Homework 3: Declarative Programming

CSC 600-01 Programming Languages

Spring 2017

Emilio Quiambao

3 / 15 / 2017

1. Family relationships

SOURCE CODE:

```
m([john, chris, nick, pete, jack, phoenix, ezra, jace]).
f([jane, nora, megan, jenny, jill, may, haley]).

family([john, jane, [chris, nick, nora]]).
family([chris, jenny, [jill]]).
family([pete, may, [jack]]).
family([jack, jill, [ezra, phoenix]]).
family([ezra, may, [haley]]).
family([phoenix, megan, [jace]]).

male(X) :- m(M), member(X, M).

female(X) :- f(F), member(X, F).

father(F, C) :- family([F, _, CHILDREN]), member(C, CHILDREN).
mother(M, C) :- family([_, M, CHILDREN]), member(C, CHILDREN).
parent(P, C) :- father(P, C); mother(P, C).

siblings1(X, Y) :- father(F, X), father(F, Y),
                   mother(M1, X), mother(M2, Y),
                   M1 \= M2, X \= Y.
siblings1(X, Y) :- mother(M, X), mother(M, Y),
                   father(F1, X), father(F2, Y),
                   F1 \= F2, X \= Y.

siblings2(X, Y) :- father(F, X), father(F, Y),
                   mother(M, X), mother(M, Y), X \= Y.
```

```
brother1(X, Y) :- male(X), siblings1(X, Y).
brother2(X, Y) :- male(X), siblings2(X, Y).

sister1(X, Y) :- female(X), siblings1(X, Y).
sister2(X, Y) :- female(X), siblings2(X, Y).

cousins(X, Y) :- parent(P1, X), parent(P2, Y),
                 (siblings1(P1, P2); siblings2(P1, P2)).

uncle(U, N) :- father(F, N), (siblings1(U, F); siblings2(U, F)).
aunt(A, N) :- mother(M, N), (siblings1(A, M); siblings2(A, M)).

grandchild(GC, GP) :- parent(P, GC), parent(GP, P).
grandson(GS, GP) :- grandchild(GS, GP), male(GS).
granddaughter(GD, GP) :- grandchild(GD, GP), female(GD).

greatgrandparent(GGP, GGC) :- grandchild(GGC, GP), parent(GGP, GP).
ancestor(ANC, DES) :- parent(ANC, DES); parent(P, DES), ancestor(ANC, P).
```

CONSOLE:

male(x) example

```
| ?- male(jack).
true ? ;
no
```

female(x) example

```
| ?- female(X).
X = jane ? ;
X = nora ? ;
X = megan ? ;
X = jenny ? ;
X = jill ? ;
X = may ? ;
X = haley
yes
```

father example

```
| ?- father(X, chris).
X = john ? ;
no
```

mother example

```
| ?- mother(X, chris).
X = jane ? ;
no
```

parent example

```
| ?- parent(X,ezra).
X = jack ? ;
X = jill ? ;
(16 ms) no
```

siblings1 example

```
| ?- siblings1(X,Y).
X = jack
Y = haley ? ;
X = haley
Y = jack ? ;
(16 ms) no
```

siblings2 example

```
| ?- siblings2(X,Y).
X = chris
Y = nick ? ;
X = chris
Y = nora ? ;
X = nick
Y = chris ? ;
X = nick
Y = nora ? ;
X = nora
Y = chris ? ;
X = nora
Y = nick ? ;
X = ezra
Y = phoenix ? ;
X = phoenix
Y = ezra ? ;
(31 ms) no
```

brother1 example

```
| ?- brother1(X,Y).
X = jack
Y = haley ? ;
no
```

brother2 example

```
| ?- brother2(X,Y).
X = chris
Y = nick ? ;
X = chris
Y = nora ? ;
X = nick
Y = chris ? ;
X = nick
Y = nora ? ;
X = phoenix
Y = ezra ? ;
X = ezra
Y = phoenix ? ;
(31 ms) no
```

sister1 example

```
| ?- sister1(X,Y).
X = haley
Y = jack ? ;
no
```

sister2 example

```
| ?- sister2(X,Y).
X = nora
Y = chris ? ;
X = nora
Y = nick ? ;
no
```

cousins example

```
| ?- cousins(X,Y).
X = haley
Y = jace ? ;
X = jace
Y = haley ? ;
no
```

uncle example

```
| ?- uncle(X,Y).
X = nick
Y = jill ? ;
X = nora
Y = jill ? ;
X = haley
Y = ezra ? ;
X = haley
Y = phoenix ? ;
X = phoenix
Y = haley ? ;
X = ezra
Y = jace ? ;
(16 ms) no
```

