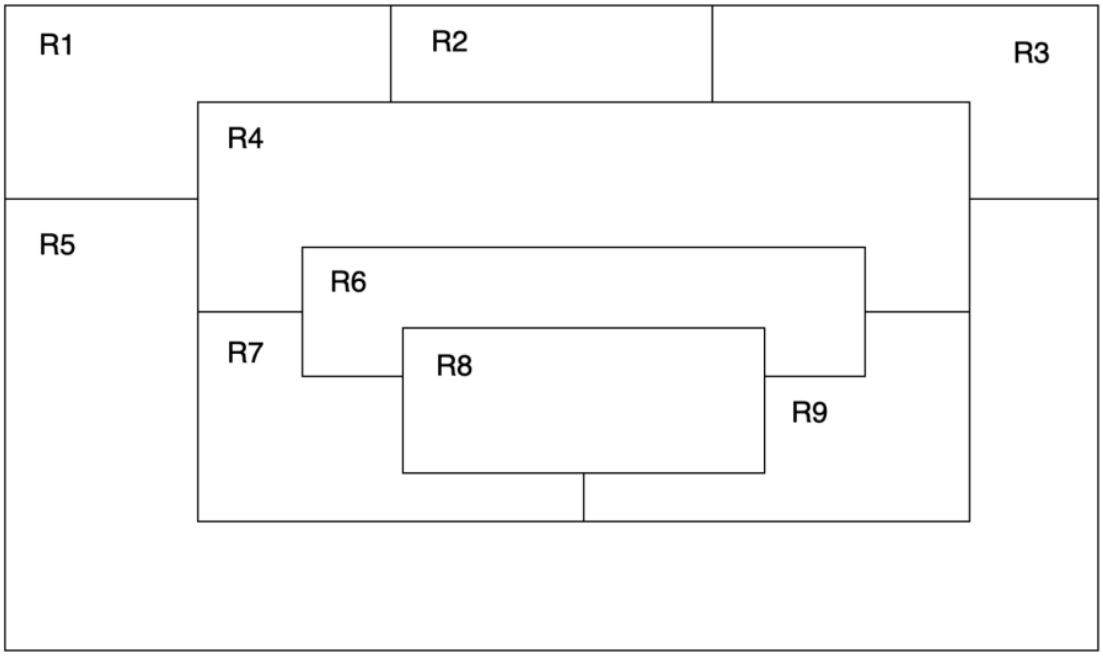
First Prolog Assignment

Learning Abstract : What I learned and found most interesting is the similarities between Racket and Prolog. The Head|Tail referencing is basically identical to rackets car and cdr. I knew how to do them right away without much explanation because it already made so much sense! I think pattern matching is pretty cool too. Its basically written down in english, what you want to show and what you dont.

