Jonathan Julian

Hw 1.5

#1.

Program

m([jonathan, jayden, bryan, joe, joseph, jon, natale, braden, jacob, phillip, garret]).

f([mary, samantha, onna, christina, kathy, judy, jodi, makayla, tori]).

family([natale,[joe]]).

family([joe, judy,[mary, joseph, jon]]).

family([mary, [garret, phillip]]).

family([joseph, jodi,[braden,jacob]]).

family([jon, christina,[jonathan, makayla]]).

family([bryan, kathy,[samantha, tori]]).

family([jonathan, samantha,[jayden,onna]]).

member2(X, [X|\_]):- X\=[\_|\_].

member2(X, [\_|T]):-

member2(X, T).

nestedList(Y, [[H|T1]|T]):-

member2(Y, [H|T1]);

nestedList(Y, T).

nestedList(Y, [\_|T]):- nestedList(Y,T).

male(X):- m(Z), member2(X, Z).

female(X):- f(Y), member2(X, Y).

parent(X, Y):- family(W), member2(X, W) , nestedList(Y, W), X\=Y.

father(X, Y):- male(X), parent(X,Y), X\=Y.

mother(X,Y):-female(X), parent(X,Y), X\=Y.

sibling1(X,Y):-parent(Z, Y), parent(Z,X), X\=Y.

sibling2(X,Y):- parent(Z, Y), parent(Z, X), parent(W, Y), parent(W, X), X\=Y, W\=Z.

brother1(X,Y):- male(X), male(Y), sibling1(X,Y).

brother2(X,Y):-male(X), male(Y), sibling2(X,Y).

sister1(X,Y):- female(X), female(Y), sibling1(X,Y).

sister2(X,Y):- female(X), female(Y), sibling2(X,Y).

uncle(X,Y):- male(X), parent(W, Y), sibling1(X,W).

aunt(X,Y):- female(X), parent(W,Y), sibling1(X,W).

cousins(X,Y):- parent(W, X), parent(Z,Y), sibling1(Z,W),X\=Y.

grandchild(X,Y):- parent(X, W), parent(W, Y).

grandson(X,Y):- male(Y), grandchild(X,Y).

granddaughter(X,Y) :- female(Y), grandchild(X,Y).

greatgrandparent(X,Y) :- parent(X, W), grandchild(W, Y).

ancestor(Ancestor, Child) :- parent(Ancestor, Child).

ancestor(Ancestor, Child) :- parent(Ancestor, Child1),ancestor(Child1, Child).

Output:

?- setof(X, male(X), Xlist).

Xlist = [braden, bryan, garret, jacob, jayden, joe, jon, jonathan, joseph|...].

?- setof(X, female(X), Xlist).

Xlist = [christina, jodi, judy, kathy, makayla, mary, onna, samantha, tori].

?- setof([X,Y], father(X,Y), Xlist).

Xlist = [[bryan, samantha], [bryan, tori], [joe, jon], [joe, joseph], [joe, mary], [jon, jonathan], [jon, makayla], [jonathan|...], [...|...]|...].

?- setof([X,Y], mother(X,Y), Xlist).

Xlist = [[christina, jonathan], [christina, makayla], [jodi, braden], [jodi, jacob], [judy, jon], [judy, joseph], [judy, mary], [kathy|...], [...|...]|...].

?- setof([X,Y], parent(X,Y), Xlist).

Xlist = [[bryan, samantha], [bryan, tori], [christina, jonathan], [christina, makayla], [jodi, braden], [jodi, jacob], [joe, jon], [joe|...], [...|...]|...].

?- setof([X,Y], sibling1(X,Y), Xlist).

Xlist = [[braden, jacob], [garret, phillip], [jacob, braden], [jayden, onna], [jon, joseph], [jon, mary], [jonathan, makayla], [joseph|...], [...|...]|...].

?- setof([X,Y], sibling2(X,Y), Xlist).

Xlist = [[braden, jacob], [jacob, braden], [jayden, onna], [jon, joseph], [jon, mary], [jonathan, makayla], [joseph, jon], [joseph|...], [...|...]|...].

?- setof([X,Y], brother1(X,Y), Xlist).

Xlist = [[braden, jacob], [garret, phillip], [jacob, braden], [jon, joseph], [joseph, jon], [phillip, garret]].

?- setof([X,Y], brother2(X,Y), Xlist).

Xlist = [[braden, jacob], [jacob, braden], [jon, joseph], [joseph, jon]].

?- setof([X,Y], sister2(X,Y), Xlist).

