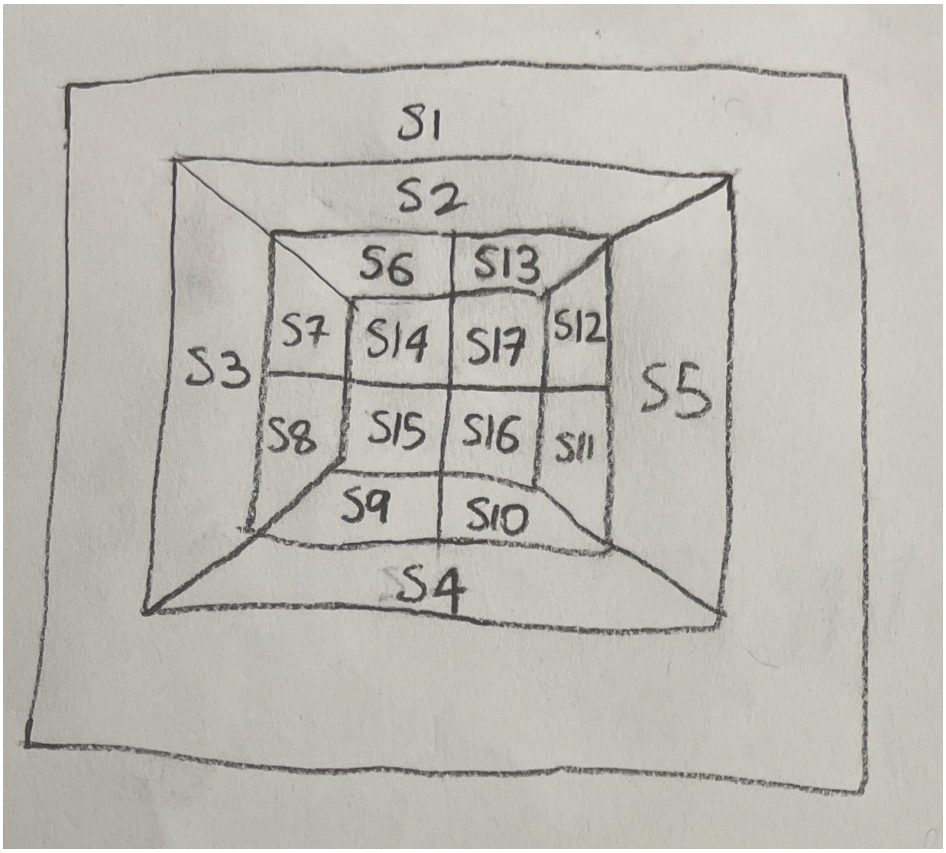
Prolog Programming Assignment #1: Various Computations

Learning Abstract

For this assignment, it is my first Prolog Programming Assignment which consist of four tasks. The first task is to create a map color program that is based on the map coloring program from the third Prolog Lesson. The second task is to create a shapes world program from the fourth Prolog Lesson. The third task is to create and extend a Prolog program based on Pokemon trading cards from the document that the professor provided. The fourth task is to create a Head/ Tail Referencing Prolog program and list processors Prolog Program from the fifth Prolog Lesson as well as creating an another list processors Prolog Program.

Task 1: Map Coloring

Image of Map with the regions labelled



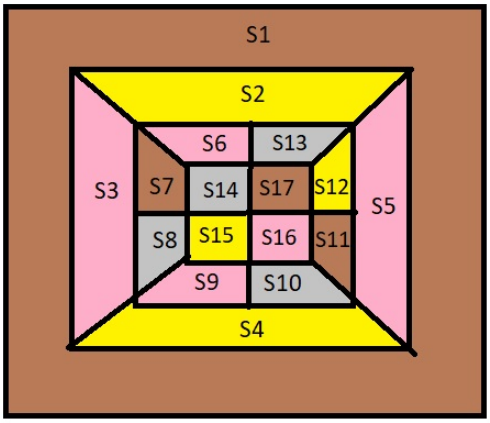
Prolog Code based on Map Coloring

```
different(brown,yellow).
different(brown,pink).
different(brown,grey).
different(pink,grey).
different(pink,brown).
different(pink,yellow).
different(yellow,pink).
different(yellow,grey).
different(yellow,brown).
different(grey,yellow).
different(grey,pink).
different(grey,brown).

coloring(S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,S11,S12,S13,S14,S15,S16,S17):-

    different(S1,S2),
    different(S1,S3),
    different(S1,S4),
    different(S1,S5),
    different(S2,S3),
    different(S2,S5),
    different(S2,S6),
    different(S2,S13),
    different(S3,S4),
    different(S3,S8),
    different(S4,S5),
    different(S4,S9),
    different(S4,S10),
    different(S5,S11),
    different(S5,S12),
    different(S6,S7),
    different(S6,S13),
    different(S6,S14),
    different(S7,S14),
    different(S7,S8),
    different(S8,S15),
    different(S9,S8),
    different(S9,S10),
    different(S9,S15),
    different(S10,S11),
    different(S10,S16),
    different(S11,S16),
    different(S11,S12),
    different(S12,S13),
    different(S12,S17),
    different(S13,S17),
    different(S14,S15),
    different(S14,S17),
    different(S15,S16),
    different(S16,S17).
```

