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:

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

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

% --- 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:

```
#1: cen(pikachu).

#2: cen(raichu).

#3: cen(Name).

#4: cen(Name), write(Name),nl,fail.

#5: evolves(squirtle,wartortle).

#6: evolves(wartortle ,squirtle).

#7: evolves(squirtle,blastoise).

#8: evolves(X,Y), evolves(Y,Z).

#9: evolves(X,Y), evolves(Y,Z), write(X --> Z),nl,fail.

#10: pokemon(name(X),_,_,_), write(X),nl,fail.

#11: pokemon(name(X),fire,_,_), write(X),nl,fail.

#12: pokemon(Name,Kind,_,_), write(nks(Name,kind(Kind))),nl,fail.

#13: pokemon(name(N),_,_,attack(waterfall,_)).

#14: pokemon(name(N),_,_,attack(poison-powder,_)).

#15: pokemon(_,water,_,attack(Attack,_)), write(Attack),nl,fail.

#16: pokemon(name(poliwhirl),_,hp(HP),_).

#17: pokemon(name(butterfree),_,hp(HP),_).

#18: pokemon(name(Name),_,hp(Hp),_),Hp > 85, write(Name),nl,fail.

#19: pokemon(name(Name),_,_,attack(_,Ap)),Ap > 60, write(Name),nl,fail.

#20: pokemon(name(Name),_,hp(Hp),_),cen(Name), write(Name : Hp),nl,fail.
```

Part 3:

Included are the additions made to the pokemon.pro document.

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

% --- display_cen:: lists the names of all of the "creatio ex nihilo" Pokemon

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

display_cen.


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

% --- display_not_cen:: lists the names of all of the Pokemon who are not "creatio ex nihilo" Pokemon

display_not_cen :- pokemon(name(Name),_,_,_), not(cen(Name)), write(Name), nl, fail.

display_not_cen.


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

% --- generator(Name, Type) :: Name is the name of the Pokemon and Type is the type the Pokemon is

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



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

% --- display_names:: lists the names of all of the Pokemon

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

display_names.


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

```
% --- display_attacks:: lists the attacks of all of the Pokemon

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

display_attacks.


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

% --- display_cen_attacks:: lists the attacks of all of the "creatio ex nihilo" Pokemon

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

display_cen_attacks.


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

% --- display_cen_attacks:: lists the attacks of all of the "creatio ex nihilo" Pokemon

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

display_cen_attacks.


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

% --- indicate_attack(Name) :: Name is the name of the Pokemon

indicate_attack(Name) :- pokemon(name(Name),_,_,attack(Attack,_)), write(Name-->Attack),
nl, fail.

indicate_attack(Name).


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

% --- indicate_attacks:: lists the names and corresponding attacks of all of the Pokemon

indicate_attacks :- pokemon(name(Name),_,_,attack(Attack,_)), write(Name->Attack), nl, fail.
```

```
indicate_attacks.

% ---------------------------------------------------------------------
% --- powerful(Name) :: Name is the name of the Pokemon
powerful(Name) :- pokemon(name(Name),_,_,attack(_,Ap)),Ap > 55.

% ---------------------------------------------------------------------
% --- tough(Name) :: Name is the name of the Pokemon
tough(Name) :- pokemon(name(Name),_,hp(Hp),_),Hp > 100.

% ---------------------------------------------------------------------
% --- awesome(Name) :: Name is the name of the Pokemon

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

% ---------------------------------------------------------------------
% --- powerful_but_vulnerable(Name) :: Name is the name of the Pokemon

powerful_but_vulnerable(Name) :- powerful(Name), not(tough(Name)).

% ---------------------------------------------------------------------
% --- type(Name, Type) :: Name is the name of the Pokemon and Type is the type the Pokemon
is
type(Name,Type) :- pokemon(name(Name),Type,_,_).
```

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

% --- dump_kind(Kind):: Kind is the kind of Pokemon we wish to dump info about

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

dump_kind(Kind).


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

% --- family(Name) :: Name is the name of the Pokemon

family(Name) :- write(Name),write(' '), evolves(Name,Y), write(Y), write(' '), evolves(Y,Z),
write(Z).

family(Name).


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

% --- families:: lists all of the evolutionary Pokemon families

families :-cen(Name), write(Name), write(' '), evolves(Name,Y), write(Y), write(' '),

( evolves(Y,Z)

-> write(Z), nl, fail

; nl, fail

).

families.