Xlist = [[samantha, tori], [tori, samantha]].

?- setof([X,Y], sister1(X,Y), Xlist).

Xlist = [[samantha, tori], [tori, samantha]].

?- setof([X,Y], cousins(X,Y), Xlist).

Xlist = [[braden, garret], [braden, jonathan], [braden, makayla], [braden, phillip], [garret, braden], [garret, jacob], [garret, jonathan], [garret|...], [...|...]|...].

?- setof([X,Y], uncle(X,Y), Xlist).

Xlist = [[jon, braden], [jon, garret], [jon, jacob], [jon, phillip], [joseph, garret], [joseph, jonathan], [joseph, makayla], [joseph|...]].

?- setof([X,Y], aunt(X,Y), Xlist).

Xlist = [[makayla, jayden], [makayla, onna], [mary, braden], [mary, jacob], [mary, jonathan], [mary, makayla], [tori, jayden], [tori|...]].

?- setof([X,Y], grandchild(X,Y), Xlist).

Xlist = [[bryan, jayden], [bryan, onna], [christina, jayden], [christina, onna], [joe, braden], [joe, garret], [joe, jacob], [joe|...], [...|...]|...].

?- setof([X,Y], grandson(X,Y), Xlist).

Xlist = [[bryan, jayden], [christina, jayden], [joe, braden], [joe, garret], [joe, jacob], [joe, jonathan], [joe, phillip], [jon|...], [...|...]|...].

?- setof([X,Y], granddaughter(X,Y), Xlist).

Xlist = [[bryan, onna], [christina, onna], [joe, makayla], [jon, onna], [judy, makayla], [kathy, onna], [natale, mary]].

?- setof([X,Y], greatgrandparent(X,Y), Xlist).

Xlist = [[joe, jayden], [joe, onna], [judy, jayden], [judy, onna], [natale, braden], [natale, garret], [natale, jacob], [natale|...], [...|...]|...].

?- setof([X,Y], ancestor(X,Y), Xlist).

Xlist = [[bryan, jayden], [bryan, onna], [bryan, samantha], [bryan, tori], [christina, jayden], [christina, jonathan], [christina, makayla], [christina|...], [...|...]|...].

#2

Program:

member2(X,[X|\_]).

member2(X, [\_|T]):- member2(X, T).

firstmember(X, [X|\_]).

two(X,Y,[X,Y|\_]).

two(X,Y, [\_|T]):- two(X,Y,T).

three(X,Y,Z,[X,Y,Z|\_]).

three(X,Y,Z,[\_|T]):-three(X,Y,Z,T).

append1([],L2,L2).

append1([X|T1],T2,[X|T3]):-append1(T1, T2, T3).

delete1(X, [X|T], T).

delete1(X, [\_|T], [T|T1]):-delete1(X,T,T1).

append2(X, [],[X]).

append2(X, [H|T],[H|T2] ):- append2(X, T, T2).

insert1(X, L ,XL):- delete1(X, XL, L).

length1([], 0).

length1([\_|T],X):-length1(T,X1), X is X1+1.

reverse1(L,L2):-reverse1(L, L2, []).

reverse1([], L, L).

reverse1([H|T], L, R):- reverse1(T, L, [H|R]).

palindrome(L):- reverse1(L, L2), L2=L.

showlist1([H|T]):- write(H), tab(1), showlist1(T).

Output:

?- findall(X,(member2(X,[])),Results).

Results = [].

?- findall(X,(member2(X,[1,2,3,4])),Results).

Results = [1, 2, 3, 4].

?- findall(X,(firstmember(X,[])),Results).

Results = [].

?- findall(X,(firstmember(X,[1,2,3,4])),Results).

Results = [1].

?- findall([X,Y],(two(X,Y,[])),Results).

Results = [].

?- findall([X,Y],(two(X,Y,[1,2,3,4])),Results).

Results = [[1, 2], [2, 3], [3, 4]].

?- findall([X,Y,Z],(three(X,Y,Z,[])),Results).

Results = [].

?- findall([X,Y,Z],(three(X,Y,Z,[1,2,3,4])),Results).

Results = [[1, 2, 3], [2, 3, 4]].

?- append1([], [1,2,3], L2).

L2 = [1, 2, 3].

?- append1([1,2,3], [1,2,3], L2).

L2 = [1, 2, 3, 1, 2, 3].

?- delete1(1, [], L2).

false.

?- delete1(1, [1,2,3], L2).

L2 = [2, 3] .

?- append2(1, [], L2).

L2 = [1] .