aunt example

```
| ?- aunt(X,Y).
no
```

grandchild example

```
| ?- grandchild(ezra, Y).
Y = pete ? ;
Y = may ? ;
Y = chris ? ;
Y = jenny ? ;
(16 ms) no
```

grandson example

```
| ?- grandson(phoenix, Y).
Y = pete ? ;
Y = may ? ;
Y = chris ? ;
Y = jenny ? ;
no
```

granddaughter example

```
| ?- granddaughter(X,Y).
X = jill
Y = john ? ;
X = jill
Y = jane ? ;
X = haley
Y = jack ? ;
X = haley
Y = jill ? ;
(16 ms) no
```

greatgrandparent example

```
| ?- greatgrandparent(X, haley).
X = pete ? ;
X = may ? ;
X = chris ? ;
X = jenny ? ;
(31 ms) no
```

ancestor example

```
| ?- ancestor(X, phoenix).
X = jack ? ;
X = jill ? ;
X = pete ? ;
X = may ? ;
X = chris ? ;
X = jenny ? ;
X = john ? ;
X = jane ? ;
no
```

2. Operations

SOURCE CODE:

```
member_test(E, [H | T]) :- E = H; member_test(E, T).

first_element(E, [H | _]) :- E = H.

last_element(E, [E]).
last_element(E, [_ | T]) :-  last_element(E, T).

two_adj_elements(E1, E2, [E1, E2 | T]).
two_adj_elements(E1,E2, [_ | T]) :- two_adj_elements(E1, E2, T).

three_adj_elements(E1, E2, E3, [E1, E2, E3 | T]).
three_adj_elements(E1, E2, E3, [_ | T]) :- three_adj_elements(E1, E2, E3, T).

myappend([], L, L).
myappend([H1|T1], L2, [H1|T3]) :- myappend(T1, L2, T3).

del_element(E, [E | T], T).
del_element(E, [H | T], [H | T1]) :- del_element(E, T, T1).

append_element(E, L, L2) :- myappend(L, [E], L2).

insert_element(E, L, L2) :- del_element(E, L2, L).

mylength([], 0).
mylength([_ | T], N) :- mylength(T, N1), N is N1+1.

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

```
palindrome(L) :- myreverse(L, L1), L1 = L.


mydisplay([H|T]) :- write(H), write(', '), mydisplay(T).
mydisplay([]).
```

CONSOLE:

membership testing example

```
| ?- member_test(5, [1,2,3,4,5,6,7,8,5,5]).
true ? ;
true ? ;
true ? ;
no
```

first element example

```
| ?- first_element(X,[5,8,9]).
X = 5
yes
```

last element example

```
| ?- last_element(X,[5,8,9]).
X = 9 ? ;
no
```

two adjacent elements example

```
| ?- two_adj_elements(5,6,[1,5,6,5,6,3]).
true ? ;
true ? ;
no
```

three adjacent elements example

```
| ?- three_adj_elements(7,8,9,[6,7,8,9,10]).
true ? ;
no
```

append example

```
| ?- myappend([3,4,5],[6,7,8],X).
X = [3,4,5,6,7,8]
yes
```

delete element example

```
| ?- del_element(z,[1,z,2,3,z],X).
X = [1,2,3,z] ? ;
X = [1,z,2,3] ? ;
no
```

append element example

```
| ?- append_element(4,[2,3],X).
X = [2,3,4]
yes
```

insert element example

```
| ?- insert_element(a,[1,1,1],X).
X = [a,1,1,1] ? ;
X = [1,a,1,1] ? ;
X = [1,1,a,1] ? ;
X = [1,1,1,a] ? ;
no
```

length example

```
| ?- mylength([1,2,3,4,5],X).
X = 5
yes
```

reverse example

```
| ?- myreverse([a,b,c,1,2,3],X).
X = [3,2,1,c,b,a]
yes
```

palindrome example

```
| ?- palindrome([r,a,c,e,c,a,r]).
yes
| ?- palindrome([h,e,l,l,o]).
no
```

display example

```
| ?- mydisplay([hello, world, 1, 2, 3]).
hello, world, 1, 2, 3,
yes
```

3. 8 queens problem

SOURCE CODE:

```
queens([]).
queens([row/col | others]) :- queens(others),
                      member(col,[1,2,3,4,5,6,7,8]),
                      clear(row/col, others).

clear(_, []).
clear(row/col, [row1/col1 | others]) :- Col =\= Col1,
                                (Col1 - Col) =\= (Row1 - Row),
                                (Col1 - Col) =\= (Row - Row1),
                                clear(row/col, others).

member(obj, [obj | tail]).
member(obj, [head | tail]) :- member(obj, tail).

board([1/C1, 2/C2, 3/C3, 4/C4, 5/C5, 6/C6, 7/C7, 8/C8]).
```