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

% --- lineage(Name) :: Name is the name of the Pokemon

lineage(Name) :-
```

```
pokemon(name(Name),Type,Hp,Attack), write(pokemon(name(Name))),write(','),
write(Type),write(','), write(Hp), write(','), write(Attack), write(')'), nl,

evolves(Name,Y), pokemon(name(Y),Ytype,Yhp,Yattack), write(pokemon(name(Y))),write(','),
write(Ytype),write(','), write(Yhp), write(','), write(Yattack), write(')'), nl,

evolves(Y,Z), pokemon(name(Z),Ztype,Zhp,Zattack), write(pokemon(name(Z))),write(','),
write(Ztype),write(','), write(Zhp), write(','), write(Zattack), write(')'), nl.

lineage(Name).
```

Part 4:

```
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').
Warning: c:/users/owner/documents/prolog/pokemon.pro:108:
Warning:    Singleton variables: [Name]
Warning: c:/users/owner/documents/prolog/pokemon.pro:138:
Warning:    Singleton variables: [Kind]
Warning: c:/users/owner/documents/prolog/pokemon.pro:143:
Warning:    Singleton variables: [Name]
Warning: c:/users/owner/documents/prolog/pokemon.pro:160:
Warning:    Singleton variables: [Name]
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 = 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
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.

?- aweseom(Name).
ERROR: Unknown procedure: aweseom/1 (DWIM could not correct goal)
?- 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.

?- family(pikachu).
pikachu raichu
true.

?- family(bulbasaur).
bulbasaur ivysaur venusaur
true
Unknown action: f (h for help)
Action? ,

?- 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))
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:

Included are the additions made to the pokemon.pro document.

```
cen(fennekin).

cen(chespin).

cen(piplup).

cen(pawmi).


evolves(fennekin,braixen).

evolves(braixen,delphox).

evolves(chespin,quilladin).

evolves(quilladin,chesnaught).

evolves(piplup,prinplup).

evolves(prinplup,empoleon).

evolves(pawmi,pawmo).

evolves(pawmo,pawmot).


pokemon(name(fennekin), fire, hp(40), attack(scratch, 40)).

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

pokemon(name(delphox), fire, hp(75), attack(psyshock, 80)).


pokemon(name(chespin), grass, hp(60), attack(vine-whip, 45)).

pokemon(name(quilladin), grass, hp(60), attack(bite, 60)).

pokemon(name(chesnaught), grass, hp(90), attack(seed-bomb, 80)).
```

```
pokemon(name(piplup), water, hp(50), attack(pound, 40)).

pokemon(name(prinplup), water, hp(65), attack(bubble-beam, 65)).

pokemon(name(empoleon), water, hp(85), attack(drill-peck, 80)).


pokemon(name(pawmi), electric, hp(45), attack(thunder-shock, 40)).

pokemon(name(pawmo), electric, hp(60), attack(spark, 65)).

pokemon(name(pawmot), electric, hp(70), attack(discharge, 80)).
```

Part 6:

For this demo, I mostly copied the part 4 demo but inserted some testing of the new Pokémon as well.

```
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').
Warning: c:/users/owner/documents/prolog/pokemon.pro:138:
Warning:    Singleton variables: [Name]
Warning: c:/users/owner/documents/prolog/pokemon.pro:168:
Warning:    Singleton variables: [Kind]
Warning: c:/users/owner/documents/prolog/pokemon.pro:173:
Warning:    Singleton variables: [Name]
Warning: c:/users/owner/documents/prolog/pokemon.pro:190:
Warning:    Singleton variables: [Name]
true.

?- display_cen.
pikachu
bulbasaur
caterpie
charmander
vulpix
poliwag
squirtle
staryu
fennekin
chespin
piplup
pawmi
true.

?- display_not_cen.
raichu
ivysaur
venusaur
metapod
butterfree
charmeleon
charizard
ninetails
poliwhirl
poliwrath
wartortle
blastoise
starmie
braixen
delphox
quilladin
chesnaught
prinplup
empoleon
pawmo
pawmot
true.

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

?- 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 = pawmi ;
Name = pawmo ;
Name = pawmot.

?- generator(Name,grass).
Name = bulbasaur
Unknown action: (h for help)
Action? ;
Name = ivysaur ;
Name = venusaur ;
Name = caterpie ;
Name = metapod ;
Name = butterfree ;
Name = chespin ;
Name = quilladin ;
Name = chesnaught.

?- display_names.
pikachu
raichu
bulbasaur
ivysaur
venusaur
caterpie
metapod
butterfree
charmander
charmeleon
charizard
vulpix
ninetails
poliwag
poliwhirl
poliwrath
squirtle
wartortle
blastoise
staryu
starmie
fennekin
braixen
delphox
chespin
quilladin
chesnaught
piplup
prinplup
empoleon
pawmi
pawmo
pawmot
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
scratch
flame-charge
psyshock
vine-whip
bite
seed-bomb
pound
bubble-beam
drill-peck
thunder-shock
spark
discharge
true.
```

```
?- display_cen_attacks.
gnaw
leech-seed
gnaw
scratch
confuse-ray
water-gun
bubble
slap
scratch
vine-whip
pound
thunder-shock
true .

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

?- indicate_attack(pawmo).
pawmo-->spark
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
fennekin->scratch
braixen->flame-charge
delphox->psyshock
chespin->vine-whip
quilladin->bite
chesnaught->seed-bomb
piplup->pound
prinplup->bubble-beam
empoleon->drill-peck
pawmi->thunder-shock
pawmo->spark
pawmot->discharge
true.

?- powerful(Name).
Name = raichu ;
Name = venusaur ;
Name = butterfree ;
Name = charizard ;
Name = ninetails ;
Name = wartortle ;
Name = blastoise ;
Name = delphox ;
Name = quilladin ;
Name = chesnaught ;
Name = prinplup ;
Name = empoleon ;
Name = pawmo ;
Name = pawmot.

?-
```

```
?- 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 = delphox ;
Name = quilladin ;
Name = chesnaught ;
Name = prinplup ;
Name = empoleon ;
Name = pawmo ;
Name = pawmot.

?- type(prinplup,Type).
Type = water.
```

```
?-  type(Name,fire),write(Name),nl,fail.
charmander
charmeleon
charizard
vulpix
ninetails
fennekin
braixen
delphox
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(50),attack(pound,40))
pokemon(name(prinplup),water,hp(65),attack(bubble-beam,65))
pokemon(name(empoleon),water,hp(85),attack(drill-peck,80))
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(chespin),grass,hp(60),attack(vine-whip,45))
pokemon(name(quilladin),grass,hp(60),attack(bite,60))
pokemon(name(chesnaught),grass,hp(90),attack(seed-bomb,80))
true.

?- family(piplup).
piplup prinplup empoleon
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
fennekin braixen delphox
chespin quilladin chesnaught
piplup prinplup empoleon
pawmi pawmo pawmot
true.

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

?- lineage(quilladin).
pokemon(name(quilladin)),grass,hp(60),attack(bite,60))
pokemon(name(chesnaught)),grass,hp(90),attack(seed-bomb,80))
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
```

Task 2 – List Processing:

Head/Tail Exercises:

#1: H = red, T = [yellow, blue, green].

#2: false.

#3: F = red.

#4: S = yellow.

#5: F = red, S = yellow, R = [blue, green].

#6: List = [this, and, that].

#7: List = [this, and, that].

#8: false.

#9: true.

#10: Row = Column, Column = 1, Rest = [cell(3, 2), cell(1, 3)].

#11: X = one(un, uno), Y = [two(dos, deux), three(trois, tres)].

List Processing Code:

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

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

% --- File: list_processors.pro

% --- Line: Some list processing functions

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

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

% --- Facts ...
```

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

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


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

% --- first(L, X) :: L is a list, X is the variable being assigned


first([H|_], H).



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

% --- rest(L, X) :: L is a list, X is the variable being assigned


rest([_|T], T).



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

% --- last(L, X) :: L is a list, X is the variable being assigned


last([H|[]], H).

last([_|T], Result) :- last(T, Result).



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

% --- nth(N, L, X) :: N is a number, L is a list, X is the variable being assigned
```

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


% ----------------------------------------------------------------------
% --- writelist(L) :: L is a list

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


% ----------------------------------------------------------------------
% --- sum(L, X) :: L is a list, X is the variable being assigned

