Temitope Emokpae 4/16/2023

CSC 344

Prolog Programming Assignment #2: A Favorite Pokemon KB plus Simple List Processing Exercises

Learning Abstract

Task 1 involves establishing and interacting with a Pokemon knowledge base, and then extending the KB in a number of ways and interrogating the extended KB. Task 2 affords you an opportunity to engage in a variety of list processing exercises.

Task 1 - Pokemon

Part 1: Initial Pokemon KB

```
% ------------------------------------------------------------------------
% ------------------------------------------------------------------------
% --- File: pokemon.pro
% --- Line: Loosely represented Pokemon
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% --- cen(P) :: Pokemon P was "creatio ex nihilo"

cen(pikachu).
cen(bulbasaur).
cen(caterpie).
cen(charmander).
cen(vulpix).
cen(poliwag).
cen(squirtle).
cen(staryu).

% ------------------------------------------------------------------------
% --- evolves(P,Q) :: Pokemon P directly evolves to pokemon Q

evolves(pikachu,raichu).
evolves(bulbasaur,ivysaur).
evolves(ivysaur,venusaur).
evolves(caterpie,metapod).
evolves(metapod,butterfree).
evolves(charmander,charmeleon).
evolves(charmeleon,charizard).
evolves(vulpix,ninetails).
evolves(poliwag,poliwhirl).
evolves(poliwhirl,poliwrath).
```

```
evolves(squirtle,wartortle).
evolves(wartortle,blastoise).
evolves(staryu,starmie).

% ------------------------------------------------------------------------
% --- pokemon(name(N),T,hp(H),attach(A,D)) :: There is a pokemon with
% --- name N, type T, hit point value H, and attach named A that does
% --- damage D.

pokemon(name(pikachu), electric, hp(60), attack(gnaw, 10)).
pokemon(name(raichu), electric, hp(90), attack(thunder-shock, 90)).

pokemon(name(bulbasaur), grass, hp(40), attack(leech-seed, 20)).
pokemon(name(ivysaur), grass, hp(60), attack(vine-whip, 30)).
pokemon(name(venusaur), grass, hp(140), attack(poison-powder, 70)).

pokemon(name(caterpie), grass, hp(50), attack(gnaw, 20)).
pokemon(name(metapod), grass, hp(70), attack(stun-spore, 20)).
pokemon(name(butterfree), grass, hp(130), attack(whirlwind, 80)).

pokemon(name(charmander), fire, hp(50), attack(scratch, 10)).
pokemon(name(charmeleon), fire, hp(80), attack(slash, 50)).
pokemon(name(charizard), fire, hp(170), attack(royal-blaze, 100)).

pokemon(name(vulpix), fire, hp(60), attack(confuse-ray, 20)).
pokemon(name(ninetails), fire, hp(100), attack(fire-blast, 120)).

pokemon(name(poliwag), water, hp(60), attack(water-gun, 30)).
pokemon(name(poliwhirl), water, hp(80), attack(amnesia, 30)).
pokemon(name(poliwrath), water, hp(140), attack(dashing-punch, 50)).

pokemon(name(squirtle), water, hp(40), attack(bubble, 10)).
```

```
pokemon(name(wartortle), water, hp(80), attack(waterfall, 60)).
pokemon(name(blastoise), water, hp(140), attack(hydro-pump, 60)).

pokemon(name(staryu), water, hp(40), attack(slap, 20)).
pokemon(name(starmie), water, hp(60), attack(star-freeze, 20)).