Demo based on Map Coloring

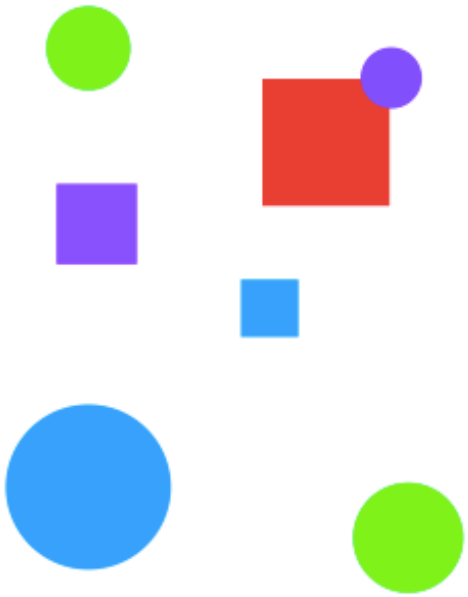
```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.5.20-DIRTY)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

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


% /Users/lamin/Documents/CSC344/map_coloring.pl compiled 0.00 sec, 13 clauses
?- coloring([S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,S11,S12,S13,S14,S15,S16,S17]).
S1 = S7, S7 = S11, S11 = S17, S17 = brown,
S2 = S4, S4 = S12, S12 = S15, S15 = yellow,
S3 = S5, S5 = S6, S6 = S9, S9 = S16, S16 = pink,
S8 = S10, S10 = S13, S13 = S14, S14 = grey .

?-
```

Image of colored Map based on Map Coloring



Task 2: The Floating Shapes World

Image of The Floating Shapes World



The Prolog KB

```
square(sera,side(7),color(purple)).
square(sara,side(5),color(blue)).
square(sarah,side(11),color(red)).

circle(carla,radius(4),color(green)).
circle(cora,radius(7),color(blue)).
circle(connie,radius(3),color(purple)).
circle(claire,radius(5),color(green)).

circles :- circle(Name,_,_), write(Name),nl,fail.
circles.

squares :- square(Name,_,_), write(Name),nl,fail.
squares.

shapes :- circles,squares.

blue(Name) :- square(Name,_,color(blue)).
blue(Name) :- circle(Name,_,color(blue)).

large(Name) :- area(Name,A), A >= 100.
```

```
small(Name) :- area(Name,A), A < 100.

area(Name,A) :- circle(Name,radius(R),_), A is 3.14 * R * R.
area(Name,A) :- square(Name,side(S),_), A is S * S.
```

Demo of The Prolog KB

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.5.20-DIRTY)
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).


% /Users/lamin/Documents/CSC344/shapes_world_1.pl compiled 0.00 sec,
18 clauses
?- listing(squares).
squares :-
    square(Name, _, _),
    write(Name),
    nl,
    fail.
squares.

true.

?- squares.
sera
sara
sarah
true.

?- listing(circles).
circles :-
    circle(Name, _, _),
    write(Name),
    nl,
    fail.
circles.

true.

?- circles.
carla
cora
connie
claire
true.

?- listing(shapes).
shapes :-
```

```
    circles,
    squares.

true.

?- shapes.
carla
cora
connie
claire
sera
sara
sarah
true.

?- blue(Shape).
Shape = sara ;
Shape = cora.

?- large(Name),write(Name),nl,fail.
cora
sarah
false.

?- small(Name),write(Name),nl,fail.
carla
connie
claire
sera
sara
false.

?- area(cora,A).
A = 153.86 .

?- area(carla,A).
A = 50.24 .

