CSCI-722 - Data Analytics Cognitive Comp

Name: Rajkumar Pillai

Note: The code for the solutions are present in tree.pl and dcg_example.pl file.

Question-1

Solution:

The rules:

```
%!  %%%%%%%%%% offspring
offspring(A,B) :- parent(B,A).


%!  %%%%%%%%%% grandparents
grandparents(C,A):-
    parent(C,B), parent(B,A).


%!  %%%%%%%%%% ancestors
ancestor(A,C):-
    offspring(C,A);(offspring(B,A),ancestor(B,C)).


%!  %%%%%%%%%%%%%%% sibling
sibling(A,B):-
   offspring(A,C) ,offspring(B,C) ,A \= B.


%!  %%%%%%%%%% uncle
uncle(A,B) :-
          offspring(B,C), sibling(C,D), male(A),uncle(A,D).

uncle(A,B) :-
          ancestor(D,B), sibling(D,A), male(A).


%!  %%%%%%%%%% aunt
aunt(A,B) :-
          offspring(B,C), sibling(C,D), female(A),aunt(A,D).

aunt(A,B) :-
          ancestor(D,B), sibling(D,A), female(A).


%!  %%%%%%%%%%%%%% cousin
cousin(A,B) :-
    ancestor(C,A) , ancestor(D,B) , C\=D, sibling(C,D).

cousin(A,B):-
    offspring(A,C), offspring(B,D), C\=D, cousin(C,D).
```

a.) **Who are my parents?**
Ans: margaretMoore, richardFranklin

b.) **Who are my grandparents?**
Ans: blancheAdams, oscarFranklin

c.) **Who are my ancestors?**
Ans: abnerMead, amosAdams, annaHowe, anneDudley, anthonyFranklin, arthurAylesworth, averyWhite, betsyGriggs, blancheAdams, calebHazen, catherineNeville, ceciliaWilloughby, cicelyGray, davidWhite, dencyPhillips, dimmisAylesworth, dorothyYorke, eliWhite, elishaFranklin, elizabethMorse, ellenWhite, esekFranklin, gov_simonBradstreet, hannahHeald, henryAdams, hoseaPhillips1, hoseaPhillips2, jamesFranklin1, jamesFranklin2, janeStrangeways, jesseTwining, joanBeaufort, joannaMunroe, johnAdams1, johnAdams2, johnBradstreet, johnFranklin, johnMunroe, johnPhillips, josephAdams, joshuaWhite, katherineTwining, leviAdams, llewellynFranklin, ltJonasMunroe, lydiaJohnson, margaretMoore, maryDoak, maryFranklin, maryGoodwin, mercyBradstreet, mercyHazen, miltonAdams, oscarFranklin, oscaraFranklin, ralphdeNeville, rebeccaCutter, richardFranklin, sarahMead, sir_edwardSutton, sir_henryDudley, sir_johnSutton, sir_rogerDudley, sir_thomasDudley, sir_williamWilloughby, susannahThorne, thomasAdams, thomasStrangeways, unknownWifeOfsir_henryDudley, williamMunroe .

d.) **Who are my uncles?**
Ans: charlesFranklin, delvinFranklin, edwinFranklin, haroldFranklin, jamesFranklin2, llewellynoscarFranklin, williamFranklin

e.) **Who are my aunts?**
Ans: cecilyNeville, doraWhite, dorisFranklin, dorothyAdams, edithAdams, ednaAdams, ellenFranklin, ireneAdams, jeanFranklin, maryFranklin, sadieAdams, velmaAdams, violaWhite

f.) **Who is my cousin?**
Ans: alfredWhiteson, anthonyFranklin, bessieWhite, blancheAdams, carolRomanowski, charlesFranklin, delvinFranklin, dencyPhillips, dimmisAylesworth, doraWhite, dorisFranklin, dorothyAdams, edithAdams, ednaAdams, edwinFranklin, elishaFranklin, ellenFranklin, ellenWhite, esekFranklin, haroldFranklin, hoseaPhillips1, hoseaPhillips2, ireneAdams, jeanFranklin, johnFranklin, johnPhillips, king_richardIII, llewellynFranklin, llewellynoscarFranklin, oscarFranklin, oscaraFranklin, richardFranklin, sadieAdams, velmaAdams, violaWhite, williamFranklin