```

Part 2: Interaction demo with the Initial KB

```
?- cen(pikachu).
true.

?- cen(raichu).
false.

?- cen(Name).
Name = pikachu ;
Name = bulbasaur ;
Name = caterpie ;
Name = charmander ;
Name = vulpix ;
Name = poliwag ;
Name = squirtle ;
Name = staryu.
```

```
?- cen(Name),write(Name),nl,fail.
pikachu
bulbasaur
caterpie
charmander
vulpix
poliwag
squirtle
staryu
false.

?- evolves(squirtle,wartortle).
true.

?- evolves(wartortle,squirtle).
false.

?- evolves(squirtle,blastoise).
false.

?- evolves(X,Y),evolves(Y,Z).
X = bulbasaur,
Y = ivysaur,
Z = venusaur ;
X = caterpie,
Y = metapod,
Z = butterfree ;
X = charmander,
Y = charmeleon,
Z = charizard ;
X = poliwag,
Y = poliwhirl,
Z = poliwrath ;
X = squirtle,
Y = wartortle,
Z = blastoise ;
false.

?- evolves(X,Y), evolves(Y,Z), write(X), write( '-->'), write(Z),nl,fail.
bulbasaur-->venusaur
caterpie-->butterfree
charmander-->charizard
poliwag-->poliwrath
squirtle-->blastoise
false.

?- pokemon(name(N),_,_,_),write(N),nl,fail.
pikachu
raichu
bulbasaur
ivysaur
venusaur
caterpie
metapod
butterfree
charmander
charmeleon
charizard
vulpix
ninetails
poliwag
poliwhirl
poliwrath
squirtle
wartortle
blastoise
staryu
starmie
false.

?- pokemon(name(N),fire,_,_),write(N),nl,fail.
charmander
charmeleon
charizard
vulpix
ninetails
false.
```

```
?- pokemon(N,Element,_,_),write(nks(N,kind(Element))),nl,fail.
nks(name(pikachu),kind(electric))
nks(name(raichu),kind(electric))
nks(name(bulbasaur),kind(grass))
nks(name(ivysaur),kind(grass))
nks(name(venusaur),kind(grass))
nks(name(caterpie),kind(grass))
nks(name(metapod),kind(grass))
nks(name(butterfree),kind(grass))
nks(name(charmander),kind(fire))
nks(name(charmeleon),kind(fire))
nks(name(charizard),kind(fire))
nks(name(vulpix),kind(fire))
nks(name(ninetails),kind(fire))
nks(name(poliwag),kind(water))
nks(name(poliwhirl),kind(water))
nks(name(poliwrath),kind(water))
nks(name(squirtle),kind(water))
nks(name(wartortle),kind(water))
nks(name(blastoise),kind(water))
nks(name(staryu),kind(water))
nks(name(starmie),kind(water))
false.

?- pokemon(name(N),_,_,attack(waterfall,_)).
N = wartortle ;
false.

?- pokemon(name(N),_,_,attack(poison-powder,_)).
N = venusaur ;
false.

?- pokemon(_,water,_,attack(Ok,_)),write(Ok),nl,fail.
water-gun
amnesia
dashing-punch
bubble
waterfall
hydro-pump
slap
star-freeze
false.

?- pokemon(name(poliwhirl),_,hp(HP),_).
HP = 80.

?- pokemon(name(butterfree),_,hp(HP),_).
HP = 130.

?- pokemon(name(N),_,hp(HP),_),HP>85,write(N),nl,false.
raichu
venusaur
butterfree
charizard
ninetails
poliwrath
blastoise
false.

?- pokemon(name(N),_,_,attack(_,A)),A>60,write(N),nl,false.
raichu
venusaur
butterfree
charizard
ninetails
false.

?- pokemon(name(N),_,hp(HP),_),cen(N),write(N),write(: ),write(HP),nl,false.
pikachu:60
bulbasaur:40
caterpie:50
charmander:50
vulpix:60
poliwag:60
squirtle:40
staryu:40
false.

?-
```

Part 3: KB Extension

```
% ------------------------------------------------------------------------
% --- KB Extension
% ------------------------------------------------------------------------
%Define a parameterless predicate called display cen to display the names of all of the \creatio ex nihilo" pokemon.
display_cen :- cen(Name),write(Name),nl,fail.

% ------------------------------------------------------------------------
%Define a parameterless predicate called display not cen to display the names of all of the pokemon who are not \creatio
display_not_cen :- evolves(_,S),write(S),nl,fail.

% ------------------------------------------------------------------------
%Define a predicate called generator taking two parameters, the name of a pokemon and the type of a pokemon, which returr
generator(N,T) :- pokemon(name(N),T,_,_).

% ------------------------------------------------------------------------
%Define the parameterless predicate called display names to list the names of all of the pokemon represented in the KB.
display_names :- pokemon(name(N),_,_,_),write(N),nl,fail.

% ------------------------------------------------------------------------
%Define the parameterless predicate called display attacks to list the names of all of the attacks that the pokemon repre
display_attacks :- pokemon(_,_,_,attack(A,_)),write(A),nl,fail.

% ------------------------------------------------------------------------
%Define the parameterless predicate called display cen attacks to list the names of all of the attacks that just
%the creation ex nihilo pokemon represented in the KB can unleash.
display_cen_attacks :- pokemon(name(N),_,_,attack(A,_)), cen(N),write(A),nl,fail.

% ------------------------------------------------------------------------
%Define a predicate called indicate attack taking one parameter, the name of a pokemon, which displays, for
%the named pokemon, a short text of the form: NAME {> ATTACK.
indicate_attack(N) :- pokemon(name(N),_,_,attack(A,_)), write(N), write("-->"),write(A),nl,fail.

% ------------------------------------------------------------------------
%Define a parameterless predicate called indicate attacks which displays a short text of the form NAME {>
%ATTACK for each pokemon in the KB, one short text per line.
indicate_attacks :- pokemon(name(N),_,_,attack(A,_)), write(N), write("-->"),write(A),nl,fail.

% ------------------------------------------------------------------------
%Define a predicate called powerful taking one parameter, the name of a pokemon, which succeeds only if the
%attack associated with the named pokemon yields with more than 55 units of damage.
powerful(N) :- pokemon(name(N),_,_,attack(_,D)), D > 55.

% ------------------------------------------------------------------------
%Define a predicate called tough taking one parameter, the name of a pokemon, which succeeds only if the the
%named pokemon can absorb at least 100 units of damage (that is, has an hp count that is more than 100).
tough(N) :- pokemon(name(N),_,hp(H),_), H > 100.

% ------------------------------------------------------------------------
%Define a predicate called awesome taking one parameter, the name of a pokemon, which succeeds only if the
%the named pokemon is both powerful and tough
awesome(N) :- pokemon(name(N),_,hp(H),attack(_,D)), D > 55, H > 100.

% ------------------------------------------------------------------------
%Define a predicate called powerful but not vulnerable taking one parameter, the name of a pokemon, which
%succeeds only if the the named pokemon is powerful and not tough.
powerful_but_vulnerable(N) :- pokemon(name(N),_,hp(H),attack(_,D)), D > 55, H =< 100.

% ------------------------------------------------------------------------
%Define a predicate called type taking two parameters, the name of a pokemon, and the type of a pokemon,
%which succeeds only if the the named pokemon is of the specified type.
type(N,T) :- pokemon(name(N),T,_,_).

