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

Written by **David Hennigan**

Learning Abstract

This assignment, consisting of two tasks, played a vital role in the development of my prolog understanding. The first task being to become acquainted with a pokemon knowledge base, add functionality to it, and add 12 pokemon. The second task consisted of list processing exercises.

Task 1 – Pokemon

Part 1: Initial Pokemon KB

```
% ------------------------------------------------------------------

% ------------------------------------------------------------------

% --- File: pokemon.pro

% --- Line: Just a few facts about 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

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

?- 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(Name),fire,_,_),write(Name),nl,fail.

charmander

charmeleon

charizard

vulpix

ninetails

false.

?- pokemon(Name,Kind,_,_),write('nks('),write(Name),write(', kind('),write(Kind),write('))'),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 .

?- pokemon(name(N),_,_,attack(poison-powder,_)).

N = venusaur .

?- pokemon(_,water,_,attack(Attack,_)),write(Attack),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(Name),_,hp(HP),_), HP > 85, write(Name),nl,fail.

raichu

venusaur

butterfree

charizard

ninetails

poliwrath

blastoise

false.

?- pokemon(name(Name),_,_,attack(_,Damage)), Damage > 60, write(Name),nl,fail.

raichu

venusaur

butterfree

charizard

ninetails

false.

?- cen(Name),pokemon(name(Name),_,hp(HP),_),write(Name),write(': '),write(HP),nl,fail.

pikachu: 60

bulbasaur: 40

caterpie: 50

charmander: 50

vulpix: 60

poliwag: 60

squirtle: 40

staryu: 40

false.

Part 3: KB Extension

```
%-----------------------------------------------------------------------
%--- diplay_cen :: displays all of the cen pokemon

display_cen :- cen(Name),write(Name),nl,fail.
display_cen.

%-----------------------------------------------------------------------
%--- display_not_cen :: displays all of the not cent pokemon

display_not_cen :- evolves(X,Y),write(Y),nl,fail.
display_not_cen.

%-----------------------------------------------------------------------
```

```
%--- generator(Name,Type) :: Name of the cen pokemon and the given type
%--- display true if a pokemon with that type exists in our kb

generator(Name,Type) :- cen(Name),pokemon(name(Name),Type,_,_).

%------------------------------------------------------------------------
%--- display_names :: Displays all of the pokemon in the kb

display_names :- pokemon(name(Name),_,_,_),write(Name),nl,fail.
display_names.

%------------------------------------------------------------------------
%--- display_attacks :: Displays all of the attacks in the kb

display_attacks :- pokemon(_,_,_,attack(Attack,_)),write(Attack),nl,fail.
display_attacks.

%------------------------------------------------------------------------
%--- display_cen_attacks :: Displays all of the attacks from cen pokemon
%--- in the kb

display_cen_attacks :- pokemon(name(Name),_,_,attack(Attack,_)),cen(Name),write(Attack),nl,fail.
display_cen_attacks.

%------------------------------------------------------------------------
%--- indicate_attack(Name) :: Displays the pokemons Name and its associated
%---                          attack

indicate_attack(Name) :- pokemon(name(Name),_,_,attack(Attack,_)),write(Name),write(' -->
'),write(Attack).
```

```
%-----------------------------------------------------------------------
%--- indicate_attacks :: Displays all of the pokemon with their associated
%---                     attacks

indicate_attacks :- indicate_attack(Name),nl,fail.
indicate_attacks.

%-----------------------------------------------------------------------
%--- powerful(Name) :: succeeds if a pokemon has an attack with more than 55
%---                   units of damage.

powerful(Name) :- pokemon(name(Name),_,_,attack(_,Damage)), Damage > 55.

%-----------------------------------------------------------------------
%--- tough(Name) :: succeeds if a pokemon can take more than 100 units of
%---                damage.

tough(Name) :- pokemon(name(Name),_,hp(HP),_), HP > 100.

%-----------------------------------------------------------------------
%--- awesome(Name) :: succeeds if a pokemon is both powerful and tough

awesome(Name) :- powerful(Name),tough(Name).