Prolog Output:

```
[2] ?- setof(X,parent(X,carolRomanowski),List).
List = [margaretMoore, richardFranklin].

[2] ?- setof(X,grandparents(X,carolRomanowski),List).
List = [blancheAdams, oscarFranklin].

[2] ?- setof(X,ancestor(X,carolRomanowski),List);true.
List = [abnerMead, amosAdams, annaHowe, anneDudley, anthonyFranklin, arthurAylesworth, averyWhite, betsyGriggs, blancheAdams, calebHazen, catherineNeville, ceciliaWilloughby, cicelyGray, davidWhite, dencyPhillip
immisAylesworth, dorothyYorke, eliWhite, elishaFranklin, elizabethMorse, ellenWhite, esekFranklin, gov_simonBradstreet, hannahHeald, henryAdams, hoseaPhillips1, hoseaPhillips2, jamesFranklin1, jamesFranklin2, jane
rangeways, jesseTwining, joanBeaufort, joannaMunroe, johnAdams1, johnAdams2, johnBradstreet, johnFranklin, johnMunroe, johnPhillips, josephAdams, joshuaWhite, katherineTwining, leviAdams, llewellynFranklin, ltJon
unroe, lydiaJohnson, margaretMoore, maryDoak, maryFranklin, maryGoodwin, mercyBradstreet, mercyHazen, miltonAdams, oscarFranklin, oscaraFranklin, ralphdeNeville, rebeccaCutter, richardFranklin, sarahMead, sir_
Sutton, sir_henryDudley, sir_johnSutton, sir_rogerDudley, sir_thomasDudley, sir_williamWilloughby, susannahThorne, thomasAdams, thomasStrangeways, unknownWifeOfsir_henryDudley, williamMunroe] .

[2] ?- setof(X,uncle(X,carolRomanowski),List).
List = [charlesFranklin, delvinFranklin, edwinFranklin, haroldFranklin, jamesFranklin2, llewellynoscarFranklin, williamFranklin].

[2] ?- setof(X,aunt(X,carolRomanowski),List);true.
List = [cecilyNeville, doraWhite, dorisFranklin, dorothyAdams, edithAdams, ednaAdams, ellenFranklin, ireneAdams, jeanFranklin, maryFranklin, sadieAdams, velmaAdams, violaWhite] .

[2] ?- setof(X,cousin(X,carolRomanowski),List);true.
List = [alfredWhiteson, anthonyFranklin, bessieWhite, blancheAdams, carolRomanowski, charlesFranklin, delvinFranklin, dencyPhillips, dimmisAylesworth, doraWhite, dorisFranklin, dorothyAdams, edithAdams, ednaAdam
dwinFranklin, elishaFranklin, ellenFranklin, ellenWhite, esekFranklin, haroldFranklin, hoseaPhillips1, hoseaPhillips2, ireneAdams, jeanFranklin, johnFranklin, johnPhillips, king_richardIII, llewellynFranklin, llewel
lynoscarFranklin, oscarFranklin, oscaraFranklin, richardFranklin, sadieAdams, velmaAdams, violaWhite, williamFranklin] .
```

Question-2

Solution:

a.) Define Prolog rules to list both members of an ancestor couple; for example, return my mother's and father's names in a single query.

Ans:

```
%!  %%%%%%%%%%%%%%% ancestor_pair
ancestorpair(A,B) :-
                setof(A, offspring(B,A), A);(offspring(B,C), ancestorpair(A,C)).
```