```

```
% ------------------------------------------------------------------------
%Define a predicate called dump kind taking one parameter, the kind of a pokemon, which displays complete
%information for all of the pokemon in the KB of the specified kind, doing so in a manner that is consistent with
%the representation of the pokemon in the KB.
dump_kind(T) :- pokemon(name(N),T,hp(H),attack(A,D)), write(pokemon(name(N),T,hp(H),attack(A,D))),nl,fail.

% ------------------------------------------------------------------------
%Define a predicate called family taking one parameter, presumed to be a \creatio ex nihilo" pokemon, which
%displays the \evolutionary family" of the specified pokemon, all on a given line, as illustrated in the demo.
family(N) :- write(name(N)),write(' '),evolves(N,Y),write(Y),write(' '),evolves(Y,Z),write(Z).

%------------------------------------------------------------------------
%Define a parameterless predicate called families to display all of the evolutionary pokemon families, representing
families :- cen(Name),nl,write(Name),evolves(Name,Y),write(' '),write(Y),evolves(Y,Z),write(' '),write(Z), fail.

% ------------------------------------------------------------------------
%Define a predicate called lineage taking one parameter, the name of a pokemon, which displays all of the information for
lineage(N) :- pokemon(name(N),T,hp(H),attack(A,D)), write(pokemon(name(N),T,hp(H),attack(A,D))),evolves(N,Y),nl,
pokemon(name(Y),T1,hp(H1),attack(A1,D1)), write(pokemon(name(Y),T1,hp(H1),attack(A1,D1))),evolves(Y,Z),nl,
pokemon(name(Z),T2,hp(H2),attack(A2,D2)), write(pokemon(name(Z),T2,hp(H2),attack(A2,D2))).

