

Programmieren Sie für ein Zugangssystem die Entscheidungsfindung. Es stehen zwei Sensoren zur Verfügung.

RFID Scanner Alle Mitarbeiter sind mit einem Ausweis mit RFID Chip ausgerüstet. Der Scanner ließt die Werte vom Chip und gibt dann entweder `Zugang` oder `kein Zugang` zurück.

Auf Intervention seitens des Managements wurde ein weiterer Fall geschaffen.

Im Falle, dass der Mitarbeiter seinen Ausweis nicht mit hat oder dieser nicht gelesen werden kann gibt der Sensor `kein Wert` zurück.

Biometrischer Sensor Um Diebstahl der Ausweise zu verhindern wird jede Person biometrisch gescannt und mit einer internen Datenbank verglichen. Dies liefert eine Liste mit Vertrauenswahrscheinlichkeiten. Die Länge entspricht der Anzahl der gespeicherten Daten und kann stark schwanken.

Anforderungen

- Das Programm muss Backtracking-sicher ausgeführt werden.
- Personen, die der RFID Sensor mit `kein Zugang` bewertet, dürfen keinen Zugang erhalten
- Für Personen mit `Zugang` darf keine der Vertrauenswahrscheinlichkeiten unter 60% liegen
- Für Personen mit `kein Wert` nicht unter 80%
- In beiden Fällen muss die Länge der Liste mindestens drei Werte betragen

Hinweise Der Test sollte auf einen Namen, den Wert des RFID Sensors und die Liste von Wahrscheinlichkeitswerten erfolgen.

Wie etwa Beispielsweise:

```
zugang('John␣Exmitarbeiter', 'kein␣Zugang', [0.4, 0.3, 0.8, 0.9, 0.6]).
false

zugang('CEO', 'kein␣Wert',[0.9, 1.0, 0.9, 0.8]).
true

zugang('Hacker␣MakHackerson', 'Zugang', [1.0, 1,0]).
false

zugang('Joe␣Normalo', 'Zugang', [1.0, 0.7, 0.6, 0.8, 0.9, 0.8]).
true
```