==========

==========

Prolog Assignment 1

Matilda Knapp

4/8/2022

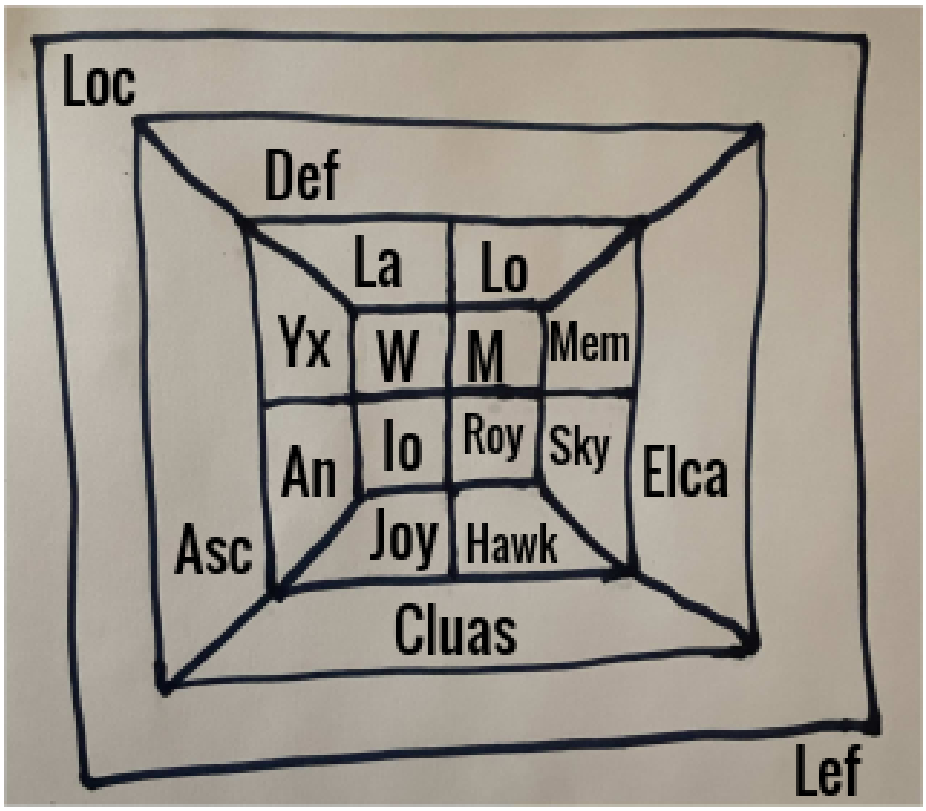
==========

Abstract:

This is the first Prolog Assignment for 344. There are 4 tasks with multiple parts. The first is coloring a map with Prolog, the second is messing with Shapes World. The third is pokemon based definitions, then lastly list processing exercises.

Task One Map Coloring:

Original map:



Source code (Map Of Oshen):

