1. Finde den Fehler

Finden sie in folgender Datenbasis alle acht Fehler. Wiederholungsfehler zählen als ein Fehler.

```
1  /* startflughafen, zielflughafen, Fluggesellschaft
2  flugRoute(Dresden, Mallorca, Eurowings)
3  flugRoute(Dresden, Zürich, Swiss)
4  flugRoute(Dresden, Hurghada, Germania)
5  flugRoute(Dresden, Malaga, Eurowings)
6  flugRoute(Dresden, Frankfurt, Lufthansa)
7  flugRoute(Mallorca, Zürich, Vueling)
8  flugRoute(Mallorca, Frankfurt, Lufthansa)
9  flugRoute(Mallorca, Dresden, Eurowings)
10 flugRoute(Mallorca, Ibiza, Iberia)
11 flugRoute(Mallorca, Malaga, Ryanair)
12 betriebsdaten(Ryanair, flugzeuge(0, ))
13 betriebsdaten(Eurowings, flugzeuge(2, [flugzeug(Boeing, treibstoffVerbrauch(1.4))]))
14 betriebsdaten(Swiss, flugzeuge(2, [flugzeug(Boeing, treibstoffVerbrauch(2.3))]))
15 betriebsdaten(Germania, flugzeuge(1, [flugzeug(Airbus, treibstoffVerbrauch(0.9))]))
16 betriebsdaten(Lufthansa, flugzeuge(3, [flugzeug(Airbus, treibstoffVerbrauch(1.3))]))
17 betriebsdaten(Vueling, flugzeuge(2, [flugzeug(Dornier, treibstoffVerbrauch(1.8))]))
18 betriebsdaten(Iberia, flugzeuge(1, [flugzeug(Boeing, treibstoffVerbrauch(1.2))]))

20 return_Flug(X,Y) :- flugRoute(X,Y) flugRoute(Y,X)
21 /* Der Betrieb ist gefährdet wenn eine Flugroute existiert und keine Flugzeuge verfügbar sind
       ↪ */
22 betriebGefährdet(X) :- flugRoute(_,_,X), betriebsdaten(X, flugzeuge(Y, _)), istGleich(Y, 0)
23 /* Der gesamte TreibstoffVerbrauch errechnet sich aus dem Verbrauch aller Flugzeuge */
24 gesamtTreibstoffVerbrauch(Fluglinie, X) :- betriebsdaten(Fluglinie, flugzeuge(Y, [flugzeug(_,
       ↪ treibstoffVerbrauch(Z))|_])), X = Y * Z
```

Lösung:

1. In Zeile 1 fehlt das Kommentarbeendungszeichen
   `/*startflughafen, zielflughafen, Fluggesellschaft */`
2. Großgeschriebene Terme sind Variablen! Entweder diese klein schreiben oder escapen! Zum Beispiel: `flugRoute('Dresden', 'Mallorca', 'Eurowings').`
3. Zeilen müssen mit einem Punkt beendet werden!
   Zum Beispiel: `flugRoute('Dresden', 'Mallorca', 'Eurowings').`
4. In Zeile 12 fehlt ein Parameter. Die leere Liste in diesem Fall.
   `betriebsdaten('Ryanair', flugzeuge(0, [])).`
5. In Zeile 20 wird ein nicht definierter Fakt zum Suchen benutzt. Der korrekte Fakt hat drei Parameter. Für diesen Zweck kann die anonyme Variable genutzt werden.

6. Ebenso in Zeile 20 ist die Konjunktion (Komma) zwischen den Suchfakten vergessen worden.
   `return_Flug(X,Y):- flugRoute(X,Y,_), flugRoute(Y,X,_).`
7. In Zeile 22 wird eine nicht definierte Regel benutzt.
   Entweder `istGleich/2` definieren oder ersetzen.
   `istGleich(X,Y):- X=:=Y.`
8. In Zeile 24 muss die Berechnung mit dem `is` Schlüsselwort durchgeführt werden!
   `X is Y * Z`

2. Vergleiche

Verändern Sie die `vergleich/1` Regel, so dass diese die Datenbasis vergleicht. Atome und Strukturen sollen lediglich auf Gleichheit getestet werden. Zahlen hingegen sollen auf größer, kleiner oder gleich getestet werden.