%-----------------------------------------------------------------------
%--- powerful_but_vulnerable(Name) :: succeeds if a pokemon is powerful
%---                                  but not tough.

powerful_but_vulnerable(Name) :- powerful(Name), pokemon(name(Name),_,hp(HP),_), HP < 101.
```

```
%-----------------------------------------------------------------------
%--- type(Name,Type) :: specifies whether a pokemon is of a specific type

type(Name,Type) :- pokemon(name(Name),Type,_,_).

%-----------------------------------------------------------------------
%--- dump_kind(Kind) :: displays all of the information for a pokemon of
%---                    Kind

dump_kind(Kind) :- pokemon(Name,Kind,Type,Attack),write('pokemon('),write(Name),write(','),
                   write(Kind),write(','),write(Type),write(','),write(Attack),write(')'),nl,fail.
dump_kind(Kind).

%-----------------------------------------------------------------------
%--- family(Name) :: displays the family of the cen pokemon

family(Name) :- cen(Name),evolves(Name,Evolution), (evolves(Evolution,NextEvolution) ->
                write(Name),write(' '),write(Evolution),write(' '),write(NextEvolution) ;
                cen(Name),evolves(Name,Evolution),write(Name),write(' '),write(Evolution)).



%-----------------------------------------------------------------------
%--- families :: displays all of the families.

families :- family(Name),nl,fail.
families.

%-----------------------------------------------------------------------
%--- lineage(Name) :: displays all of the information associated with a
```

```
%---            pokemon and its evolutions

lineage(Name) :- pokemon(name(Name),Type,HP,Attack), write('pokemon(name('),write(Name),
        write('),'),write(Type),write(','),write(HP),write(','),write(Attack),write(')'),
        evolves(Name,Evolution),nl,lineage(Evolution).
lineage(Name).
```

Part 4: Interaction demo with the Augmented KB

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
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
true.
```

?- display_not_cen.

raichu

ivysaur

venusaur

metapod

butterfree

charmeleon

charizard

ninetails

poliwhirl

poliwrath

wartortle

blastoise

starmie

true.

?- generator(Name,fire).

Name = charmander ;

Name = vulpix ;

false.

?- generator(Name,water).

Name = poliwag ;

Name = squirtle ;

Name = staryu ;

false.

?- generator(Name,electric).

Name = pikachu ;

false.

?- generator(Name,grass).

Name = bulbasaur ;

Name = caterpie ;

false.

?- display_names.

pikachu

raichu

bulbasaur

ivysaur

venusaur

caterpie

metapod

butterfree

charmander

charmeleon

charizard

vulpix

ninetails

poliwag

poliwhirl

poliwrath

squirtle

wartortle

blastoise

staryu

starmie

true.

?- 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

true.

?- display_cen_attacks.

gnaw

leech-seed

gnaw

scratch

confuse-ray

water-gun

bubble

slap

true.

?- indicate_attack(charmander).

charmander --> scratch

true.

?- indicate_attack(bulbasaur).

bulbasaur --> leech-seed

true.

?- 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

true.

?- 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))
true.

?- 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))
true.

?- family(pikachu).
pikachu raichu
true.

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

?- family(caterpie).
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

true.

?- lineage(pikachu).

pokemon(name(pikachu),electric,hp(60),attack(gnaw,10))

pokemon(name(raichu),electric,hp(90),attack(thunder-shock,90))

true .

?- 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))

true .

?- lineage(blastoise).

pokemon(name(blastoise),water,hp(140),attack(hydro-pump,60))

true.