[[abnerMead, mercyHazen], [amosAdams, betsyGriggs], [annaHowe, johnAdams1], [anneDudley, gov_simonBradstreet], [anthonyFranklin], [arthurAylesworth, maryFranklin], [averyWhite, dencyPhillips], [blancheAdams, oscarFranklin], [calebHazen, mercyBradstreet], [catherineNeville, thomasStrangeways], [ceciliaWilloughby, sir_edwardSutton], [cicelyGray, sir_johnSutton], [davidWhite, sarahMead], [dimmisAylesworth], [dorothyYorke, sir_thomasDudley], [eliWhite, lydiaJohnson], [elishaFranklin], [elizabethMorse, joshuaWhite], [ellenWhite, miltonAdams], [esekFranklin], [hannahHeald, leviAdams], [henryAdams], [hoseaPhillips1], [hoseaPhillips2], [jamesFranklin1], [jamesFranklin2], [janeStrangeways, sir_williamWilloughby], [jesseTwining, maryGoodwin], [joanBeaufort, ralphdeNeville], [joannaMunroe, johnAdams2], [johnBradstreet], [johnFranklin], [johnMunroe], [johnPhillips], [josephAdams, rebeccaCutter], [katherineTwining, llewellynFranklin], [ltJonasMunroe], [margaretMoore, richardFranklin], [maryDoak, oscaraFranklin], [sir_henryDudley, unknownWifeOfsir_henryDudley], [sir_rogerDudley, susannahThorne], [thomasAdams], [williamMunroe]]

Prolog Output:

[2] ?- setof(X,ancestorpair(X,carolRomanowski),List),true.
List = [[abnerMead, mercyHazen], [amosAdams, betsyGriggs], [annaHowe, johnAdams1], [anneDudley, gov_simonBradstreet], [anthonyFranklin], [arthurAylesworth, maryFranklin], [averyWhite, dencyPhillips], [blancheAdar
oscarFranklin], [calebHazen, mercyBradstreet], [catherineNeville, thomasStrangeways], [ceciliaWilloughby, sir_edwardSutton], [cicelyGray, sir_johnSutton], [davidWhite, sarahMead], [dimmisAylesworth], [dorothyYorke,
sir_thomasDudley], [eliWhite, lydiaJohnson], [elishaFranklin], [elizabethMorse, joshuaWhite], [ellenWhite, miltonAdams], [esekFranklin], [hannahHeald, leviAdams], [henryAdams], [hoseaPhillips1], [hoseaPhillips2], [j
amesFranklin1], [jamesFranklin2], [janeStrangeways, sir_williamWilloughby], [jesseTwining, maryGoodwin], [joanBeaufort, ralphdeNeville], [joannaMunroe, johnAdams2], [johnBradstreet], [johnFranklin], [johnMunroe], [j
ohnPhillips], [josephAdams, rebeccaCutter], [katherineTwining, llewellynFranklin], [ltJonasMunroe], [margaretMoore, richardFranklin], [maryDoak, oscaraFranklin], [sir_henryDudley, unknownWifeOfsir_henryDudley], [sir
_rogerDudley, susannahThorne], [thomasAdams], [williamMunroe]]

b.) Find one other relationship besides the ones in #1.

Ans:

```
%!   %%%%%%%%%%%%%%%%%%%%%% son
son(A,B):-male(A),offspring(A,B).
```

Prolog Output:

```
?- setof(X,son(X,oscarFranklin),List).
List = [haroldFranklin, llewellynoscarFranklin, richardFranklin].
```

Sons of oscarFranklin: haroldFranklin, llewellynoscarFranklin, richardFranklin

Question-3

Solution:

a.) Define a set of DCGs and a separate lexicon to parse the following sentences:–

1. Large crowds listened to the orchestra.

2. the man rejected elective surgery.

3. An airplane flew over the city.

Ans:

```
% DCG for sentence: Large crowds listened to the orchestra.
% the man rejected elective surgery.
% An airplane flew over the city


adjective_1 --> [Large].
noun_subject_1 --> [crowds].
verb_1 --> [listened].
pp --> [to].
det_1 --> [the].
noun_object_2  --> [orchestra].
noun_subject_2 --> [man].
verb_2 --> [rejected].
adjective_2 --> [elective].
noun_object_1 --> [surgery].
noun_subject_3 --> [airplane].
det_2 --> [An].
verb_3 --> [flew].
noun_object_3 --> [city].
p --> [over].

s --> noun_phrase_subject, vp.
noun_phrase_subject --> adjective_1, noun_subject_1.
noun_phrase_subject --> det_1, noun_subject_2.
noun_phrase_subject --> det_2, noun_subject_3.
vp --> verb_1, noun_object_1.
vp --> verb_2, noun_object_2.
vp --> verb_3, noun_object_3.
noun_object_1 --> pp, noun_phrase.
noun_object_2 --> adjective_2, noun_object_1.
noun_object_3 --> p, np_obj_other.
noun_phrase --> det_1, noun_object_2.
np_obj_other --> det_1, noun_object_3.
```

