1.

:- use\_module(library(clpfd)).

solve :-

% This is a verbose solution that will be improved upon.

% Label each row as A..F.

Puzzle = [A, B, C, D, E, F ],

% Each row is made of 6 values.

A = [A1, A2, A3, A4, A5, A6 ],

B = [B1, B2, B3, B4, B5, B6 ],

C = [C1, C2, C3, C4, C5, C6 ],

D = [D1, D2, D3, D4, D5, D6 ],

E = [E1, E2, E3, E4, E5, E6 ],

F = [F1, F2, F3, F4, F5, F6 ],

% Each column is also 6 values, using the variables from the rows.

Co1 = [A1, B1, C1, D1, E1, F1 ],

Co2 = [A2, B2, C2, D2, E2, F2 ],

Co3 = [A3, B3, C3, D3, E3, F3 ],

Co4 = [A4, B4, C4, D4, E4, F4 ],

Co5 = [A5, B5, C5, D5, E5, F5 ],

Co6 = [A6, B6, C6, D6, E6, F6 ],

% "ins" forces each element of a list to be in a particular range.

% It is part of "constraint logic programming".

A ins 1..6,

B ins 1..6,

C ins 1..6,

D ins 1..6,

E ins 1..6,

F ins 1..6,

% All items in a row must be different.

all\_different(A),

all\_different(B),

all\_different(C),

all\_different(D),

all\_different(E),

all\_different(F),

% All items in a column must be different.

all\_different(Co1),

all\_different(Co2),

all\_different(Co3),

all\_different(Co4),

all\_different(Co5),

all\_different(Co6),

(A1 + B1 + B2 + B3 + C3 #= 24),

(A2 - A3 #= 1 ; A3 - A2 #= 1),

(A4 // A5 #= 2 ; A5 // A4 #= 2),

(A6 \* B6 \* B5 \* B4 \* C4 #= 72),

(C1 + D1 + D2 #= 14),

(C2 #= 5),

(C5 #= 2),

(C6 \* D6 \* D5 #= 18),

(D3 \* D4 \* E3 \* E4 #= 60),

(E1 \* F1 #= 3),

(E2 + F2 + F3 + F4 + F5 + E5 #= 22),

(E6 - F6 #= 1 ; F6 - E6 #= 1),

label(A),

label(B),

label(C),

label(D),

label(E),

label(F),

write(Puzzle).

OUTPUT:

1 ?- solve.

[[5,3,4,2,1,6],[2,6,5,3,4,1],[4,5,6,1,2,3],[6,4,1,5,3,2],[1,2,3,4,6,5],[3,1,2,6,5,4]]

true .

2A.

animal(grizzlybear).

animal(moose).

animal(seal).

animal(zebra).

place(circus).

place(rockband).

place(spaceship).

place(train).

solve :-

animal(JoanneAnimal), animal(LouAnimal), animal(RalphAnimal), animal(WinnieAnimal),

all\_different([JoanneAnimal, LouAnimal, RalphAnimal, WinnieAnimal]),

place(JoannePlace), place(LouPlace), place(RalphPlace), place(WinniePlace),

all\_different([JoannePlace, LouPlace, RalphPlace, WinniePlace]),

Triples = [ [joanne, JoanneAnimal, JoannePlace],

[lou, LouAnimal, LouPlace],

[ralph, RalphAnimal, RalphPlace],

[winnie, WinnieAnimal, WinniePlace] ],

% 1.

\+ member([joanne, seal, \_],Triples),

\+ member([lou, seal, \_],Triples),

\+ member([\_, seal, spaceship],Triples),

\+ member([\_, seal, train],Triples),

% 2.

\+ member([joanne, grizzlybear, \_],Triples),

member([joanne, \_, circus],Triples),

% 3.

member([winnie, zebra, \_], Triples),

% 4.

\+ member([\_, grizzlybear, spaceship],Triples),

tell(joanne, JoanneAnimal, JoannePlace),

tell(lou, LouAnimal, LouPlace),

tell(ralph, RalphAnimal, RalphPlace),

tell(winnie, WinnieAnimal, WinniePlace).

all\_different([H | T]) :- member(H, T), !, fail.

all\_different([\_ | T]) :- all\_different(T).

all\_different([\_]).

tell(X, Y, Z) :-

write(X), write(' took the '), write(Y),

write(' to the '), write(Z), write('.'), nl.

OUTPUT:

1 ?- solve.

joanne took the moose to the circus.

lou took the grizzlybear to the train.

ralph took the seal to the rockband.

winnie took the zebra to the spaceship.

true .

2B.

day(tuesday).

day(wednesday).

day(thursday).

day(friday).

object(balloon).

object(clothesline).

object(frisbee).

object(watertower).

before(tuesday,wednesday).

before(wednesday,thursday).

before(thursday,friday).

earlier(X,Y):- before(X,Y).

earlier(X,Z) :- before(X,Y), earlier(Y,Z).

solve :-

day(BarradaDay),day(GortDay),day(KlatuDay),day(NiktoDay),

all\_different([BarradaDay, GortDay, KlatuDay, NiktoDay]),

object(BarradaObj),object(GortObj),object(KlatuObj),object(NiktoObj),

all\_different([BarradaObj, GortObj, KlatuObj, NiktoObj]),

Triples = [[barrada, BarradaDay, BarradaObj],

[gort, GortDay, GortObj],

[klatu, KlatuDay, KlatuObj],

[nikto, NiktoDay, NiktoObj]],

% 1.

\+ member([gort, \_, frisbee],Triples),

(member([\_, X, balloon], Triples), earlier(KlatuDay,X)),

(member([\_,Y,frisbee],Triples), earlier(Y,KlatuDay)),

% 2.

(member([\_, friday, clothesline],Triples) ;

member([barrada, friday, \_],Triples) ;

member([barrada, friday, clothesline], Triples)),

% 3.

\+ member([nikto, tuesday, \_],Triples),

% 4.

\+ member([klatu, \_, watertower],Triples),

tell(barrada, BarradaDay, BarradaObj),

tell(gort, GortDay, GortObj),

tell(klatu, KlatuDay, KlatuObj),

tell(nikto, NiktoDay, NiktoObj).

all\_different([H | T]) :- member(H, T), !, fail.

all\_different([\_ | T]) :- all\_different(T).

all\_different([\_]).

tell(X, Y, Z) :-

write(X), write(' saw the '), write(Z),

write(' on '), write(Y), write('.'), nl.

OUTPUT:

1 ?- solve.

barrada saw the balloon on friday.

gort saw the watertower on tuesday.

klatu saw the clothesline on thursday.

nikto saw the frisbee on wednesday.

true .