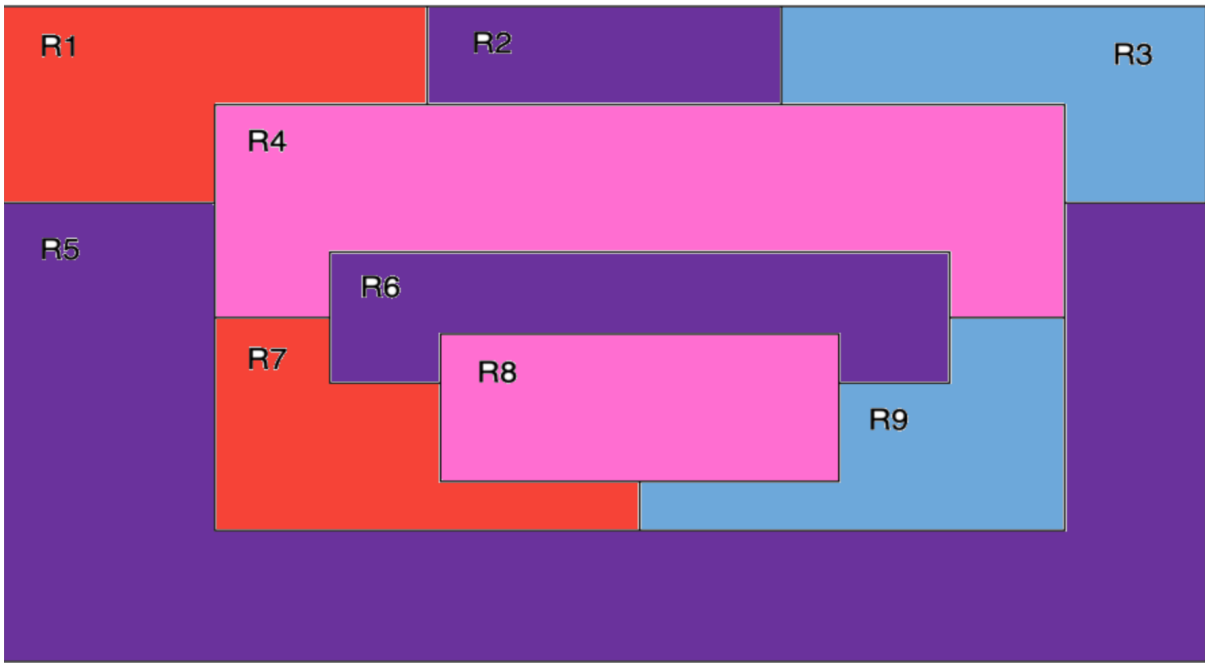
Task 1: Map Coloring



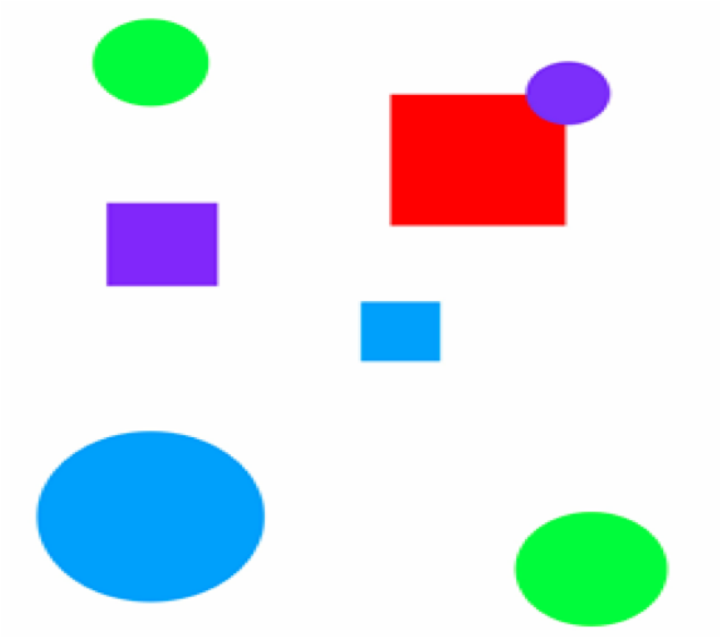
```
different(red,blue).
different(red,purple).
different(red,pink).

different(purple,blue).
different(purple,pink).
different(purple,red).

different(blue,purple).
different(blue,pink).
different(blue,purple).

different(pink,blue).
different(pink,purple).
different(pink,red).
```

```
coloring(R1,R2,R3,R4,R5,R6,R7,R8,R9) :-
    different(R1,R2),
    different(R1,R4),
    different(R1,R5),

    different(R2,R3),
    different(R2,R4),

    different(R3,R4),
    different(R3,R5),

    different(R4,R5),
    different(R4,R6),
    different(R4,R7),
    different(R4,R9),

    different(R5,R7),
    different(R5,R9),

    different(R6,R7),
    different(R6,R8),
    different(R6,R9),

    different(R7,R8),
    different(R7,R9),

    different(R8,R9).
```

```
% /Users/kyraedwards/Desktop/344/assignment4/coloring.pl compiled into coloring 0.00 sec, 13 clauses
?- coloring(R1,R2,R3,R4,R5,R6,R7,R8,R9).
R1 = R7, R7 = red,
R2 = R5, R5 = R6, R6 = purple,
R3 = R9, R9 = blue,
R4 = R8, R8 = pink
```



Task 2 : The Floating Shapes World



```
%
% --- square(N,side(L),color(C)) :: N is the name of a square with side L
% --- and color C
square(sera,side(7),color(purple)).
square(sara,side(5),color(blue)).
square(sarah,side(11),color(red)).
% ------------------------------------------------------------------------
% --- circle(N,radius(R),color(C)) :: N is the name of a circle with
% --- 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 of the circles
circles :- circle(Name,_,_), write(Name),nl,fail.
circles.
```

```
% ------------------------------------------------------------------------
% Rules ...
% ------------------------------------------------------------------------
% ------------------------------------------------------------------------
% --- circles :: list the names of all of the circles
circles :- circle(Name,_,_), write(Name),nl,fail.
circles.
% ------------------------------------------------------------------------
% --- squares :: list the names of all of the squares
squares :- square(Name,_,_), write(Name),nl,fail.
squares.
% ------------------------------------------------------------------------

% --- squares :: list the names of all of the shapes
shapes :- circles,squares.
% ------------------------------------------------------------------------
% --- blue(Name) :: Name is a blue shape
blue(Name) :- square(Name,_,color(blue)).
blue(Name) :- circle(Name,_,color(blue)).
% ------------------------------------------------------------------------
% --- large(Name) :: Name is a large shape
large(Name) :- area(Name,A), A >= 100.
% ------------------------------------------------------------------------
% --- small(Name) :: Name is a small shape
small(Name) :- area(Name,A), A < 100.
% ------------------------------------------------------------------------
% --- 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.
```

```
?- 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

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

?- cen(raichu).
false.

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

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