Prolog Trace Output:

1. Tracing of sentence: Large crowds listened to the orchestra.

```
?- trace.
true.

[trace]  ?- s([Large,crowds,listened,to,the,orchestra],[]).
   Call: (8) s([_2612, crowds, listened, to, the, orchestra], []) ? creep
   Call: (9) noun_phrase_subject([_2612, crowds, listened, to, the, orchestra], _2912) ? creep
   Call: (10) adjective_1([_2612, crowds, listened, to, the, orchestra], _2912) ? creep
   Exit: (10) adjective_1([_2612, crowds, listened, to, the, orchestra], [crowds, listened, to, the, orchestra]) ? creep
   Call: (10) noun_subject_1([crowds, listened, to, the, orchestra], _2912) ? creep
   Exit: (10) noun_subject_1([crowds, listened, to, the, orchestra], [listened, to, the, orchestra]) ? creep
   Exit: (9) noun_phrase_subject([_2612, crowds, listened, to, the, orchestra], [listened, to, the, orchestra]) ? creep
   Call: (9) vp([listened, to, the, orchestra], []) ? creep
   Call: (10) verb_1([listened, to, the, orchestra], _2912) ? creep
   Exit: (10) verb_1([listened, to, the, orchestra], [to, the, orchestra]) ? creep
   Call: (10) noun_object_1([to, the, orchestra], []) ? creep
   Call: (11) pp([to, the, orchestra], _2912) ? creep
   Exit: (11) pp([to, the, orchestra], [the, orchestra]) ? creep
   Call: (11) noun_phrase([the, orchestra], []) ? creep
   Call: (12) det([the, orchestra], _2912) ? creep
   Exit: (12) det([the, orchestra], [orchestra]) ? creep
   Call: (12) noun_object_2([orchestra], []) ? creep
   Exit: (12) noun_object_2([orchestra], []) ? creep
   Exit: (11) noun_phrase([the, orchestra], []) ? creep
   Exit: (10) noun_object_1([to, the, orchestra], []) ? creep
   Exit: (9) vp([listened, to, the, orchestra], []) ? creep
   Exit: (8) s([_2612, crowds, listened, to, the, orchestra], []) ? creep
true .
```

2. Tracing of sentence: the man rejected elective surgery.

```
[trace] [2]  ?- s([the,man,rejected,elective,surgery],[]).
   Call: (24) s([the, man, rejected, elective, surgery], []) ? creep
   Call: (25) noun_phrase_subject([the, man, rejected, elective, surgery], _4386) ? creep
   Call: (26) adjective_1([the, man, rejected, elective, surgery], _4386) ? creep
   Exit: (26) adjective_1([the, man, rejected, elective, surgery], [man, rejected, elective, surgery]) ? creep
   Call: (26) noun_subject_1([man, rejected, elective, surgery], _4386) ? creep
   Fail: (26) noun_subject_1([man, rejected, elective, surgery], _4386) ? creep
   Redo: (25) noun_phrase_subject([the, man, rejected, elective, surgery], _4386) ? creep
   Call: (26) det([the, man, rejected, elective, surgery], _4386) ? creep
   Exit: (26) det([the, man, rejected, elective, surgery], [man, rejected, elective, surgery]) ? creep
   Call: (26) noun_subject_2([man, rejected, elective, surgery], _4386) ? creep
   Exit: (26) noun_subject_2([man, rejected, elective, surgery], [rejected, elective, surgery]) ? creep
   Exit: (25) noun_phrase_subject([the, man, rejected, elective, surgery], [rejected, elective, surgery]) ? creep
   Call: (25) vp([rejected, elective, surgery], []) ? creep
   Call: (26) verb_1([rejected, elective, surgery], _4386) ? creep
   Fail: (26) verb_1([rejected, elective, surgery], _4386) ? creep
   Redo: (25) vp([rejected, elective, surgery], []) ? creep
   Call: (26) verb_2([rejected, elective, surgery], _4386) ? creep
   Exit: (26) verb_2([rejected, elective, surgery], [elective, surgery]) ? creep
   Call: (26) noun_object_2([elective, surgery], []) ? creep
   Call: (27) adjective_2([elective, surgery], _4386) ? creep
   Exit: (27) adjective_2([elective, surgery], [surgery]) ? creep
   Call: (27) noun_object_1([surgery], []) ? creep
   Exit: (27) noun_object_1([surgery], []) ? creep
   Exit: (26) noun_object_2([elective, surgery], []) ? creep
   Exit: (25) vp([rejected, elective, surgery], []) ? creep
   Exit: (24) s([the, man, rejected, elective, surgery], []) ? creep
true .
```