?- append2(1, [1,2,3], L2).

L2 = [1, 2, 3, 1] .

?- insert1(5, [1,2,3,4,5,6,7,8], L2).

L2 = [5, 1, 2, 3, 4, 5, 6, 7, 8] .

?- length1([],X).

X = 0.

?- length1([1,2,3,4],X).

X = 4.

?- reverse1([1,2,3,4],L2).

L2 = [4, 3, 2, 1].

?- palindrome([1,2,3,4]).

false.

?- palindrome([1,2,3,2,1]).

true.

?- showlist1([]).

false.

?- showlist1([1,2,3,4,5]).

1 2 3 4 5

false.

#3

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

%at most there can be 1 queen in each file. (vertical)

%at most there can be 1 queen in each rank. (horizontal)

%abs(Q0-Q) checks the diagonal distance between queens.

n\_queens(N, Qs) :-

length(Qs, N), %creating list of length N

Qs ins 1..N, %labeling Q's from 1 to N (shows each file)

safe\_queens(Qs). %checking safety of queens

safe\_queens([]). %base case of 0 queens

safe\_queens([Q|Qs]) :- safe\_queens(Qs,Q, 1), safe\_queens(Qs).

%checking each queen to be safe.

safe\_queens([],\_,\_). %checks for attack of no list

safe\_queens([Q|Qs], Q0, D0):- %list of queens, checking against other D0 represents diagonal

Q0 #\=Q, %direct check of location being equal

abs(Q0-Q) #\= D0, %checks diagonal to right both up and down

D #= D0 + 1, %increments D for next value.

safe\_queens(Qs, Q0, D). %checks remainder of list

%solution gathered from tutorial found here:

%https://www.youtube.com/watch?v=l\_tbL9RjFdo

%additional comments provided to show understanding of solution

Output:

?- n\_queens(8, Qs), label(Qs).

Qs = [1, 5, 8, 6, 3, 7, 2, 4] ;

Qs = [1, 6, 8, 3, 7, 4, 2, 5] ;

Qs = [1, 7, 4, 6, 8, 2, 5, 3] ;

Qs = [1, 7, 5, 8, 2, 4, 6, 3] ;

Qs = [2, 4, 6, 8, 3, 1, 7, 5] ;

Qs = [2, 5, 7, 1, 3, 8, 6, 4] ;

Qs = [2, 5, 7, 4, 1, 8, 6, 3] ;

Qs = [2, 6, 1, 7, 4, 8, 3, 5] ;

Qs = [2, 6, 8, 3, 1, 4, 7, 5] ;

Qs = [2, 7, 3, 6, 8, 5, 1, 4] ;

Qs = [2, 7, 5, 8, 1, 4, 6, 3] ;

Qs = [2, 8, 6, 1, 3, 5, 7, 4] ;

Qs = [3, 1, 7, 5, 8, 2, 4, 6] ;

Qs = [3, 5, 2, 8, 1, 7, 4, 6] ;

Qs = [3, 5, 2, 8, 6, 4, 7, 1] ;

Qs = [3, 5, 7, 1, 4, 2, 8, 6] ;

Qs = [3, 5, 8, 4, 1, 7, 2, 6] ;

Qs = [3, 6, 2, 5, 8, 1, 7, 4] ;

Qs = [3, 6, 2, 7, 1, 4, 8, 5] ;

Qs = [3, 6, 2, 7, 5, 1, 8, 4] ;

Qs = [3, 6, 4, 1, 8, 5, 7, 2] ;

Qs = [3, 6, 4, 2, 8, 5, 7, 1] ;

Qs = [3, 6, 8, 1, 4, 7, 5, 2] ;

Qs = [3, 6, 8, 1, 5, 7, 2, 4] ;

Qs = [3, 6, 8, 2, 4, 1, 7, 5] ;

Qs = [3, 7, 2, 8, 5, 1, 4, 6] ;

Qs = [3, 7, 2, 8, 6, 4, 1, 5] ;

Qs = [3, 8, 4, 7, 1, 6, 2, 5] ;

Qs = [4, 1, 5, 8, 2, 7, 3, 6] ;

Qs = [4, 1, 5, 8, 6, 3, 7, 2] ;

Qs = [4, 2, 5, 8, 6, 1, 3, 7] ;

Qs = [4, 2, 7, 3, 6, 8, 1, 5] ;

Qs = [4, 2, 7, 3, 6, 8, 5, 1] ;

Qs = [4, 2, 7, 5, 1, 8, 6, 3] ;