```

Part 4: Interaction demo with the Augmented KB

```
File  Edit  Settings  Run  Debug  Help
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.0)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- consult('pokemon.pro').
true.

?- display_cen.
pikachu
bulbasaur
caterpie
charmander
vulpix
poliwag
squirtle
staryu
false.

?- display_not_cen.
raichu
ivysaur
venusaur
metapod
butterfree
charmeleon
charizard
ninetails
poliwhirl
poliwrath
wartortle
blastoise
starmie
false.

?- generator(Name,fire).
Name = charmander ;
Name = charmeleon ;
Name = charizard ;
Name = vulpix ;
Name = ninetails.

?- generator(Name,water).
Name = poliwag ;
Name = poliwhirl ;
Name = poliwrath ;
Name = squirtle ;
Name = wartortle ;
Name = blastoise ;
Name = staryu ;
Name = starmie.
```

```
?- generator(Name,electric).
Name = pikachu ;
Name = raichu.

?- generator(Name,grass).
Name = bulbasaur ;
Name = ivysaur ;
Name = venusaur ;
Name = caterpie ;
Name = metapod ;
Name = butterfree.

?- display_names.
pikachu
raichu
bulbasaur
ivysaur
venusaur
caterpie
metapod
butterfree
charmander
charmeleon
charizard
vulpix
ninetails
poliwag
poliwhirl
poliwrath
squirtle
wartortle
blastoise
staryu
starmie
false.

?- display_attacks.
gnaw
thunder-shock
leech-seed
vine-whip
poison-powder
gnaw
stun-spore
whirlwind
scratch
slash
royal-blaze
confuse-ray
fire-blast
water-gun
amnesia
dashing-punch
bubble
waterfall
hydro-pump
slap
star-freeze
false.

?- display_cen_attacks.
gnaw
leech-seed
gnaw
scratch
confuse-ray
water-gun
bubble
slap
false.

?- indicate_attack(charmander).
charmander-->scratch
false.

?- indicate_attack(bulbasaur).
bulbasaur-->leech-seed
false.

?- indicate_attacks.
pikachu-->gnaw
raichu-->thunder-shock
bulbasaur-->leech-seed
ivysaur-->vine-whip
venusaur-->poison-powder
caterpie-->gnaw
metapod-->stun-spore
butterfree-->whirlwind
charmander-->scratch
charmeleon-->slash
charizard-->royal-blaze
vulpix-->confuse-ray
ninetails-->fire-blast
poliwag-->water-gun
poliwhirl-->amnesia
poliwrath-->dashing-punch
squirtle-->bubble
wartortle-->waterfall
blastoise-->hydro-pump
staryu-->slap
starmie-->star-freeze
false.

?- powerful(Name).
Name = raichu ;
Name = venusaur ;
Name = butterfree ;
Name = charizard ;
Name = ninetails ;
Name = wartortle ;
Name = blastoise ;
false.

?- tough(Name).
Name = venusaur ;
Name = butterfree ;
Name = charizard ;
Name = poliwrath ;
Name = blastoise ;
false.

?- awesome(Name).
Name = venusaur ;
Name = butterfree ;
Name = charizard ;
Name = blastoise ;
false.
```

```
?- powerful_but_vulnerable(Name).
Name = raichu ;
Name = ninetails ;
Name = wartortle ;
false.

?- type(squirtle,Type).
Type = water.

?- type(caterpie,Type).
Type = grass.

?- type(Name,fire),write(Name),nl,fail.
charmander
charmeleon
charizard
vulpix
ninetails
false.

?- dump_kind(water).
pokemon(name(poliwag),water,hp(60),attack(water-gun,30))
pokemon(name(poliwhirl),water,hp(80),attack(amnesia,30))
pokemon(name(poliwrath),water,hp(140),attack(dashing-punch,50))
pokemon(name(squirtle),water,hp(40),attack(bubble,10))
pokemon(name(wartortle),water,hp(80),attack(waterfall,60))
pokemon(name(blastoise),water,hp(140),attack(hydro-pump,60))
pokemon(name(staryu),water,hp(40),attack(slap,20))
pokemon(name(starmie),water,hp(60),attack(star-freeze,20))
false.

?- dump_kind(grass).
pokemon(name(bulbasaur),grass,hp(40),attack(leech-seed,20))
pokemon(name(ivysaur),grass,hp(60),attack(vine-whip,30))
pokemon(name(venusaur),grass,hp(140),attack(poison-powder,70))
pokemon(name(caterpie),grass,hp(50),attack(gnaw,20))
pokemon(name(metapod),grass,hp(70),attack(stun-spore,20))
pokemon(name(butterfree),grass,hp(130),attack(whirlwind,80))
false.

?- family(pikachu).
name(pikachu) raichu
false.

?- family(bulbasaur).
name(bulbasaur) ivysaur venusaur
true.

?- family(caterpie).
name(caterpie) metapod butterfree
true.

?- families.

pikachu raichu
bulbasaur ivysaur venusaur
caterpie metapod butterfree
charmander charmeleon charizard
vulpix ninetails
poliwag poliwhirl poliwrath
squirtle wartortle blastoise
staryu starmie
false.

?- lineage(pikachu).
pokemon(name(pikachu),electric,hp(60),attack(gnaw,10))
pokemon(name(raichu),electric,hp(90),attack(thunder-shock,90))
false.

?- lineage(squirtle).
pokemon(name(squirtle),water,hp(40),attack(bubble,10))
pokemon(name(wartortle),water,hp(80),attack(waterfall,60))
pokemon(name(blastoise),water,hp(140),attack(hydro-pump,60))
true .

?- lineage(wartortle).
pokemon(name(wartortle),water,hp(80),attack(waterfall,60))
pokemon(name(blastoise),water,hp(140),attack(hydro-pump,60))
false.

?- lineage(blastoise).
pokemon(name(blastoise),water,hp(140),attack(hydro-pump,60))
false.

?- lineage(charmander).
pokemon(name(charmander),fire,hp(50),attack(scratch,10))
pokemon(name(charmeleon),fire,hp(80),attack(slash,50))
pokemon(name(charizard),fire,hp(170),attack(royal-blaze,100))
true .
```