```
?- evolves(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.
```

```
?- evloves(A,B), evolves(B,C), write(A-->C), nl, fail.
Correct to: "evolves(A,B)"? yes
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(nks(Name,kind(Kind))),nl,fail.
nks(pikachu,kind(electric))
nks(raichu,kind(electric))
nks(bulbasaur,kind(grass))
nks(ivysaur,kind(grass))
nks(venusaur,kind(grass))
nks(caterpie,kind(grass))
nks(metapod,kind(grass))
nks(butterfree,kind(grass))
nks(charmander,kind(fire))
nks(charmeleon,kind(fire))
nks(charizard,kind(fire))
nks(vulpix,kind(fire))
nks(ninetails,kind(fire))
nks(poliwag,kind(water))
nks(poliwhirl,kind(water))
nks(poliwrath,kind(water))
nks(squirtle,kind(water))
nks(wartortle,kind(water))
nks(blastoise,kind(water))
nks(staryu,kind(water))
nks(starmie,kind(water))
false.

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

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

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

```
?- pokemon(_,water,_,attack(Attack,_)), wr
water-gun
amnesia
dashing-punch
bubble
waterfall
hydro-pump
slap
star-freeze
false.
```

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

```
?- pokemon(name(poliwhirl),_hp(HP),_).
ERROR: Syntax error: Operator expected
ERROR: pokemon(name(poliwhirl),_h
ERROR: ** here **
ERROR: p(HP),_) .
?- pokemon(name(poliwhirl),_,hp(HP),_).
HP = 80.
```

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

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

```
%
-------------------------------------------------------------------------
--
%
-------------------------------------------------------------------------
--
% --- File: pokemon.pro
% --- Line: Loosely represented 2-D shapes world (simple take on
SHRDLU)
%
-------------------------------------------------------------------------
--

%
-------------------------------------------------------------------------
--
% --- 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.
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.

?- 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, 5(
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
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(metapod),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(butterfree),grass,hp(130),attack(whirlwind,80))
false.

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

Task 4: L

1. Your demo corresponding to the “Head/Tail referencing exercises

```
?- [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)].
```

2. KB list processors pro

```
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).
```

3. Demo

```
?- 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], P).
P = [d, e, f, g, a, b].

?- 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(0,[four,three,two,one,zero],Element).
Element = four .

?- 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], Lang
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 = 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 = peach .

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

```
?- 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] .
```

4. KB associated with the“Example List Processors”that is provided in Lesson 5.

```
product([],1).
product([H|T],Product):-
product(T,P),
Product is H * P.

factorial(0,0).
factorial(N,Name):-
iota(N,I), product(I,Product),
Name is Product.

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

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

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

is_palindrome([]).
is_palindrome([_]).
is_palindrome(L):-
first(L,FE), last(L,LE),
FE = LE,
but_first(L,A), but_last(A,B), is_palindrome(B).

noun_phrase(N):-
pick([big,small,soft,hard,angry,happy],Adjective),
pick([apple,bee,cat,dog,fox,goose,boy,coffee],Noun),
add_last(Adjective,[the],Theadj), add_last(Noun,Theadj,N).

sentence(N):-
pick([saw,had,did,went,said,took,made],Verb),
noun_phrase(A), noun_phrase(B),
add_last(Verb,A,X),append(X,B,N).
```

5. The demo that you are asked to create in the “List Processing Exercises” section of Lesson 5.

```
?- consult('list_processors.pro')
true.

?- product([ ],P).
P = 1.

?- product([1,3,5,7,9],Product).
Product = 945.

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

?- product([ ],P).
P = 1.

?- 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.
```

```
?- product([1,3,5,7,9],Product).
Product = 945.

?- factorial(9,Product).
Product = 362880 .

?- 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].

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

?- noun_phrase(NP).
NP = [the, big, goose] ;
false.

?- noun_phrase(NP)
|    .
NP = [the, soft, coffee] .

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

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

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

?- noun_phrase(NP).
NP = [the, soft, fox] .

?- noun_phrase(NP).
NP = [the, big, fox] .

?- sentence(S).
S = [the, hard, goose, made, the, hard, coffee] .

?- sentence(S).
S = [the, angry, goose, made, the, small, goose] .

?- sentence(S).
S = [the, soft, bee, made, the, hard, boy] .

?- sentence(S).
S = [the, hard, apple, went, the, small, boy] .

?- sentence(S).
S = [the, happy, fox, took, the, angry, cat] .