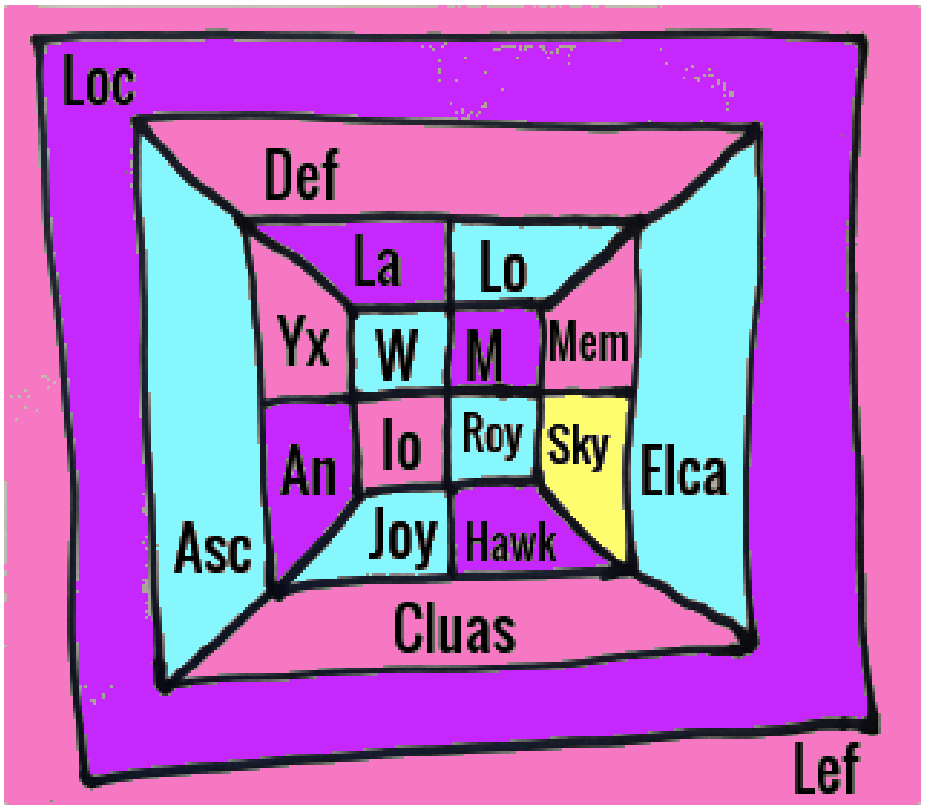
```
% ------------------------------------------------------------------------
% File: mymapcolor.pl
% Line: Program to color map 4 colors, I have named this map, Map Of Oshen.
% More: The colors I am using are pink, purple, blue, yellow.
% More: The abbreviations represent the countries within Oshen.
% ------------------------------------------------------------------------
% different(X,Y) :: X is not equal to Y
different(pink,purple).
different(pink,blue).
different(pink,yellow).
different(purple,pink).
different(purple,blue).
different(purple,yellow).
different(blue,pink).
different(blue,purple).
different(blue,yellow).
different(yellow,pink).
different(yellow,purple).
different(yellow,blue).
% ------------------------------------------------------------------------
% coloring(LOC,DEF,ASC,CLUAS,LEF,ELCA,JOY,AN,YX,LA,LO,W,M,MEM,SKY,HAWK,JOY,IO,ROY)
% :: Oshen
% Oshen countries represented by standard abbreviation are colored.
% so that none of the countries sharing a border are the same color.
coloring(LOC,DEF,ASC,CLUAS,LEF,ELCA,JOY,AN,YX,LA,LO,W,M,MEM,SKY,HAWK,JOY,IO,ROY):-
  different(LEF, LOC),
  different(LOC, DEF),
  different(LOC, ASC),
  different(LOC, CLUAS),
  different(LOC, ELCA),
  different(DEF, LA),
  different(DEF, LO),
  different(DEF, ASC),
  different(ASC, YX),
```

```
different(ASC, AN),
different(ASC, CLUAS),
different(CLUAS, JOY),
different(CLUAS, HAWK),
different(CLUAS, ELCA),
different(ELCA, MEM),
different(ELCA, SKY),
different(ELCA, DEF),
different(LA, LO),
different(LA, W),
different(LA, YX),
different(LO, MEM),
different(LO, M),
different(YX, AN),
different(YX, W),
different(AN, YX),
different(AN, JOY),
different(AN, IO),
different(JOY, IO),
different(JOY, HAWK),
different(HAWK, ROY),
different(HAWK, SKY),
different(SKY, ROY),
different(SKY, MEM),
different(MEM, M),
different(W, IO),
different(W, M),
different(M, ROY),
different(IO, ROY).
```

Demo of map coloring code:

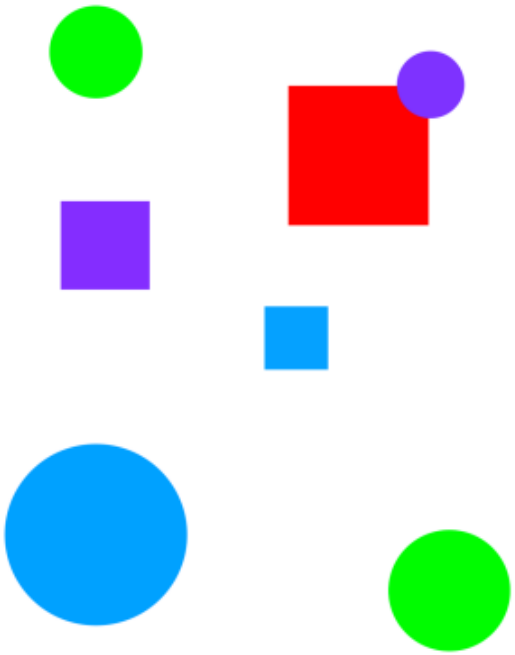
```
% c:/Users/roses/Desktop/Prolog Set/mymapcolor.pl compiled 0.00 sec, 13 clauses
?- coloring(LOC,DEF,ASC,CLUAS,LEF,ELCA,JOY,AN,YX,LA,LO,W,M,MEM,SKY,HAWK,JOY,IO,ROY).
LOC = AN, AN = LA, LA = M, M = HAWK, HAWK = purple,
DEF = CLUAS, CLUAS = LEF, LEF = YX, YX = MEM, MEM = IO, IO = pink,
ASC = ELCA, ELCA = JOY, JOY = LO, LO = W, W = ROY, ROY = blue,
SKY = yellow
```