Part 5: KB Augmented by 12 Pokemon

```
22  cen(shinx).
23  cen(sprigatito).
24  cen(scorbunny).
25  cen(piplup).

evolves(shinx,luxio).
evolves(luxio,luxray).
evolves(sprigatito,floragato).
evolves(floragato,meowscarada).
evolves(scorbunny,raboot).
evolves(raboot,cinderace).
evolves(piplup,prinplup).
evolves(prinplup,empoleon).
```

```
74  pokemon(name(shinx), electric, hp(45), attack(tackle, 20)).
75  pokemon(name(luxio), electric, hp(60), attack(thunder-shock, 85)).
76  pokemon(name(luxray), electric, hp(80), attack(electric-terrain, 110)).
77
78  pokemon(name(sprigatito), grass, hp(40), attack(leafage, 30)).
79  pokemon(name(floragato), grass, hp(61), attack(seed-bomb, 55)).
80  pokemon(name(meowscarada), grass, hp(76), attack(leaf-storm, 90)).
81
82  pokemon(name(scorbunny), fire, hp(50), attack(growl, 10)).
83  pokemon(name(raboot), fire, hp(65), attack(flame-charge, 60)).
84  pokemon(name(cinderace), fire, hp(85), attack(pyro-ball, 100)).
85
86  pokemon(name(piplup), water, hp(53), attack(pound, 10)).
87  pokemon(name(prinplup), water, hp(64), attack(bubble-beam, 40)).
88  pokemon(name(empoleon), water, hp(84), attack(hydro-pump, 80)).
89
```

Part 6: Interaction demo with the KB Augmented by 12 Pokemon

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.0)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- consult('pokemon.pro').
true.

?- display_cen.
pikachu
bulbasaur
caterpie
charmander
vulpix
poliwag
squirtle
staryu
shinx
sprigatito
scorbunny
piplup
false.

?- display_not_cen.
raichu
ivysaur
venusaur
metapod
butterfree
charmeleon
charizard
ninetails
poliwhirl
poliwrath
wartortle
blastoise
starmie
luxio
luxray
floragato
meowscarada
raboot
cinderace
prinplup
empoleon
false.

?- generator(Name,fire).
Name = charmander ;
Name = charmeleon ;
Name = charizard ;
Name = vulpix ;
Name = ninetails ;
Name = scorbunny ;
Name = raboot ;
Name = cinderace.

?- generator(Name,water).
Name = poliwag ;
Name = poliwhirl ;
Name = poliwrath ;
Name = squirtle ;
Name = wartortle ;
Name = blastoise ;
Name = staryu ;
Name = starmie ;
Name = piplup ;
Name = prinplup ;
Name = empoleon.

?- generator(Name,electric).
Name = pikachu ;
Name = raichu ;
Name = shinx ;
Name = luxio ;
Name = luxray.

?- generator(Name,grass).
Name = bulbasaur ;
Name = ivysaur ;
Name = venusaur ;
Name = caterpie ;
Name = metapod ;
Name = butterfree ;
Name = sprigatito ;
Name = floragato ;
Name = meowscarada.
```

```
?- display_attacks.
gnaw
thunder-shock
leech-seed
vine-whip
poison-powder
gnaw
stun-spore
whirlwind
scratch
slash
royal-blaze
confuse-ray
fire-blast
water-gun
amnesia
dashing-punch
bubble
waterfall
hydro-pump
slap
star-freeze
tackle
thunder-shock
electric-terrain
leafage
seed-bomb
leaf-storm
growl
flame-charge
pyro-ball
pound
bubble-beam
hydro-pump
false.

?-

?- display_cen_attacks.
gnaw
leech-seed
gnaw
scratch
confuse-ray
water-gun
bubble
slap
tackle
leafage
growl
pound
false.

?- indicate_attack(charmander).
charmander-->scratch
false.

?- indicate_attack(bulbasaur).
bulbasaur-->leech-seed
false.

?- indicate_attacks.
pikachu-->gnaw
raichu-->thunder-shock
bulbasaur-->leech-seed
ivysaur-->vine-whip
venusaur-->poison-powder
caterpie-->gnaw
metapod-->stun-spore
butterfree-->whirlwind
charmander-->scratch
charmeleon-->slash
charizard-->royal-blaze
vulpix-->confuse-ray
ninetails-->fire-blast
poliwag-->water-gun
poliwhirl-->amnesia
poliwrath-->dashing-punch
squirtle-->bubble
wartortle-->waterfall
blastoise-->hydro-pump
staryu-->slap
starmie-->star-freeze
shinx-->tackle
luxio-->thunder-shock
luxray-->electric-terrain
sprigatito-->leafage
floragato-->seed-bomb
meowscarada-->leaf-storm
scorbunny-->growl
raboot-->flame-charge
cinderace-->pyro-ball
piplup-->pound
prinplup-->bubble-beam
empoleon-->hydro-pump
false.