Qs = [4, 2, 8, 5, 7, 1, 3, 6] ;

Qs = [4, 2, 8, 6, 1, 3, 5, 7] ;

Qs = [4, 6, 1, 5, 2, 8, 3, 7] ;

Qs = [4, 6, 8, 2, 7, 1, 3, 5] ;

Qs = [4, 6, 8, 3, 1, 7, 5, 2] ;

Qs = [4, 7, 1, 8, 5, 2, 6, 3] ;

Qs = [4, 7, 3, 8, 2, 5, 1, 6] ;

Qs = [4, 7, 5, 2, 6, 1, 3, 8] ;

Qs = [4, 7, 5, 3, 1, 6, 8, 2] ;

Qs = [4, 8, 1, 3, 6, 2, 7, 5] ;

Qs = [4, 8, 1, 5, 7, 2, 6, 3] ;

Qs = [4, 8, 5, 3, 1, 7, 2, 6] ;

Qs = [5, 1, 4, 6, 8, 2, 7, 3] ;

Qs = [5, 1, 8, 4, 2, 7, 3, 6] ;

Qs = [5, 1, 8, 6, 3, 7, 2, 4] ;

Qs = [5, 2, 4, 6, 8, 3, 1, 7] ;

Qs = [5, 2, 4, 7, 3, 8, 6, 1] ;

Qs = [5, 2, 6, 1, 7, 4, 8, 3] ;

Qs = [5, 2, 8, 1, 4, 7, 3, 6] ;

Qs = [5, 3, 1, 6, 8, 2, 4, 7] ;

Qs = [5, 3, 1, 7, 2, 8, 6, 4] ;

Qs = [5, 3, 8, 4, 7, 1, 6, 2] ;

Qs = [5, 7, 1, 3, 8, 6, 4, 2] ;

Qs = [5, 7, 1, 4, 2, 8, 6, 3] ;

Qs = [5, 7, 2, 4, 8, 1, 3, 6] ;

Qs = [5, 7, 2, 6, 3, 1, 4, 8] ;

Qs = [5, 7, 2, 6, 3, 1, 8, 4] ;

Qs = [5, 7, 4, 1, 3, 8, 6, 2] ;

Qs = [5, 8, 4, 1, 3, 6, 2, 7] ;

Qs = [5, 8, 4, 1, 7, 2, 6, 3] ;

Qs = [6, 1, 5, 2, 8, 3, 7, 4] ;

Qs = [6, 2, 7, 1, 3, 5, 8, 4] ;

Qs = [6, 2, 7, 1, 4, 8, 5, 3] ;

Qs = [6, 3, 1, 7, 5, 8, 2, 4] ;

Qs = [6, 3, 1, 8, 4, 2, 7, 5] ;

Qs = [6, 3, 1, 8, 5, 2, 4, 7] ;

Qs = [6, 3, 5, 7, 1, 4, 2, 8] ;

Qs = [6, 3, 5, 8, 1, 4, 2, 7] ;

Qs = [6, 3, 7, 2, 4, 8, 1, 5] ;

Qs = [6, 3, 7, 2, 8, 5, 1, 4] ;

Qs = [6, 3, 7, 4, 1, 8, 2, 5] ;

Qs = [6, 4, 1, 5, 8, 2, 7, 3] ;

Qs = [6, 4, 2, 8, 5, 7, 1, 3] ;

Qs = [6, 4, 7, 1, 3, 5, 2, 8] ;

Qs = [6, 4, 7, 1, 8, 2, 5, 3] ;

Qs = [6, 8, 2, 4, 1, 7, 5, 3] ;

Qs = [7, 1, 3, 8, 6, 4, 2, 5] ;

Qs = [7, 2, 4, 1, 8, 5, 3, 6] ;

Qs = [7, 2, 6, 3, 1, 4, 8, 5] ;

Qs = [7, 3, 1, 6, 8, 5, 2, 4] ;

Qs = [7, 3, 8, 2, 5, 1, 6, 4] ;

Qs = [7, 4, 2, 5, 8, 1, 3, 6] ;

Qs = [7, 4, 2, 8, 6, 1, 3, 5] ;

Qs = [7, 5, 3, 1, 6, 8, 2, 4] ;

Qs = [8, 2, 4, 1, 7, 5, 3, 6] ;

Qs = [8, 2, 5, 3, 1, 7, 4, 6] ;

Qs = [8, 3, 1, 6, 2, 5, 7, 4] ;

Qs = [8, 4, 1, 3, 6, 2, 7, 5] ;

false.