Map as colored by code:



Floating Shape World:

Image of (Shape World):



Source code (Shape World):

```
% File: shapes_world_1.pl
% Line: Loosely represents a 2-D shape world.
% Facts: square(N,side(L),color(C)) and color C.

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

% Note: circle(N,radius(R),color(C)) :: N is the name of a circle with
% Note: radius R and color C.

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

% Rules: circles :: list the names of all the circles.

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

```
% Rules: squares :: list the names of all of the squares.

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

% Rules: squares :: list the names of all of the shapes.

   shapes :- circles,squares.

% Rules: blue(Name) :: Name is a blue shape.

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

% Rules: large(Name) :: Name is a large shape.

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

% Rules: small(Name) :: Name is a small shape.

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

% Rules: area(Name,A) :: A is the area of the shape with name Name.
   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 (Shape World):

```
% c:/Users/roses/Desktop/Prolog Set/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 Three Pokemon:

Part One Demo Pokemon:

```
% c:/Users/roses/Desktop/Prolog Set/pokemon.pl compiled 0.00 sec, 42 clauses
?- 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.


?- evoles(squirtle,wartortle).
Correct to: "evolves(squirtle,wartortle)"?
Please answer 'y' or 'n'? yes
true.

?- evolves(wartortle,squirtle).
false.


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

```
?- evolves(A,B), evolves(B,C).
A = bulbasaur,
B = ivysaur,
C = venusaur ;
A = caterpie,
B = metapod,
C = butterfree ;
A = charmander,
B = charmeleon,
C = charizard ;
A = poliwag,
B = poliwhirl,
C = poliwrath ;
A = squirtle,
B = wartortle,
C = blastoise ;
false.


?- evolves(A,B),evolves(B,C),write(A),write(-->),write(C),nl,fail.
bulbasaur-->venusaur
caterpie-->butterfree
charmander-->charizard
poliwag-->poliwrath
squirtle-->blastoise
false.

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

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

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

?- pokemon(_,water,_,attack(Act,_)),write(Act),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(Name),_,hp(H),_), H > 85, write(Name), nl, fail.
raichu
venusaur
butterfree
charizard
ninetails
poliwrath
blastoise
false.
```

(Important Note) You have a typo here in the instructions.

```
?- pokemon(_,_,_,attack(Act,Hp)),Hp > 60,write(Act),nl,fail.
thunder-shock
poison-powder
whirlwind
royal-blaze
fire-blast
false.

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

Extended Code Pokemon:

```
% ------------------------------------------------------------------------
% --- display_names :: list the names of all pokemon
    display_names :- pokemon(name(Name),_,_,_), write(Name),nl,fail.
    display_names.

% ------------------------------------------------------------------------
% --- display_attacks :: list the names of all attacks
    display_attacks :- pokemon(_,_,_,attack(Name,_)),write(Name),nl,fail.
    display_attacks.
```

```
% ------------------------------------------------------------------------
% --- Checks if pokemon is powerful
    powerful(Name) :- pokemon(name(Name),_,_,attack(_,Hp)), Hp > 55.

% ------------------------------------------------------------------------
% --- Checks if pokemon is tough
    tough(Name) :- pokemon(name(Name),_,hp(Hp),_), Hp >= 100.

% ------------------------------------------------------------------------
% --- Checks if pokemon is of a certain type
    type(Name,Type) :- pokemon(name(Name),Type,_,_).

% ------------------------------------------------------------------------
% --- Info dump based on type :: dump_kind
    dump_kind(Type) :- pokemon(name(Name),Type,hp(Hp),attack(Act,H)),
    write(pokemon(name(Name))),write(','),write(Type),write(','),
    write(hp(Hp)),write(','),write(attack(Act,H)),nl,fail.