?- powerful(Name).
Name = raichu ;
Name = venusaur ;
Name = butterfree ;
Name = charizard ;
Name = ninetails ;
Name = wartortle ;
Name = blastoise ;
Name = luxio ;
Name = luxray ;
Name = meowscarada ;
Name = raboot ;
Name = cinderace ;
Name = empoleon.
```

```
?- tough(Name).
Name = venusaur ;
Name = butterfree ;
Name = charizard ;
Name = poliwrath ;
Name = blastoise ;
false.

?- awesome(Name).
Name = venusaur ;
Name = butterfree ;
Name = charizard ;
Name = blastoise ;
false.

?- powerful_but_vulnerable(Name).
Name = raichu ;
Name = ninetails ;
Name = wartortle ;
Name = luxio ;
Name = luxray ;
Name = meowscarada ;
Name = raboot ;
Name = cinderace ;
Name = empoleon.

?- type(squirtle,Type).
Type = water.

?- type(caterpie,Type).
Type = grass.

?- type(Name,fire),write(Name),nl,fail.
charmander
charmeleon
charizard
vulpix
ninetails
scorbunny
raboot
cinderace
false.

?- dump_kind(water).
pokemon(name(poliwag),water,hp(60),attack(water-gun,30))
pokemon(name(poliwhirl),water,hp(80),attack(amnesia,30))
pokemon(name(poliwrath),water,hp(140),attack(dashing-punch,50))
pokemon(name(squirtle),water,hp(40),attack(bubble,10))
pokemon(name(wartortle),water,hp(80),attack(waterfall,60))
pokemon(name(blastoise),water,hp(140),attack(hydro-pump,60))
pokemon(name(staryu),water,hp(40),attack(slap,20))
pokemon(name(starmie),water,hp(60),attack(star-freeze,20))
pokemon(name(piplup),water,hp(53),attack(pound,10))
pokemon(name(prinplup),water,hp(64),attack(bubble-beam,40))
pokemon(name(empoleon),water,hp(84),attack(hydro-pump,80))
false.
?- dump_kind(grass).
pokemon(name(bulbasaur),grass,hp(40),attack(leech-seed,20))
pokemon(name(ivysaur),grass,hp(60),attack(vine-whip,30))
pokemon(name(venusaur),grass,hp(140),attack(poison-powder,70))
pokemon(name(caterpie),grass,hp(50),attack(gnaw,20))
pokemon(name(metapod),grass,hp(70),attack(stun-spore,20))
pokemon(name(butterfree),grass,hp(130),attack(whirlwind,80))
pokemon(name(sprigatito),grass,hp(40),attack(leafage,30))
pokemon(name(floragato),grass,hp(61),attack(seed-bomb,55))
pokemon(name(meowscarada),grass,hp(76),attack(leaf-storm,90))
false.

?- family(pikachu).
name(pikachu) raichu
false.

?- family(bulbasaur).
name(bulbasaur) ivysaur venusaur
true.

?- family(caterpie).
name(caterpie) metapod butterfree
true.

?- families.

pikachu raichu
bulbasaur ivysaur venusaur
caterpie metapod butterfree
charmander charmeleon charizard
vulpix ninetails
poliwag poliwhirl poliwrath
squirtle wartortle blastoise
staryu starmie
shinx luxio luxray
sprigatito floragato meowscarada
scorbunny raboot cinderace
piplup prinplup empoleon
false.

?- lineage(pikachu).
pokemon(name(pikachu),electric,hp(60),attack(gnaw,10))
pokemon(name(raichu),electric,hp(90),attack(thunder-shock,90))
false.

?- lineage(squirtle).
pokemon(name(squirtle),water,hp(40),attack(bubble,10))
pokemon(name(wartortle),water,hp(80),attack(waterfall,60))
pokemon(name(blastoise),water,hp(140),attack(hydro-pump,60))
true .

?- lineage(wartortle).
pokemon(name(wartortle),water,hp(80),attack(waterfall,60))
pokemon(name(blastoise),water,hp(140),attack(hydro-pump,60))
false.

?- lineage(blastoise).
pokemon(name(blastoise),water,hp(140),attack(hydro-pump,60))
false.

?- lineage(charmander).
pokemon(name(charmander),fire,hp(50),attack(scratch,10))
pokemon(name(charmeleon),fire,hp(80),attack(slash,50))
pokemon(name(charizard),fire,hp(170),attack(royal-blaze,100))
true .