?- 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

```
% ------------------------------------------------------------------
% ------------------------------------------------------------------
% --- File: pokemon.pro
% --- Line: Just a few facts about 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).

%--- Additions to kb

cen(mudkip).
cen(torchic).
cen(treecko).
cen(elekid).

% ------------------------------------------------------------------
% --- 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).

%--- Additions to kb

evolves(mudkip,marshtomp).
evolves(marshtomp,swampert).
evolves(torchic,combusken).
evolves(combusken,blaziken).
evolves(treecko,grovyle).
evolves(grovyle,sceptile).
evolves(elekid,electabuzz).
evolves(electabuzz,electivire).

% -----------------------------------------------------------------------
% --- 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)).

%--- Additions to kb

pokemon(name(mudkip), water, hp(50), attack(water-gun, 30)).
pokemon(name(marshtomp), water, hp(75), attack(water-pulse, 60)).
pokemon(name(swampert), water, hp(110), attack(muddy-water, 90)).

pokemon(name(torchic), fire, hp(45), attack(scratch, 10)).
pokemon(name(combusken), fire, hp(60), attack(flame-charge, 50)).
pokemon(name(blaziken), fire, hp(90), attack(blaze-kick, 85)).

pokemon(name(treecko), grass, hp(40), attack(scratch, 10)).
pokemon(name(grovyle), grass, hp(50), attack(leafage, 40)).
pokemon(name(sceptile), grass, hp(75), attack(leaf-blade, 90)).

pokemon(name(elekid), electric, hp(45), attack(thundershock, 40)).
pokemon(name(electabuzz), electric, hp(70), attack(thunder-punch, 75)).
pokemon(name(electivire), electric, hp(120), attack(zap-cannon, 120)).

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

Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
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

mudkip

torchic

treecko

elekid

true.

?- display_not_cen.

raichu

ivysaur

venusaur

metapod

butterfree

charmeleon

charizard

ninetails

poliwhirl

poliwrath

wartortle

blastoise

starmie

marshtomp

swampert

combusken

blaziken

grovyle

sceptile

electabuzz

electivire

true.

?- generator(Name,fire).

Name = charmander ;

Name = vulpix ;

Name = torchic ;

false.

?- generator(Name,water).

Name = poliwag ;

Name = squirtle ;

Name = staryu ;

Name = mudkip ;

false.

?- generator(Name,electric).

Name = pikachu ;

Name = elekid ;

false.

?- generator(Name,grass).

Name = bulbasaur ;

Name = caterpie ;

Name = treecko ;

false.

?- display_names.
pikachu
raichu
bulbasaur
ivysaur
venusaur
caterpie
metapod
butterfree
charmander
charmeleon
charizard
vulpix
ninetails
poliwag
poliwhirl
poliwrath
squirtle
wartortle
blastoise
staryu
starmie
mudkip
marshtomp
swampert
torchic
combusken
blaziken

treecko

grovyle

sceptile

elekid

electabuzz

electivire

true.

?- 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

water-gun

water-pulse

muddy-water

scratch

flame-charge

blaze-kick

scratch

leafage

leaf-blade

thundershock

thunder-punch

zap-cannon

true.

?- display_cen_attacks.

gnaw

leech-seed

gnaw

scratch

confuse-ray

water-gun

bubble

slap

water-gun

scratch

scratch

thundershock

true.

?- indicate_attack(electabuzz).

electabuzz --> thunder-punch

true.

?- indicate_attack(mudkip).

mudkip --> water-gun

true.

?- 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

mudkip --> water-gun

marshtomp --> water-pulse

swampert --> muddy-water

torchic --> scratch

combusken --> flame-charge

blaziken --> blaze-kick

treecko --> scratch

grovyle --> leafage

sceptile --> leaf-blade

elekid --> thundershock

electabuzz --> thunder-punch

electivire --> zap-cannon

true.

?- powerful(Name).

Name = raichu ;

Name = venusaur ;

Name = butterfree ;

Name = charizard ;

Name = ninetails ;

Name = wartortle ;

Name = blastoise ;

Name = marshtomp ;

Name = swampert ;

Name = blaziken ;

Name = sceptile ;

Name = electabuzz ;

Name = electivire.

?- tough(Name).

Name = venusaur ;

Name = butterfree ;

Name = charizard ;

Name = poliwrath ;

Name = blastoise ;

Name = swampert ;

Name = electivire.

?- awesome(Name).

Name = venusaur ;

Name = butterfree ;

Name = charizard ;

Name = blastoise ;

Name = swampert ;

Name = electivire.

?- powerful_but_vulnerable(Name).

Name = raichu ;

Name = ninetails ;

Name = wartortle ;

Name = marshtomp ;

Name = blaziken ;

Name = sceptile ;

Name = electabuzz ;

false.

?- type(elekid,Type).

Type = electric.

?- type(treecko,Type).

Type = grass.

?- type(Name,fire),write(Name),nl,fail.
charmander
charmeleon
charizard
vulpix
ninetails
torchic
combusken
blaziken
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(mudkip),water,hp(50),attack(water-gun,30))
pokemon(name(marshtomp),water,hp(75),attack(water-pulse,60))
pokemon(name(swampert),water,hp(110),attack(muddy-water,90))
true.

?- 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(treecko),grass,hp(40),attack(scratch,10))

pokemon(name(grovyle),grass,hp(50),attack(leafage,40))

pokemon(name(sceptile),grass,hp(75),attack(leaf-blade,90))

true.

?- family(elekid).

elekid electabuzz electivire

true.

?- family(treecko).

treecko grovyle sceptile

true.

?- family(mudkip).

mudkip marshtomp swampert

true.

?- families.

pikachu raichu

bulbasaur ivysaur venusaur

caterpie metapod butterfree

charmander charmeleon charizard

vulpix ninetails

poliwag poliwhirl poliwrath

squirtle wartortle blastoise

staryu starmie

mudkip marshtomp swampert

torchic combusken blaziken

treecko grovyle sceptile

elekid electabuzz electivire

true.

?- lineage(torchic).

pokemon(name(torchic),fire,hp(45),attack(scratch,10))

pokemon(name(combusken),fire,hp(60),attack(flame-charge,50))

pokemon(name(blaziken),fire,hp(90),attack(blaze-kick,85))