% ------------------------------------------------------------------------
% --- display the names of all of the “creatio ex nihilo” pokemon
    display_cen :- pokemon(name(Name),_,_,_),cen(Name),write(Name),nl,fail.
    display_cen.

% ------------------------------------------------------------------------
% --- Display family of specified pokemon
    family(One) :- cen(One),evolves(One,Two),evolves(Two,Three),
    write(One),write(' '),write(Two),write(' '),write(Three).
    family(One) :- cen(One),evolves(One,Two),\+evolves(Two,_),
    write(One),write(' '),write(Two).

% ------------------------------------------------------------------------
% --- Display families of specified pokemon
    families :- cen(Name),family(Name),nl,fail.
    families.

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

```
% --- Display detailed lineage
    lineage(A) :-
    evolves(A,B),evolves(B,C),
    pokemon(name(A),Atype,hp(Ahp),attack(Atac,Acp)),
    write(name(A)),write(','),write(Atype),write(','),write(hp(Ahp)),
    write(','),write(attack(Atac,Acp)),nl,
    pokemon(name(B),Btype,hp(Bhp),attack(Btac,Bcp)),
    write(name(B)),write(','),write(Btype),write(','),write(hp(Bhp)),
    write(','),write(attack(Btac,Bcp)),nl,
    pokemon(name(C),Ctype,hp(Chp),attack(Ctac,Ccp)),
    write(name(C)),write(Ctype),write(hp(Chp)),write(attack(Ctac,Ccp)).
    lineage(A) :-
    evolves(A,B),\+evolves(B,_),
    pokemon(name(A),Atype,hp(Ahp),attack(Atac,Acp)),
    write(name(A)),write(','),write(Atype),write(','),write(hp(Ahp)),
    write(','),write(attack(Atac,Acp)),nl,
    pokemon(name(B),Btype,hp(Bhp),attack(Btac,Bcp)),
    write(name(B)),write(','),write(Btype),write(','),write(hp(Bhp)),
    write(','),write(attack(Btac,Bcp)).
    lineage(A) :-
    \+evolves(A,_),evolves(_,_),
    pokemon(name(A),Atype,hp(Ahp),attack(Atac,Acp)),
    write(name(A)),write(','),write(Atype),write(','),write(hp(Ahp)),
    write(','),write(attack(Atac,Acp)).
```

Part Two Demo Pokemon:

```
% c:/Users/roses/Desktop/Prolog Set/pokemon_plus.pl compiled 0.00 sec, 0 clauses
?- 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.

?- 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.
```

(Important Note) **You have another typo here in the instructions.**

```
?- tough(Name), write(Name), nl, fail.
venusaur
butterfree
charizard
ninetails
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
true.


?- family(pikachu).
pikachu raichu
true.

?- 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
true.
```

```
?- lineage(caterpie).
name(caterpie),grass,hp(50),attack(gnaw,20)
name(metapod),grass,hp(70),attack(stun-spore,20)
name(butterfree)grasshp(130)attack(whirlwind,80)
true .
```

```
?- lineage(metapod).
name(metapod),grass,hp(70),attack(stun-spore,20)
name(butterfree),grass,hp(130),attack(whirlwind,80)
true.
```

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

Task Four Head Tail List Processing:

Head Tail Referencing exercise:

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

I believe that H = red, and T = [yellow,blue,green], will be displayed.

(Demo)

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

(I got this one right)

///

?- [H, T] = [red, yellow, blue, green]. <redacted>

I believe that the list [red,yellow,blue,green] will be displayed.

(Demo)

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

(I got this one wrong)

///

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

I believe that F = red will be displayed.

(Demo)

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

(I got this correct)

///

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

I believe S = yellow will be displayed.

(Demo)

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

(I got this correct)

///

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

I believe that, F = red. will display, S = yellow. will display, and R = [blue,green]. will display.

(Demo)

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

(I got this correct)

///

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

I think that the list [this, and, that] will appear.

(Demo)

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

(I made a small mistake but I'm mostly correct)

///

?- List = [this, and, that]. <redacted>

I think List = [this, and that], will appear.

(Demo)

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

(I got this correct)

///

?- [a,[b, c]] = [a, b, c]. <redacted>

I think true will display.