?-
```

Task 2 - List Processing

Head/Tail Exercises

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.0)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- [H|T] = [red, yellow, blue, green].
H = red,
T = [yellow, blue, green].

?- [H,T] = [red, yellow, blue, green].
false.

?- [F|_] = [red, yellow, blue, green].
F = red.

?- [_|[S|_]] = [red, yellow, blue, green].
S = yellow.

?- [F|[S|R]] = [red, yellow, blue, green].
F = red,
S = yellow,
R = [blue, green].

?- List = [this|[and, that]].
List = [this, and, that].

?- List = [this, and, that].
List = [this, and, that].

?- [a,[b, c]] = [a, b, c].
false.

?- [a|[b, c]] = [a, b, c].
true.

?- [cell(Row,Column)|Rest] = [cell(1,1), cell(3,2), cell(1,3)].
Row = Column, Column = 1,
Rest = [cell(3, 2), cell(1, 3)].

?- [X|Y] = [one(un, uno), two(dos, deux), three(trois, tres)].
X = one(un, uno),
Y = [two(dos, deux), three(trois, tres)].
```

List Processing Code:

```
% ------------------------------------------------------------------------
% ------------------------------------------------------------------------
% File: list processors.pro
% Line: Loosely represented List Processor in Prolog
% ------------------------------------------------------------------------
% ------------------------------------------------------------------------

% Code: First
first([H|_], H).

% Code: Rest
rest([_|T], T).

% Code: Last
last([H|[]], H).
last([_|T], Result) :- last(T, Result).

% Code: Nth
nth(0,[H|_],H).
nth(N,[_|T],E) :- K is N - 1, nth(K,T,E).

% Code: Writelist

writelist([]).
writelist([H|T]) :- write(H), nl, writelist(T).

%Code: sum
sum([],0).
sum([Head|Tail],Sum) :- sum(Tail,SumOfTail), Sum is Head + SumOfTail.

%Code: Add first
add_first(X,L,[X|L]).

%Code: Add last
add_last(X,[],[X]).
add_last(X,[H|T],[H|TX]) :- add_last(X,T,TX).

%Code: Iota
iota(0,[]).
iota(N,IotaN) :- K is N - 1, iota(K,IotaK), add_last(N,IotaK,IotaN).

%Code: Pick
pick(L,Item) :- length(L,Length), random(0,Length,RN), nth(RN,L,Item).

%Code: Make set
make_set([],[]).
make_set([H|T],TS) :- member(H,T),
make_set(T,TS).
make_set([H|T],[H|TS]) :- make_set(T,TS).
```

```
% ------------------------------------------------------------------------
% Extended KB for list proccessors.pro
% ------------------------------------------------------------------------

% Code: Product
product([], 1).
product([Head|Tail],Product) :- product(Tail,ProductOfTail), Product is Head * ProductOfTail.

%Code: Factorial
factorial(0,0).
factorial(Num,Name) :- iota(Num,Iota), product(Iota,Product), Name is Product.

%Code: Iota
make_list(0,_,[]).
make_list(Num,Element,Name) :- K is Num - 1, make_list(K,Element,NameK), add_last(Element,NameK,Name).

%Code: but_first
but_first([],[]).
but_first([_],[]).
but_first([_|F],F).

%Code: but_last
but_last([],[]).
but_last([_],[]).
but_last([H|T],Name) :- reverse(T,[_|B]), reverse(B,RDC),
add_first(H,RDC,Name).

%Code: is_Palindrome
is_palindrome([]).
is_palindrome([_]).
is_palindrome(List) :- first(List,FirstChar),
last(List,LastChar), FirstChar = LastChar, but_first(List,F), but_last(F,G), is_palindrome(G).

%Code: noun_phrase
noun_phrase(Name) :- pick([cheerful,puny,silly,rich,smart,despair],Adj),
pick([picture,fruit,candy,airport,bunny,ocean,dress,doll],Noun),
add_last(Adj,[the],Start), add_last(Noun,Start,Name).

%Code: sentence(Name)
sentence(Name) :- noun_phrase(A), noun_phrase(B), pick([sang,walked,moved,fly,laughed,danced,drank],Verb),
add_last(Verb, A, C),
append(C, B, Name).
```