true .

?- lineage(electabuzz).

pokemon(name(electabuzz),electric,hp(70),attack(thunder-punch,75))

pokemon(name(electivire),electric,hp(120),attack(zap-cannon,120))

true .

?- lineage(elekid).

pokemon(name(elekid),electric,hp(45),attack(thundershock,40))

pokemon(name(electabuzz),electric,hp(70),attack(thunder-punch,75))

pokemon(name(electivire),electric,hp(120),attack(zap-cannon,120))

true .

?- lineage(electivire).

pokemon(name(electivire),electric,hp(120),attack(zap-cannon,120))

true.

?- lineage(mudkip).

pokemon(name(mudkip),water,hp(50),attack(water-gun,30))

pokemon(name(marshtomp),water,hp(75),attack(water-pulse,60))

pokemon(name(swampert),water,hp(110),attack(muddy-water,90))

true .

Task 2 – List Processing

Head/Tail Exercises

Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)

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
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
%--- first(List,First) :: returns the first element of a list.

first([H|_],H).
```

```
%---------------------------------------------------------------------
%--- rest(List,Remainder) :: returns the remainder of a list.

rest([_|T],T).

%---------------------------------------------------------------------
%--- last(List, Result) :: returns the last element of a list.

last([H|[]],H). %--- Requires the remainder of the list to be empty
last([_|T], Result) :- last(T, Result).

%---------------------------------------------------------------------
%--- nth(Number,List,Result) :: returns the nth element of a list.

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

%---------------------------------------------------------------------
%--- writelist(List) :: prints all of the contents of a list on seperate
%---                    lines.

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

%---------------------------------------------------------------------
%--- sum(List, Result) :: returns the sum of every element in a list.

sum([],0).
sum([Head|Tail],Sum) :-
```

```
    sum(Tail,SumOfTail),
    Sum is Head + SumOfTail.

%--------------------------------------------------------------------
%--- add_first(Element, List, Result) :: returns a new list with an
%---                                element added to the front.

add_first(X,L,[X|L]).

%--------------------------------------------------------------------
%--- add_last(Element, List, Result) :: returns a new list with an
%---                                element added to the end.

add_last(X,[],[X]).
add_last(X,[H|T],[H|TX]) :- add_last(X,T,TX).

%--------------------------------------------------------------------
%--- iota(N,Result)

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

%--------------------------------------------------------------------
%--- pick(List,Result) :: picks an element from the list.

pick(L, Item) :-
    length(L,Length),
```

```
    random(0,Length,RN),
    nth(RN,L,Item).

%----------------------------------------------------------------------
%--- make_set(List,Result) :: transforms a list into a 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).

%----------------------------------------------------------------------
%--- product(List,Result) :: takes the product of a list

product([],1).
product([H|T],Result) :- product(T,ProductOfTail),
                         Result is H * ProductOfTail.

%----------------------------------------------------------------------
%--- factorial(N,Result) :: takes the factorial of N

factorial(N,Result) :- iota(N, IotaN), product(IotaN, Result).

%----------------------------------------------------------------------
%--- make_list(N, Item, Result) :: Makes a list containing an Item N times

make_list(0, _, []).
make_list(N, Item, [Item|ResultOfRec]) :- K is N - 1, make_list(K, Item, ResultOfRec).
```

```
%---------------------------------------------------------------------
%--- but_first(List,CDR) :: Produces the cdr of a list

but_first([_],[]).
but_first([_|T],T).

%---------------------------------------------------------------------
%--- but_last(List,RDC) :: Produces the rdc of a list

but_last([_],[]).
but_last(List, RDC) :- reverse(List,ReversedList),
                       but_first(ReversedList,ReversedListWithoutFirst),
                       reverse(ReversedListWithoutFirst,RDC).

%---------------------------------------------------------------------
%--- is_palindrome(List) :: Determines if a list is a palindrome

is_palindrome([]).
is_palindrome([_]).
is_palindrome(List) :- first(List, First), last(List, Last), First = Last,
                       but_first(List,ListWithoutFirst),
                       but_last(ListWithoutFirst, ListWithElesRemoved),
                       is_palindrome(ListWithElesRemoved).

%---------------------------------------------------------------------
%--- noun_phrase(Phrase) :: produces a noun phrase consisting of the word
%                           the followed by an Adjective then a Noun.

noun_phrase([the,Adjective,Noun]) :-
```

```
    pick([ambitious,delightful,victorius,witty,zealous,gentle], Adjective),
    pick([dog,robot,samurai,dragon,sandwich,helicopter,musician,artist], Noun).

%---------------------------------------------------------------------
%--- sentence(Sentence) :: produces a sentence consisting of a noun_phrase
%---                       followed by a past tense verb then by another
%---                       noun_phrase

sentence(Sentence) :- pick([attacked,saw,followed,loved,hated,admired,taught], Verb),
                      noun_phrase(Phrase1),
                      noun_phrase(Phrase2),
                      add_last(Verb, Phrase1, PhraseWthVerb),
                      append(PhraseWthVerb, Phrase2, Sentence).
```

