```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  % Random representation of facts: %
3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4
5  % the population of norway is 4,5 million
6  population(norway, 4500000).
7  % norway is a rich country
8  rich(norway).
9  % 2 is a prime number
10 is_prime(2).
11 % someone wrote hamlet
12 wrote(someone, hamlet).
13 % all humans are mortal
14 mortal(X) :- human(X).
15 % all rich people pay taxes
16 pays_taxes(X) :- rich(X).
17 % ivar takes his umbrella if it rains
18 takes_umbrella(ivar) :- rains.
19 % Firebrigade employees are men over six feet tall.
20 firebrigade_employee(X) :- man(X), over_six_feet(X).
21
22 % italian food
23 italian(pizza).
24 italian(spaghetti).
25 % indian food
26 indian(tandoori).
27 indian(tikkaMasala).
28 indian(curry).
29 % mild food
30 mild(tandoori).
31 % ola likes italian food and mild indian food
32 likes(ola, X) :- italian(X).
33 likes(ola, X) :- indian(X), mild(X).
34 % nationality(Person, From).
35 nationality(ola, nor).
36 nationality(eyvind, nor).
37 nationality(rupinder, ind).
38 nationality(raji, ind).
39 % Norwegians like italian and mild indian food
40 likes(X, Y) :- nationality(X, nor), italian(Y).
41 likes(X, Y) :- nationality(X, nor), indian(Y), mild(Y).
42 % Indians like indian food
43 likes(X, Y) :- nationality(X, ind), indian(Y).
44
45 % star(Star).
46 star(sun).
47 star(vega).
48 star(sirius).
49 % orbits(Orbits, Element).
50 orbits(mercury, sun).
51 orbits(venus, sun).
52 orbits(earth, sun).
53 orbits(mars, sun).
54 orbits(moon, earth).
55 orbits(deimos, mars).
56 % Elements that orbit the sun are planets
57 planet(B) :- orbits(B, sun).
58 % Elements that orbit planets are satellites
59 satellite(B) :- orbits(B, A), planet(A).
60 % the sun, planets and satellites are part of solar
61 solar(sun).
62 solar(X) :- planet(X).
63 solar(X) :- satellite(X).
64
65 % reindeerLine(From, To).
66 reindeerLine(northpole,oslo).
67 reindeerLine(oslo,london).
68 reindeerLine(oslo,copenhagen).
69 reindeerLine(copenhagen,berlin).
```

```
70 reindeerLine(northpole,stockholm).
71 reindeerLine(stockholm,moscow).
72 % connections of reindeerLines
73 connection(X, Y) :- reindeerLine(X, Z), reindeerLine(Z, Y).
74 connection(X, Y) :- reindeerLine(X, Z), connection(Z, Y).
75 % direct connections of reindeerLines
76 direct_connection(X, Y) :- reindeerLine(X, Z), reindeerLine(Z, Y).
77
78 % p(Child, Mother, Father, Birthday).
79 p(anne, aase, aale, 1960).
80 p(arne, aase, aale, 1962).
81 p(beate, anne, lars, 1989).
82 p(bjorn, lise, arne, 1990).
83 child(X, Y) :- p(X, _, Y, _).
84 child(X, Y) :- p(X, Y, _, _).
85 % grandchild
86 grandchild(X, Y) :- child(X, Z), child(Z, Y).
87 % common descendants
88 common_descendants(X, Y) :- child(Z, X), child(Z, Y).
89 common_descendants(X, Y) :- grandchild(Z, X), grandchild(Z, Y).
90 common_descendants(X, Y) :- child(Z1, X), child(Z2, Y), common_descendants(Z1, Z2).
91 % siblings
92 sibling(X, Y) :- child(X, Z), child(Y, Z), X \== Y.
93 % no siblings
94 only_child(X) :- p(X, _, _, _), \+sibling(X, Anyone).
95
96 % boss(Boss, Employee).
97 boss(eva, anne).
98 boss(eva, atle).
99 boss(lars, eva).
100 % X is superior to Y
101 sup(X, Y) :- boss(X, Y).
102 sup(X, Y) :- boss(X, Z), sup(Z, Y).
103 % X has multiple bosses
104 multipleBosses(X) :- boss(B1, X), boss(B2, X), B1 \== B2.
105 % X is not its own boss, X does not have more than one boss
106 ok(X) :- \+boss(X, X), \+multipleBosses(X).
107
108 % local storage (Index, Item).
109 ls(1, ole).
110 ls(2, dole).
111 ls(3, ole).
112 % remote storage (Index, Item).
113 rs(1, ole).
114 rs(2, dole).
115 rs(3, doffen).
116 rs(5, dolly).
117 % find Item(X) at index(I), search local storage first
118 find(I, X) :- ls(I, X); rs(I, X).
119 % Item at Index(I) is different in local/remote
120 error(I) :- ls(I, Res1), rs(I, Res2), Res1 \== Res2.