(Demo)

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

(I got this wrong)

///

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

I think this will be true.

(Demo)

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

(I got this correct)

///

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

<redacted>

I think Row = 1, Column = 1, Rest = [cell(3,2), cell(1,3)].

(Demo)

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

(This was correct but I got the formatting a bit wrong)

///

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

I think X = one(un, uno), and Y = [two(dos, duex), three(trois, tres)] will display.

(Demo)

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

(This was correct)

Final Analysis: The comma seem to confuse me so I have to work on remembering that | and , are not the same.

List_Processor, Example List Processor, Demo:

```
% c:/Users/roses/Desktop/Prolog Set/list_processors.pl compiled 0.00 sec, 17 clauses
?- first([apple],First).
First = apple.

?- first([c,d,e,f,g,a,b,],P).
ERROR: Syntax error: Illegal start of term
ERROR: first([c,d,e,f,g,a,b
ERROR: ** here **
ERROR: ,],P) .
?- 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([peace],Last).
Last = peace ,


?- 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].



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

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

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

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

?- 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 = 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] ,
```

List_Processing Exercises Demo:

```
% c:/Users/roses/Desktop/Prolog Set/list_processors.pl compiled 0.00 sec, 0 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, fuzzy, kitten] .

?- noun_phrase(Np).
Np = [the, crabby, chair] .

?- noun_phrase(Np).
Np = [the, majestic, crab] .

?- noun_phrase(Np).
Np = [the, crabby, gutair] .

?- noun_phrase(Np).
Np = [the, aloof, star] .


?- sentence(S).
S = [the, aloof, computer, fought, the, crabby, crab] .

?- sentence(S).
S = [the, majestic, gutair, drew, the, handsome, corgi] .

?- sentence(S).
S = [the, shiny, kitten, won, the, majestic, kitten] .

?- sentence(S).
S = [the, majestic, sun, fought, the, shiny, gutair] .

?- sentence(S).
S = [the, handsome, kitten, drew, the, aloof, sun] .

?- sentence(S).
S = [the, shiny, corgi, drew, the, aloof, kitten] .

?- sentence(S).
S = [the, majestic, chair, awoken, the, majestic, kitten] .

?- sentence(S).
S = [the, majestic, sun, awoken, the, aloof, star] .

?- sentence(S).
S = [the, aloof, chair, drew, the, aloof, star] .
```

```
?- sentence(S).
S = [the, handsome, gutair, awoken, the, crabby, sun] .

?- sentence(S).
S = [the, majestic, star, fought, the, handsome, sun] .

?- sentence(S).
S = [the, aloof, computer, awoken, the, fuzzy, star] .

?- sentence(S).
S = [the, crabby, gutair, stopped, the, crabby, sun] .

?- sentence(S).
S = [the, crabby, chair, fought, the, aloof, crab] .

?- sentence(S).
S = [the, handsome, gutair, forgot, the, majestic, sun] .
```

Example List Processor Code:

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

make_list(0,_,[]).
make_list(N,Item,Ls) :-
   K is N - 1,
   make_list(K,Item,T),
   add_last(Item,T,Ls).

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

but_last([],[]).
but_last(Ls,Value) :-
   reverse(Ls,End),
   but_first(End,Top),
   reverse(Top,Value).

is_palindrome([]).
```

```
is_palindrome([_]).
is_palindrome(Ls) :-
   first(Ls,Top),
   last(Ls,Bottom),
   Top == Bottom,
   but_last(Ls,Newlast),
   but_first(Newlast,Next),
   is_palindrome(Next).

adjective([aloof,crabby,fuzzy,handsome,shiny,majestic]).
noun([sun,star,kitten,chair,gutair,crab,corgi,computer]).
ps([carried,stopped,won,fought,awoken,drew,forgot]).

noun_phrase([the,Adj,Noun]) :-
   adjective(Ls1),
   noun(Ls2),
   pick(Ls1,Adj),
   pick(Ls2,Noun).

sentence(Final) :-
noun_phrase(Np1),
ps(Ls),
pick(Ls,Ptv),
noun_phrase(Np2),
append(Np1,[Ptv],Next),
append(Next,Np2,Final).

% Note you never gave an example of factorial in the demo...

factorial(N,Final) :-
   iota(N,Ls),
   product(Ls,Final).

 % Turns out I spelled guitar wrong, oh well.
```