Declan ONUNKWO 04/16/2023

CSC344 – Programming Languages

Assignment: **Prolog programming assignment 2 – A favorite Pokémon KB plus Simple List Processing**

LEARNING ABSTRACT

The project is aimed to further our knowledge on Prolog programming language and practice simple list processing exercises. A program was created using Pokémon characters and their abilities. When ran, the program that can answer certain queries with answers relating to what was written in the code. This project helped me to develop a deeper appreciation for the power and elegance of prolog, as well as to improve my problem-solving abilities and programming skills.

Task 1: Pokémon

Part 1: Initial Pokémon 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

```
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('./prolog/pokemon.pro').
true.

?- cen(pikachu).
true.

?- cen(raichu).
false.

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

?- cen(P), write(P), 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(name(N), T, _, _), write('nks(name('), write(N), write('), kind('), write(T), 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 ;
false.

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

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

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

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

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

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

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

Part 3: KB Extension

```
70 display_cen :- cen(P), write(P), nl, fail.
71 display_not_cen :- evolves(_,Q), write(Q), nl, fail.
72 generator(N, T) :- pokemon(name(N), T, _, _).
73 display_names :- pokemon(name(N), _, _, _), write(N), nl, fail.
74 display_attacks :- pokemon(_, _, _, attack(A, _)), write(A), nl, fail.
75 display_cen_attacks :- pokemon(name(N), _, _, attack(A, _)), cen(N), write(A), nl, fail.
76 indicate_attack(N) :- pokemon(name(N), _, _, attack(A, _)), write(N), write(' -> '), write(A), nl, fail.
77 indicate_attacks :- pokemon(name(N), _, _, attack(A, _)), write(N), write(' -> '), write(A), nl, fail.
78 powerful(N) :- pokemon(name(N), _, _, attack(_, D)), D > 55.
79 tough(N) :- pokemon(name(N), _, hp(H), _), H > 100.
80 awesome(N) :- pokemon(name(N), _, hp(H), attack(_, D)), D > 55, H > 100.
81 powerful_but_vulnerable(N) :- pokemon(name(N), _, hp(H), attack(_, D)), D > 55, H =< 100.
82 type(N, T) :- pokemon(name(N), T, _, _).
83 dump_kind(T) :- pokemon(name(N), T, hp(H), attack(A, D)), write(pokemon(name(N), T, hp(H), attack(A, D))), nl,fail.
84 family(A) :- write(A), evolves(A, B), write(' '), write(B), evolves(B, C), write(' '), write(C).
85 families :- cen(A), nl, write(A), evolves(A, B), write(' '), write(B), evolves(B, C), write(' '), write(C), fail.
86 lineage(N) :- pokemon(name(N), T, hp(H), attack(A, D)), write(pokemon(name(N), T, hp(H), attack(A, D))), evolves(N, M)
   , nl, pokemon(name(M), T1, hp(H1), attack(A1, D1)), write(pokemon(name(M), T1, hp(H1), attack(A1, D1))), evolves(M, L)
   , nl, pokemon(name(L), T2, hp(H2), attack(A2, D2)), write(pokemon(name(L), T2, hp(H2), attack(A2, D2))).
87
```

Part 4: Interaction demo with 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('./prolog/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 .

?- 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_atatck(charmander).
Correct to: "indicate_attack(charmander)"? yes
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_vunerable(Name).
Correct to: "powerful_but_vulnerable(Name)"? yes
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).
pikachu raichu
false.

?- 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
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 Pokémon