121 % Item(X) is stored at two different indexes
122 multi(X) :- ls(I1, X), ls(I2, X), I1 \== I2.
123 multi(X) :- rs(I1, X), rs(I2, X), I1 \== I2.
124 multi(X) :- ls(I1, X), rs(I2, X), I1 \== I2.
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
```

```
140
141
142
143 %%%%%%%%%%%%%%%%%%%%%%%%
144 % More advanced facts %
145 %%%%%%%%%%%%%%%%%%%%%%%%
146
147 % empty(Xs).
148 % True if Xs is empty
149 empty([]).
150
151 % one_element_list(Xs).
152 % True if Xs only has one element
153 one_element_list([_]).
154
155 % min_two_element_list(Xs).
156 % True if Xs has at least two elements
157 min_two_element_list([_,_|_]).
158
159 % is_sorted(Xs).
160 % True if Xs is sorted
161 %
162 % eg.     is_sorted([1, 2, 3, 4]).
163 %        yes
164 %        is_sorted([2, 1, 4, 3]).
165 %        no
166 is_sorted([]).
167 is_sorted([_]).
168 is_sorted([X,Y|Z]) :- X =< Y, is_sorted([Y|Z]).
169
170 % split(Xs, Ys, Zs).
171 % True if Ys and Zs zipped is Xs
172 %
173 % eg.     split([1, 2, 3, 4], [1, 3], [2, 4]).
174 %        yes
175 %        split([1, 5, 10, 20], Ys, Zs).
176 %        Ys = [1, 10].
177 %        Zs = [5, 20].
178 split([], [], []).
179 split([X], [X], []).
180 split([X,Y|Xs], [X|Ys], [Y|Zs]) :- split(Xs, Ys, Zs).
181
182 % merge(Xs, Ys, Zs).
183 % True if Xs and Ys merged equals Zs
184 %
185 % eg.     merge([1, 5, 8], [2, 3, 4, 9], Zs).
186 %        Zs = [1, 2, 3, 4, 5, 8, 9].
187 merge([], Ys, Ys).
188 merge(Xs, [], Xs).
189 merge([X|Xs], [Y|Ys], [X|Zs]) :- X =< Y, merge(Xs, [Y|Ys], Zs).
190 merge([X|Xs], [Y|Ys], [Y|Zs]) :- Y    < X, merge([X|Xs], Ys, Zs).
191
192 % mergesort(Xs, Ys).
193 % True if Ys is Xs sorted by mergesort
194 %
195 % eg.     mergesort([1, 5, 3, 10, 9, 8], Ys).
196 %        Ys = [1, 3, 5, 8, 9, 10].
197 mergesort(Xs, []) :- empty(Xs).
198 mergesort(Xs, Xs) :- one_element_list(Xs).
199 mergesort(Xs, Res) :-     min_two_element_list(Xs),
200                           split(Xs, Ys, Zs),
201                           mergesort(Ys, Res1),
202                           mergesort(Zs, Res2),
203                           merge(Res1, Res2, Res).
204
205 % sameHead(Xs, Ys).
206 % True if first element of Xs and Ys are the same
207 %
208 % eg.     sameHead([1, 2, 3], [1, 4, 5]).
209 %         yes
```

```
210 %       sameHead([X, 1], [2, 3]).
211 %       X = 2
212 %       sameHead([1, 2], [3, 4]).
213 %       no
214 sameHead([X|_Xs], [X|_Ys]).
215
216 % zip(Xs, Ys, Zs).
217 % True if Zs is composed of alternating elements
218 % from Xs and Ys from the head
219 %
220 % eg.    zip([1, 2, 3], [a, b, c], [1, a, 2, b, 3, c]).
221 %       yes
222 %       zip([1], [a, b], [1, a, b]).
223 %       no
224 zip([], [], []).
225 zip([X|Xs], [Y|Ys], [X,Y|Zs]) :- zip(Xs, Ys, Zs).
226
227 % stutter(Xs).
228 %
229 % eg.     stutter([a, a, b, b]).
230 %       yes
231 %%       stutter([1, 1, 1, 1, 1, 1]).
232 %       yes
233 %       stutter([1, a, a, 1]).
234 %       no
235 stutter(XS) :- zip(X, X, XS).
236
237 % append3(Xs, Ys, Zs, Rs).
238 % True if Rs is a concatenated list of Xs, Ys and Zs
239 %
240 % eg.    append([1, 2], [a, b], [3, 4], Rs).
241 %       Rs = [1, 2, a, b, 3, 4].
242 append3([], [], Zs, Zs).
243 append3([], [Y|Ys], Zs, [Y|Rs]) :- append3([], Ys, Zs, Rs).
244 append3([X|Xs], Ys, Zs, [X|Rs]) :- append3(Xs, Ys, Zs, Rs).
245
246 % sublist(Xs, Ys).
247 % True if Xs is a sublist of Ys
248 %
249 % eg.     sublist([a, b], [1, 2, a, b, 3]).
250 %       yes
251 %       sublist(Xs, [a, b]).