3. Tracing of sentence: An airplane flew over the city.

```
[trace]  ?- s([An,airplane,flew,over,the,city],[]).
   Call: (8) s([_2612, airplane, flew, over, the, city], []) ? creep
   Call: (9) noun_phrase_subject([_2612, airplane, flew, over, the, city], _2910) ? creep
   Call: (10) adjective_1([_2612, airplane, flew, over, the, city], _2910) ? creep
   Exit: (10) adjective_1([_2612, airplane, flew, over, the, city], [airplane, flew, over, the, city]) ? creep
   Call: (10) noun_subject_1([airplane, flew, over, the, city], _2910) ? creep
   Fail: (10) noun_subject_1([airplane, flew, over, the, city], _2910) ? creep
   Redo: (9) noun_phrase_subject([_2612, airplane, flew, over, the, city], _2910) ? creep
   Call: (10) det_1([_2612, airplane, flew, over, the, city], _2910) ? creep
   Exit: (10) det_1([the, airplane, flew, over, the, city], [airplane, flew, over, the, city]) ? creep
   Call: (10) noun_subject_2([airplane, flew, over, the, city], _2910) ? creep
   Fail: (10) noun_subject_2([airplane, flew, over, the, city], _2910) ? creep
   Redo: (9) noun_phrase_subject([_2612, airplane, flew, over, the, city], _2910) ? creep
   Call: (10) det_2([_2612, airplane, flew, over, the, city], _2910) ? creep
   Exit: (10) det_2([_2612, airplane, flew, over, the, city], [airplane, flew, over, the, city]) ? creep
   Call: (10) noun_subject_3([airplane, flew, over, the, city], _2910) ? creep
   Exit: (10) noun_subject_3([airplane, flew, over, the, city], [flew, over, the, city]) ? creep
   Exit: (9) noun_phrase_subject([_2612, airplane, flew, over, the, city], [flew, over, the, city]) ? creep
   Call: (9) vp([flew, over, the, city], []) ? creep
   Call: (10) verb_1([flew, over, the, city], _2910) ? creep
   Fail: (10) verb_1([flew, over, the, city], _2910) ? creep
   Redo: (9) vp([flew, over, the, city], []) ? creep
   Call: (10) verb_2([flew, over, the, city], _2910) ? creep
   Fail: (10) verb_2([flew, over, the, city], _2910) ? creep
   Redo: (9) vp([flew, over, the, city], []) ? creep
   Call: (10) verb_3([flew, over, the, city], _2910) ? creep
   Exit: (10) verb_3([flew, over, the, city], [over, the, city]) ? creep
   Call: (10) noun_object_3([over, the, city], []) ? creep
   Call: (11) p([over, the, city], _2910) ? creep
   Exit: (11) p([over, the, city], [the, city]) ? creep
   Call: (11) np_obj_other([the, city], []) ? creep
   Call: (12) det_1([the, city], _2910) ? creep
   Exit: (12) det_1([the, city], [city]) ? creep
   Call: (12) noun_object_3([city], []) ? creep
   Exit: (12) noun_object_3([city], []) ? creep
   Exit: (11) np_obj_other([the, city], []) ? creep
   Exit: (10) noun_object_3([over, the, city], []) ? creep
   Exit: (9) vp([flew, over, the, city], []) ? creep
   Exit: (8) s([_2612, airplane, flew, over, the, city], []) ? creep
true .
```