```
80 % ----------------------------------------------------------------------
81 % --- 12 ADDITIONAL POKEMONS
82 % ----------------------------------------------------------------------
83
84 cen(voltorb).
85 cen(ponyta).
86 cen(hoppip).
87 cen(froakie).
88
89 evolves(voltorb, electrode).
90 evolves(ponyta, rapidash).
91 evolves(hoppip, skiploom).
92 evolves(skiploom, jumpluff).
93 evolves(froakie, frogadier).
94 evolves(frogadier, greninja).
95
96 pokemon(name(voltorb), electric, hp(40), attack(spark, 20)).
97 pokemon(name(electrode), electric, hp(60), attack(discharge, 50)).
98 pokemon(name(electabuzz), electric, hp(65), attack(shock, 20)).
99
100 pokemon(name(ponyta), fire, hp(50), attack(ember, 25)).
101 pokemon(name(rapidash), fire, hp(65), attack(flare-blitz, 120)).
102 pokemon(name(slugma), fire, hp(40), attack(lava-plume, 40)).
103
104 pokemon(name(hoppip), grass, hp(35), attack(tackle, 35)).
105 pokemon(name(skiploom), grass, hp(55), attack(bullet-seed, 45)).
106 pokemon(name(jumpluff), grass, hp(75), attack(grass-knot, 55)).
107
108 pokemon(name(froakie), water, hp(41), attack(fling, 56)).
109 pokemon(name(frogadier), water, hp(54), attack(surf, 63)).
110 pokemon(name(greninja), water, hp(72), attack(waterfall, 95)).
111
112 % ----------------------------------------------------------------------
```

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

```
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('./prolog/pokemon.pro').
true.

?- display_cen.
pikachu
bulbasaur
caterpie
charmander
vulpix
poliwag
squirtle
staryu
voltorb
ponyta
hoppip
froakie
false.

?- display_not_cen.
raichu
ivysaur
venusaur
metapod
butterfree
charmeleon
charizard
ninetails
poliwhirl
poliwrath
wartortle
blastoise
starmie
electrode
rapidash
skiploom
jumpluff
frogadier
greninja
false.

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

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

?- generator(Name, electric).
Name = pikachu ;
Name = raichu ;
Name = voltorb ;
Name = electrode ;
Name = electabuzz.

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

?- display_names.
pikachu
raichu
bulbasaur
ivysaur
venusaur
caterpie
metapod
butterfree
charmander
charmeleon
charizard
vulpix
ninetails
poliwag
poliwhirl
poliwrath
squirtle
wartortle
blastoise
staryu
starmie
voltorb
electrode
electabuzz
ponyta
rapidash
slugma
hoppip
skiploom
jumpluff
froakie
frogadier
greninja
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
spark
discharge
shock
ember
flare-blitz
lava-plume
tackle
bullet-seed
grass-knot
fling
surf
waterfall
false.

?- display_cen_attacks.
gnaw
leech-seed
gnaw
scratch
confuse-ray
water-gun
bubble
slap
spark
ember
tackle
fling
false.

?- indicate_attack(froakie).
froakie -> fling
false.

?- indicate_attack(hoppip).
hoppip -> tackle
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
voltorb -> spark
electrode -> discharge
electabuzz -> shock
ponyta -> ember
rapidash -> flare-blitz
slugma -> lava-plume
hoppip -> tackle
skiploom -> bullet-seed
jumpluff -> grass-knot
froakie -> fling
frogadier -> surf
greninja -> waterfall
false.

?- powerful(Name).
Name = raichu ;
Name = venusaur ;
Name = butterfree ;
Name = charizard ;
Name = ninetails ;
Name = wartortle ;
Name = blastoise ;
Name = rapidash ;
Name = froakie ;
Name = frogadier ;
Name = greninja.

?- 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 = rapidash ;
Name = froakie ;
Name = frogadier ;
Name = greninja.

?- type(rapidash, Type).
Type = fire.

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

?- type(Name,fire),write(Name),nl,fail.
charmander
charmeleon
charizard
vulpix
ninetails
ponyta
rapidash
slugma
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(froakie),water,hp(41),attack(fling,56))
pokemon(name(frogadier),water,hp(54),attack(surf,63))
pokemon(name(greninja),water,hp(72),attack(waterfall,95))
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(hoppip),grass,hp(35),attack(tackle,35))
pokemon(name(skiploom),grass,hp(55),attack(bullet-seed,45))
pokemon(name(jumpluff),grass,hp(75),attack(grass-knot,55))
false.

?- family(voltorb).
voltorb electrode
false.

?- family(ponyta).
ponyta rapidash
false.