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


% ----------------------------------------------------------------------
% --- add_first(E, L, X) :: E is an element, L is a list, X is the variable being assigned
```

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



% ---------------------------------------------------------------------
% --- add_last(E, L, X) :: E is an element, L is a list, X is the variable being assigned

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



% ---------------------------------------------------------------------
% --- iota(N, X) :: N is a number,  X is the variable being assigned
iota(0,[]).
iota(N,IotaN) :-
K is N - 1,
iota(K,IotaK),
add_last(N,IotaK,IotaN).



% ---------------------------------------------------------------------
% --- pick(L, X) :: L is a list, X is the variable being assigned

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

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


% ------------------------------------------------------------------------
% --- make_set(L, X) :: L is a list, X is the variable being assigned

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(L, X) :: L is a list, X is the variable being assigned

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

```
% ------------------------------------------------------------------------
% --- factorial(N, X) :: N is a number, X is the variable being assigned

factorial(N, X) :-
iota(N,K),
product(K,X).

% ------------------------------------------------------------------------
% --- make_list(N, E, L) :: N is a number, E is an element, L is the list being returned

make_list(0,_,[]).
make_list(N1,E1,L1) :-
K is N1 - 1,
make_list(K,E1,Lk),
add_last(E1,Lk,L1).


% ------------------------------------------------------------------------
% --- but_first(L, X) :: L is a list, X is the variable being assigned

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

```
% ---------------------------------------------------------------------
% --- but_last(L, X) :: L is a list, X is the variable being assigned

but_last([], []).
but_last([_], []).
but_last([H|T], [H|Result]) :-
but_last(T, Result).

% ---------------------------------------------------------------------
% --- is_palindrome(L) :: L is a list


is_palindrome([]).
is_palindrome([_]).
is_palindrome([Head|Tail]) :-
but_last(Tail, Middle),
last(Tail, Last),
Head = Last,
is_palindrome(Middle).


% ---------------------------------------------------------------------
% --- noun_phrase(X) :: X is the variable being assigned
```

```
adjectives([happy, sad, funny, charming, tall, short, brave, smart]).

nouns([cat, dog, horse, car, book, tree, chair, table]).


noun_phrase([the|T]) :-

adjectives(AdjectiveList),

nouns(NounList),

pick(AdjectiveList, Adjective),

pick(NounList, Noun),

append([Adjective, Noun], [], T).


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

% --- sentence(X) :: X is the variable being assigned


past_tense_verbs([hit, jumped, killed, healed, chased, hugged, scolded]).


sentence(L) :-

noun_phrase(FirstPhrase),

noun_phrase(SecondPhrase),

past_tense_verbs(PastTenseVerbs),

pick(PastTenseVerbs, PastTenseVerb),

make_list(1, PastTenseVerb, PastTenseVerbList),

append(FirstPhrase,PastTenseVerbList,TempList),

append(TempList,SecondPhrase,L).
```

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 ,

?-  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 = peach ,

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

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

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

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

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

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

?- 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, charming, book] ;

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

?- noun_phrase(NP).
NP = [the, short, table] ;

?- noun_phrase(NP).
NP = [the, short, dog] ;

?- noun_phrase(NP).
NP = [the, brave, dog] ;

?- sentence(S).
S = [the, short, book, healed, the, happy, table] ;

?- sentence(S).
S = [the, brave, tree, hit, the, short, chair] ;

?- sentence(S).
S = [the, short, horse, healed, the, smart, tree] ;

?- sentence(S).
S = [the, charming, chair, jumped, the, charming, book] ;

?- sentence(S).
S = [the, brave, car, healed, the, sad, tree] ;

?- sentence(S).
S = [the, short, tree, jumped, the, brave, dog] ;

?- sentence(S).
S = [the, brave, book, healed, the, funny, chair] ;

?- sentence(S).
S = [the, brave, tree, chased, the, happy, horse] ;

?- sentence(S).
S = [the, tall, table, jumped, the, brave, table] ;

?- sentence(S).
S = [the, brave, car, hugged, the, sad, dog] ;

?- sentence(S).
S = [the, smart, dog, jumped, the, brave, table] ;

?- sentence(S).
S = [the, smart, cat, killed, the, happy, car] ;

?- sentence(S).
S = [the, funny, dog, jumped, the, brave, horse] ;

?- sentence(S).
S = [the, tall, tree, healed, the, sad, tree] ;

?- sentence(S).
S = [the, sad, dog, jumped, the, funny, car] ;

?- sentence(S).
S = [the, funny, car, hugged, the, funny, book] ;

?-
```