?-
```

Task 3: Pokemon KB Interaction and Programming

Part 1: Queries

Query 1: Is pikachu a “creatio ex nihilo” (created out of nothing) Pokemon?

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

?-
```

Query 2: Is raichu a “creatio ex nihilo” pokemon?

```
?- cen(raichu).
false.

?-
```

Query 3: By means of hand intervention, list all of the “creatio ex nihilo” pokemon.

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

?-
```

Query 4: By means of the standard idiom of repetition, list all of the “creatio ex nihilo” pokemon.

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

```
?-
```

Query 5: Does squirtle evolve into wartortle?

```
?- evolves(squirtle,wartortle).
true.
```

Query 6: Does wartortle evolve into squirtle?

```
?- evolves(wartortle,squirtle).
false.

?-
```

Query 7: Does squirtle evolve into blastoise?

```
?- evolves(squirtle,blastoise).
false.

?-
```

Query 8: By means of hand intervention, list all triples of pokemon such that the first evolves into the second and the second evolves into the third.

```
?- 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.

?-
```

Query 9: By means of the standard idiom of repetition, list all pairs of pokemon such that the first evolves through an intermediary to the second - placing an arrow between each pair.

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

?-
```

Query 10: By means of the standard idiom of repetition, list the names of all of the pokemon.

```
?- 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.

?-
```

Query 11: By means of the standard idiom of repetition, list the names of all of the fire pokemon.

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

?-
```

Query 12: By means of the standard idiom of repetition, provide a summary of each pokemon and its kind, representing each pairing of name and kind in the manner suggested by the redacted demo.

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

?-
```

Query 13: What is the name of the pokemon with the waterfall attack?

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

?-
```

Query 14: What is the name of the pokemon with the poison-powder attack?

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

?-
```

Query 15: By means of the standard idiom of repetition, list the names of the attacks of all of the water pokemon.

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

?-
```

Query 16: How much damage (hp count) can poliwhirl absorb?

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

?-
```

Query 17: How much damage (hp count) can butterfree absorb?

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

?-
```

Query 18: By means of the standard idiom of repetition, list the names of all of the pokemon that can absorb more than 85 units of damage.

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

```
?-
```

Query 19: By means of the standard idiom of repetition, list the names of all of the pokemon that can dish out more than 60 units of damage with one instance of their attack.

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

?-
```

Query 20: By means of the standard idiom of repetition, list the names and the hit point value for each of the “creation ex nihilo” pokemon, with the results formatted as the redacted demo suggests.

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

Extended Prolog Code of 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)).

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

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

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

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

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

dump_kind(Element) :- pokemon(Name,Element,Hp,Attack),
write(pokemon(Name,Element,Hp,Attack)),nl,false.

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

family(Name) :- evolves(Name,X), write(Name), write(" "),
write(X), evolves(X,Y),write(" "), write(Y).

families :- cen(Name), evolves(Name,X), nl, write(Name), write("
"), write(X), evolves(X,Y),write(" "), write(Y),fail.

lineage(Name) :- pokemon(name(Name),Type,hp(H),attack(Atk,Dmg)),
write(pokemon(name(Name),Type,hp(H),attack(Attack,Damage))),nl,

evolves(Name,X), pokemon(name(X),Type2,hp(I),attack(Atk2,Dmg2)),
write(pokemon(name(Y),Type2,hp(I),attack(Atk2,Dmg2))),nl,

evolves(X,Y), pokemon(name(Y),Type3,hp(J),attack(Atk3,Damage3)),
write(pokemon(name(Y),Type3,hp(J),attack(Atk3,Dmg3))).
```

Part 2: Programs

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.5.20-DIRTY)
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).


% /Users/lamin/Documents/CSC344/pokemon.pl compiled 0.00 sec, 51
clauses
?- 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.

?- powerful(pikachu).
false.
```

```
?- powerful(blastoise).
true.

?- powerful(X), write(X), nl, fail.
raichu
venusaur
butterfree
charizard
ninetails
wartortle
blastoise
false.

?- tough(raichu).
false.

?- tough(venusaur).
true.

?- tough(Name), write(Name), nl, fail.
venusaur
butterfree
charizard
poliwrath
blastoise
false.

?- type(caterpie,grass).
true .

?- type(pikachu,water).
false.

?- type(N,electric).
N = pikachu ;
N = raichu.

?- type(N,water), write(N), nl, fail.
poliwag
poliwhirl
poliwrath
squirtle
wartortle
blastoise
staryu
```

```
starmie
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(fire).
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))
false.

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

?- family(pikachu).
pikachu raichu
false.

?- family(squirtle).
squirtle wartortle blastoise
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(caterpie).
pokemon(name(caterpie),grass,hp(50),attack(gnaw,20))
pokemon(name(_53538),grass,hp(70),attack(stun-spore,20))
pokemon(name(butterfree),grass,hp(130),attack(whirlwind,80))
true.

?- lineage(metapod).
pokemon(name(metapod),grass,hp(70),attack(stun-spore,20))
pokemon(name(_55062),grass,hp(130),attack(whirlwind,80))
false.

?- lineage(butterfree).
pokemon(name(butterfree),grass,hp(130),attack(whirlwind,80))
false.
```