?- family(froakie).
froakie frogadier greninja
true.
```

```
?- families.

pikachu raichu
bulbasaur ivysaur venusaur
caterpie metapod butterfree
charmander charmeleon charizard
vulpix ninetails
poliwag poliwhirl poliwrath
squirtle wartortle blastoise
staryu starmie
voltorb electrode
ponyta rapidash
hoppip skiploom jumpluff
froakie frogadier greninja
false.

?- lineage(voltorb).
pokemon(name(voltorb),electric,hp(40),attack(spark,20))
pokemon(name(electrode),electric,hp(60),attack(discharge,50))
false.

?- lineage(ponyta).
pokemon(name(ponyta),fire,hp(50),attack(ember,25))
pokemon(name(rapidash),fire,hp(65),attack(flare-blitz,120))
false.

?- lineage(hoppip).
pokemon(name(hoppip),grass,hp(35),attack(tackle,35))
pokemon(name(skiploom),grass,hp(55),attack(bullet-seed,45))
pokemon(name(jumpluff),grass,hp(75),attack(grass-knot,55))
true.

?- lineage(froakie).
pokemon(name(froakie),water,hp(41),attack(fling,56))
pokemon(name(frogadier),water,hp(54),attack(surf,63))
pokemon(name(greninja),water,hp(72),attack(waterfall,95))
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].
false.

?- [_|[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

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

rest([_|T], T).

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

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

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

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

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

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

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

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

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

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

make_list(0,_,[]).
make_list(N,DT,RL) :-
        K is N - 1,
        make_list(K,DT,NRL),
        add_last(DT,NRL,RL).

but_first([_|Tail], Tail).
but_last([Head|Tail], Result) :- reverse([Head|Tail], ReversedList), but_first(ReversedList, NewList), reverse(NewList, Result).

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

noun_phrase([the,A,N]) :- pick([angry, happy, shocked, hungry, excited, silly, confident], A), pick([robot, student, artist, dog, baby, professor, athlete], N).

sentence(S) :- pick([loved, trusted, watched, cried, followed, defeated, shouted, played], V), noun_phrase(P1), noun_phrase(P2), add_last(V,P1,PWV), append(PWV, P2, S).
```

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('./prolog/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 ;
false.

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

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

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

?- sum([2,3,4,57,11],SumOfPrimes).
SumOfPrimes = 77.

?- sum([2,3,4,7,11],SumOfPrimes).
SumOfPrimes = 27.

?- 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] ;
false.

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

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

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

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

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

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

?- 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('./prolog/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(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([1,2,3,4,5,6,2,3,1]).
false.

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

?- noun_phrase(NP).
NP = [the, shocked, student] .

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

?- noun_phrase(NP).
NP = [the, happy, robot] .

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

?- noun_phrase(NP).
NP = [the, hungry, athlete] .
```

```
?- sentence(S).
S = [the, silly, baby, trusted, the, angry, student] .

?- sentence(S).
S = [the, confident, dog, defeated, the, silly, robot] .

?- sentence(S).
S = [the, silly, robot, followed, the, excited, athlete] .

?- sentence(S).
S = [the, hungry, baby, followed, the, happy, student] .

?- sentence(S).
S = [the, confident, dog, watched, the, silly, robot] .

?- sentence(S).
S = [the, happy, dog, watched, the, hungry, athlete] .

?- sentence(S).
S = [the, shocked, professor, shouted, the, happy, athlete] .

?- sentence(S).
S = [the, shocked, student, defeated, the, happy, dog] .

?- sentence(S).
S = [the, confident, professor, shouted, the, shocked, athlete] .

?- sentence(S).
S = [the, happy, artist, loved, the, happy, artist] .

?- sentence(S).
S = [the, silly, baby, followed, the, excited, dog] .

?- sentence(S).
S = [the, happy, artist, cried, the, silly, dog] .

?- sentence(S).
S = [the, hungry, athlete, followed, the, confident, robot] .

?- sentence(S).
S = [the, happy, athlete, trusted, the, shocked, robot] .

?- sentence(S).
S = [the, angry, student, defeated, the, excited, dog] .

?- sentence(S).
S = [the, hungry, athlete, played, the, excited, baby] .
```