252 %       Xs = [a, b] ;
253 %       Xs = [b]    ;
254 %       Xs = []     ;
255 %       Xs = [a]    ;
256 %       false.
257 sublist(Xs, Ys) :- append3(_, Xs, _, Ys).
258
259 % onestep(Xs, Ys).
260 % True if Ys is Xs with one rewrite rules applied
261 %
262 % eg.     onestep([a, b, c, c, b, a], Ys).
263 %       Ys = [b, a, c, c, b, a] ; first rule
264 %       Ys = [a, b, c, b, a]    ; second rule
265 rewrite([a, b], [b, a]).
266 rewrite([c, c], [c]).
267
268 onestep(Xs, Ys) :-     rewrite(A, B),
269                     append3(X, A, Y, Xs),
270                     append3(X, B, Y, Ys).
271
272 % manystep(Xs, Ys).
273 % True if Ys is Xs with one or more rewrite rules applied
274 manystep(Xs, Xs).
275 manystep(Xs, Ys) :- onestep(Xs, Zs), manystep(Zs, Ys).
276
277 % normalform(Xs, Ys).
278 % True if Ys is Xs with all possible rewrite rules applied
279 normalform(Xs, Ys) :- manystep(Xs, Ys), \+onestep(Ys, _Zs).
```

```
280
281 % toList(Xt, Ys).
282 % True if Ys is the tree Xt represented as a list
283 %
284 % eg.    toList(node(leaf(1), 4, node(leaf(5), 6, leaf7)), Y).
285 %       Y = [1, 4, 5, 6, 7].
286 toList(leaf(X), [X]).
287 toList(node(L, H, R), List) :- toList(L, L1), toList(R, L2), append(L1, [H|L2], List).
288
289 % del(X, Xs, Ys).
290 % True if Ys is the list Xs with one occurence of X removed
291 %
292 % eg.    del(2, [3, 2, 1, 2, 3], Ys).
293 %       Ys = [3, 1, 2, 3].
294 %       Ys = [3, 2, 1, 3].
295 del(X, [X|Xs], Xs).
296 del(X, [Y|Xs], [Y|Ys]) :- del(X, Xs, Ys).
297
298 % delN(Xs, Num, Zs).
299 % True if Zs is the list Xs with element at place Num (1-indexed) removed
300 %
301 % eg.    delN(['fee', 'foe', 'fum'], 2, ZS).
302 %       ZS = ['fee', 'fum'].
303 delN(XS, 0, XS).
304 delN([], _, []).
305 delN([_X|XS], 1, XS).
306 delN([X|XS], N, [X|YS]) :- N > 1, M is N-1, delN(XS, M, YS).% ivar studies informatics
307 studies(ivar, informatics).
308
309
310 % addAtEnd(Xs, Y, Zs).
311 % True if Zs is the list Xs with Y added to the end
312 %
313 % eg.     addAtEnd([1, 2, 3, 4], 5, Zs).
314 %       Zs = [1, 2, 3, 4, 5].
315 addAtEnd([], Object, [Object]).
316 addAtEnd([H|T], Object, [H|List]) :- addAtEnd(T, Object, List).
317
318 % reverse(Xs, Ys).
319 % True if Ys is Xs reversed
320 %
321 % eg.     reverse([1, 2, 3], Ys).
322 %       Ys = [3, 2, 1].
323 reverse([], []).
324 reverse([X], [X]).
325 reverse([X|Xs], Y) :- reverse(Xs, Z), addAtEnd(Z, X, Y).
326
327 % natural_number(X).
328 % True if X is a natural number (successor of 0)
329 %
330 % eg.     natural_number(0).
331 %       yes
332 %       natural_number(s(s(s(0)))).
333 %       yes
334 %       natural_number(4).
335 %       no
336 natural_number(0).
337 natural_number(s(X)) :- natural_number(X).
338
339 % plus(X, Y, Z).
340 % True if Z is the result of adding X and Y.
341 %
342 % eg.     plus(s(s(0)), s(0), Z).
343 %       Z = s(s(s(0))).
344 plus(0, N, N).
345 plus(s(X), N, s(XPlusN)) :- plus(X, N, XPlusN).
346
347 % mul(X, Y, Z).
348 % True if Z is the result of multiplying X and Y
349 %
```

```
350 % eg.    mul(s(s(s(0))), s(s(s(0))), Z).
351 %        Z = s(s(s(s(s(s(s(s(s(0)))))))))).
352 %        3 * 3 = 9...
353 mul(0, Y, 0).
354 mul(s(X), Y, Z) :- mul(X, Y, XY), plus(XY, Y, Z).
355
356 % greater_than(X, Y).
357 % True if X is greater than Y
358 %
359 % eg.     greater_than(s(s(s(0))), s(s(0))).
360 %        Yes
361 %        greater_than(s(s(0)), s(s(s(0)))).
362 %         no
363 greater_than(s(_), 0).
364 greater_than(s(X), s(Y)) :- greater_than(X, Y).
```