Hinweis: Nutzen sie das `number/1` bzw. **`atom`**`/1` Prädikat um zu testen was für ein Input vorliegt. Ausgaben können mit dem **`write`**`/1` Prädikat erfolgen (z.B. **`write`**`("Ausgabe")`). Trennen Sie unterschiedliche logische Zweige mit einer Disjunktion (`;`).

```
1  test(12).
2  test(11.0).
3  test(7.22).
4  test(3.141e2).
5  test(42).

7  test(hilfe).
8  test(ein).
9  test(atom).
10 test(true).
11 test(false).

13 test(resolution(hunger, essen)).
14 test(resolution(hunger, essen)).
15 test(resolution(müde, schlafen)).

17 vergleich(Input):- true.
```

Lösung:

```
1  vergleich(Input):-
2      test(X),
3      (
4          (
```

```
5           number(Input), number(X),
6           (
7               (X=:=Input, write(X), write("␣Even␣"), write(Input));
8               (X>Input, write(X), write("␣Bigger␣"), write(Input));
9               (X<Input, write(X), write("␣Lower␣"), write(Input))
10          )
11      );
12      (
13          X=Input, write(X), write("␣Gleich␣"), write(Input)
14      )
15  ).
```

3. 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`.

Lösung:

```
22  braunkohle_förderung(nordkorea, [5.2, 10.0, 10.6, 7.2, 9.0, 9.0, 9.0, 7.0, 7.6, 7.0, 7.0,
        ↪ 7.0, 7.0, 7.0]).

24  spalten_kopf([1970,1980,1990,2000,2007,2008,2009,2010,2011,2012,2013,2014,2015,2016]).

26  gleich_oder_höhere_förderung(Niedrig, Hoch) :-
27      Niedrig =< Hoch.

29  manchmal_besser([X|_],[Y|_], [Jahr|_]) :-
30      gleich_oder_höhere_förderung(X,Y),
31      write(Jahr).

33  manchmal_besser([_|X], [_|Y], [_|Rumpf]) :-
34      manchmal_besser(X, Y, Rumpf).

36  höhere_Förderung_pro_Jahr(Land1, Land2) :-
37      braunkohle_förderung(Land1, Daten1),
38      braunkohle_förderung(Land2, Daten2),
39      Land1 \= Land2,
40      spalten_kopf(Jahre),
41      manchmal_besser(Daten1, Daten2, Jahre).
```

4. 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.

Lösung:

```
einfaches_wort(apple).
einfaches_wort(plum).
einfaches_wort(kumquad).
einfaches_wort(the).
einfaches_wort(a).
einfaches_wort(eat).
einfaches_wort(throw).
einfaches_wort(plant).
einfaches_wort(i).
einfaches_wort(you).
einfaches_wort(we).
einfaches_wort(ravenous).
einfaches_wort(lazy).

syntax(ppronomen, [i]).
syntax(ppronomen, [you]).
syntax(ppronomen, [we]).

syntax(adjektiv, [ravenous]).
syntax(adjektiv, [lazy]).

syntax(verb, [eat]).
syntax(verb, [throw]).
```

```
24  syntax(verb, [plant]).

26  syntax(substantiv, [apple]).
27  syntax(substantiv, [plum]).
28  syntax(substantiv, [kumquad]).

30  syntax(artikel, [the]).
31  syntax(artikel, [a]).

33  syntax(substantiv_phrase, [artikel, substantiv]).
34  syntax(substantiv_phrase, [substantiv]).

36  syntax(verb_phrase, [verb, adjektiv]).
37  syntax(verb_phrase, [verb]).

39  syntax(satz, [ppronomen, verb_phrase, substantiv_phrase]).

41  worte_aus(X, [X]) :-
42      einfaches_wort(X).

44  worte_aus(X, P) :-
45      syntax(X, Worte),
46      hole_alle_worte_aus(Worte, P).

48  hole_alle_worte_aus([], []).

50  hole_alle_worte_aus([H|Rest], Satz) :-
51      worte_aus(H, Kopfworte),
52      hole_alle_worte_aus(Rest, Restworte),
53      append(Kopfworte, Restworte, Satz).
```