Demo for Example List Processors

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
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.

?- 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 ;

false.

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

P = b ;

false.

?- 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 = apple .

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

Pie = blueberry .

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

Pie = blueberry .

?- 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 = apple .

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

Pie = blueberry .

?- 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 9.0.4)

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, ambitious, dragon] .

?- noun_phrase(NP).

NP = [the, witty, robot] .

?- noun_phrase(NP).

NP = [the, delightful, robot] .

?- noun_phrase(NP).

NP = [the, ambitious, musician] .

?- noun_phrase(NP).

NP = [the, witty, sandwich] .

?- sentence(S).

S = [the, ambitious, samurai, saw, the, gentle, dragon] .

?- sentence(S).

S = [the, witty, robot, saw, the, ambitious, dragon] .

?- sentence(S).

S = [the, zealous, sandwich, taught, the, witty, sandwich] .

?- sentence(S).

S = [the, gentle, dragon, attacked, the, delightful, robot] .

?- sentence(S).

S = [the, witty, musician, saw, the, victorius, musician] .

?- sentence(S).

S = [the, zealous, helicopter, saw, the, ambitious, dragon] .

?- sentence(S).

S = [the, delightful, artist, admired, the, victorius, artist] .

?- sentence(S).

S = [the, delightful, samurai, admired, the, victorius, musician] .

?- sentence(S).

S = [the, victorius, sandwich, followed, the, victorius, artist] .

?- sentence(S).

S = [the, ambitious, samurai, loved, the, zealous, helicopter] .

?- sentence(S).

S = [the, delightful, robot, followed, the, delightful, dog] .

?- sentence(S).

S = [the, victorius, robot, hated, the, delightful, samurai] .

?- sentence(S).

S = [the, delightful, musician, taught, the, ambitious, artist] .

?- sentence(S).

S = [the, victorius, dog, loved, the, gentle, artist] .

?- sentence(S).

S = [the, ambitious, robot, loved, the, delightful, sandwich] .

?- sentence(S).

S = [the, gentle, dragon, attacked, the, ambitious, dragon] .