Task 4: Lisp Processing in Prolog

Head/Tail Referencing Exercises

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.5.20-DIRTY)
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)].

?-
```

Prolog Code based on Example List Processors

```
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).
```

Demo based on Example List Processors

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.5.20-DIRTY)
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).


% /Users/lamin/Documents/CSC344/list_processors.pl compiled 0.00
sec, -4 clauses
?- 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 = apple .

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

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

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

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

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

?- 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] .

?-
```

Prolog Code based on Specifications of List Processing Exercises

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

last([H|[]], H).
last([_|T], Result) :- last(T, Result).
iota(0,[]).

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

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

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

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

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

```
nth(RN,L,Item).

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

factorial(0,0).
factorial(X,Y) :- iota(X,Iota), product(Iota,Product), Y is
Product.

make_list(0,_,[]).
make_list(Number,Anything,Name) :-
 X is Number - 1,
 make_list(X,Anything,EleK),
 add_last(Anything,EleK,Name).

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

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

is_palindrome([]).
is_palindrome([_]).
is_palindrome(Index) :-
 first(Index,First), last(Index,Last),
 First = Last,
 but_first(Index,X), but_last(X,Y),
 is_palindrome(Y).

noun_phrase(Name) :-
 pick([adorable, angry, adventurous, doubtful, powerful,
thoughtful],Adjective),
 pick([airport, pizza, island, dog, cup, ghost, image,
napkin],Noun),
 add_last(Adjective,[the],First), add_last(Noun,First,Name).

sentence(Name) :-
 noun_phrase(A), noun_phrase(B),
 pick([ate, flew, wrote, built, touched, became, saw],Verb),
 add_last(Verb,A,X),
 append(X,B,Name)
```

Demo based on Specifications of List Processing Exercises

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.5.20-DIRTY)
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).


% /Users/lamin/Documents/CSC344/list_processors_2.pl compiled
0.00 sec, 23 clauses
?- 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, doubtful, airport] .

?- noun_phrase(NP).
NP = [the, doubtful, napkin] .

?- noun_phrase(NP).
NP = [the, angry, island] .

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

?- noun_phrase(NP).
NP = [the, doubtful, island] .

?- sentence(S).
S = [the, adorable, dog, wrote, the, powerful, image] .

?- sentence(S).
S = [the, powerful, island, touched, the, adventurous, ghost] .

?- sentence(S).
S = [the, adorable, image, ate, the, adventurous, napkin] .

?- sentence(S).
S = [the, adorable, image, wrote, the, adventurous, pizza] .

?- sentence(S).
S = [the, thoughtful, cup, flew, the, thoughtful, pizza] .

?- sentence(S).
S = [the, angry, cup, became, the, adorable, napkin] .

?- sentence(S).
S = [the, adorable, airport, built, the, adventurous, cup] .

?- sentence(S).
S = [the, doubtful, cup, touched, the, thoughtful, airport] .

?- sentence(S).
S = [the, adventurous, napkin, wrote, the, doubtful, airport] .
```

```
?- sentence(S).
S = [the, angry, pizza, ate, the, powerful, image] .

?- sentence(S).
S = [the, powerful, cup, flew, the, adventurous, napkin] .

?- sentence(S).
S = [the, adventurous, pizza, became, the, adorable, ghost] .

?- sentence(S).
S = [the, doubtful, image, built, the, thoughtful, napkin] .

?- sentence(S).
S = [the, thoughtful, airport, touched, the, doubtful,
airport] .

?- sentence(S).
S = [the, adorable, airport, became, the, doubtful, island] .

?- sentence(S).
S = [the, powerful, dog, built, the, powerful, pizza] .

?-
```