Demo for Example List Processors:

```
?- consult('list processors.pro').
true.

?- first([apple],First).
First = apple.

?- first([c,d,e,f,g,a,b],P).
P = c.

?- rest([apple],Rest).
Rest = [].

?- rest([c,d,e,f,g,a,b],Rest).
Rest = [d, e, f, g, a, b].

?- last([peach],Last).
Last = peach ,

?- last([c,d,e,f,g,a,b],P).
P = b ,

?- nth(0,[zero,one,two,three,four],Element).
Element = zero ,

?- nth(3,[four,three,two,one,zero],Element).
Element = one ,

?- writelist([red,yellow,blue,green,purple,orange]).
red
yellow
blue
green
purple
orange
true.

?- sum([],Sum).
Sum = 0.

?- sum([2,3,5,7,11],SumOfPrimes).
SumOfPrimes = 28.

?- add_first(thing,[],Result).
Result = [thing].

?- add_first(racket,[prolog,haskell,rust],Languages).
Languages = [racket, prolog, haskell, rust].

?- add_last(thing,[],Result).
Result = [thing] ,

?- add_last(rust,[racket,prolog,haskell],Languages).
Languages = [racket, prolog, haskell, rust] ,

?- iota(5,Iota5).
Iota5 = [1, 2, 3, 4, 5] ,

?- iota(9,Iota9).
Iota9 = [1, 2, 3, 4, 5, 6, 7, 8, 9] ,

?- pick([cherry,peach,apple,blueberry],Pie).
Pie = cherry ,

?- pick([cherry,peach,apple,blueberry],Pie).
Pie = cherry ,

?- pick([cherry,peach,apple,blueberry],Pie).
Pie = peach ,

?- pick([cherry,peach,apple,blueberry],Pie).
Pie = blueberry ,

?- pick([cherry,peach,apple,blueberry],Pie).
Pie = cherry ,

?- pick([cherry,peach,apple,blueberry],Pie).
Pie = cherry ,

?- pick([cherry,peach,apple,blueberry],Pie).
Pie = blueberry ,

?- pick([cherry,peach,apple,blueberry],Pie).
Pie = peach ,

?- make_set([1,1,2,1,2,3,1,2,3,4],Set).
Set = [1, 2, 3, 4] ,

?- make_set([bit,bot,bet,bot,bot,bit],B).
B = [bet, bot, bit] ,

?-
```

Demo for List Processing Exercises:

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.0)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- consult('list processors.pro').
true.

?- product([],P).
P = 1.

?- product([1,3,5,7,9],Product).
Product = 945.

?- iota(9,Iota),product(Iota,Product).
Iota = [1, 2, 3, 4, 5, 6, 7, 8, 9],
Product = 362880 .

?- make_list(7,seven,Seven).
Seven = [seven, seven, seven, seven, seven, seven, seven] .

?- make_list(8,2,List).
List = [2, 2, 2, 2, 2, 2, 2, 2] .

?- but_first([a,b,c],X).
X = [b, c].

?- but_last([a,b,c,d,e],X).
X = [a, b, c, d].

?- is_palindrome([x]).
true .

?- is_palindrome([a,b,c]).
false.

?- is_palindrome([a,b,b,a]).
true .

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

?- is_palindrome([c,o,f,f,e,e,e,e,f,f,o,c]).
true .

?- noun_phrase(NP).
NP = [the, silly, fruit] .

?- noun_phrase(NP).
NP = [the, despair, airport] .

?- noun_phrase(NP).
NP = [the, silly, airport] .

?- noun_phrase(NP).
NP = [the, rich, doll] .

NP = [the, rich, doll] .

?- noun_phrase(NP).
NP = [the, smart, fruit] .

?- sentence(S).
S = [the, despair, dress, sang, the, puny, dress] .

?- sentence(S).
S = [the, silly, fruit, drank, the, silly, ocean] .

?- sentence(S).
S = [the, silly, picture, drank, the, puny, bunny] .

?- sentence(S).
S = [the, rich, ocean, sang, the, smart, ocean] .

?- sentence(S).
S = [the, silly, airport, sang, the, silly, airport] .

?- sentence(S).
S = [the, cheerful, airport, laughed, the, cheerful, bunny] .

?- sentence(S).
S = [the, cheerful, airport, laughed, the, smart, dress] .

?-

?- sentence(S).
S = [the, despair, airport, moved, the, despair, bunny] .

?- sentence(S).
S = [the, smart, doll, laughed, the, smart, dress] .

?- sentence(S).
S = [the, cheerful, fruit, drank, the, rich, bunny] .

?- sentence(S).
S = [the, smart, bunny, laughed, the, silly, airport] .

?- sentence(S).
S = [the, puny, doll, laughed, the, rich, fruit] .

?- sentence(S).
S = [the, silly, ocean, fly, the, smart, picture] .

?- sentence(S).
S = [the, smart, fruit, drank, the, rich, candy] .

?- sentence(S).
S = [the, despair, ocean, walked, the, smart, candy] .

?- sentence(S).
S = [the, puny, picture, sang, the, silly, dress] .
```