Various Computations

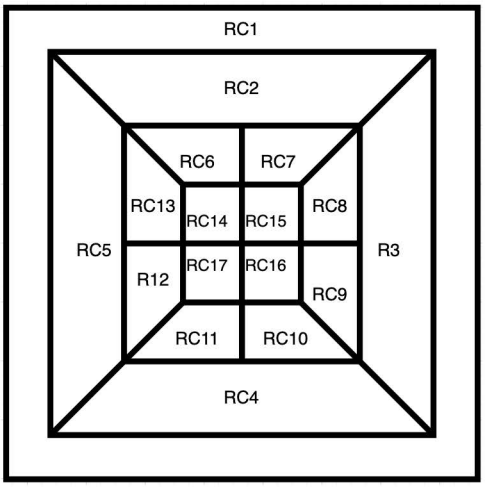
Prolog Programming
Assignment 1

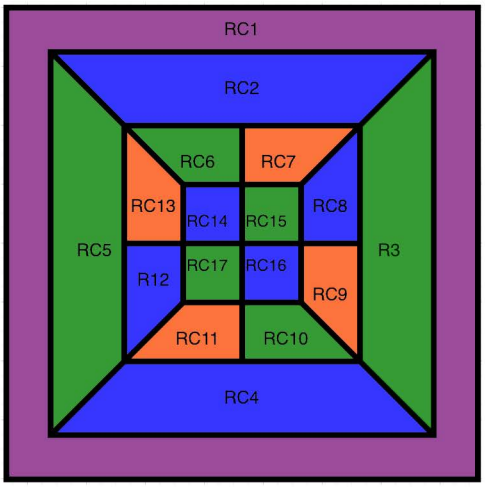
By: Miguel Cruz

Abstract

This assignment contains programming exercises that focus on knowledge representation, search, and list processing in Prolog.

Task 1: Map Coloring





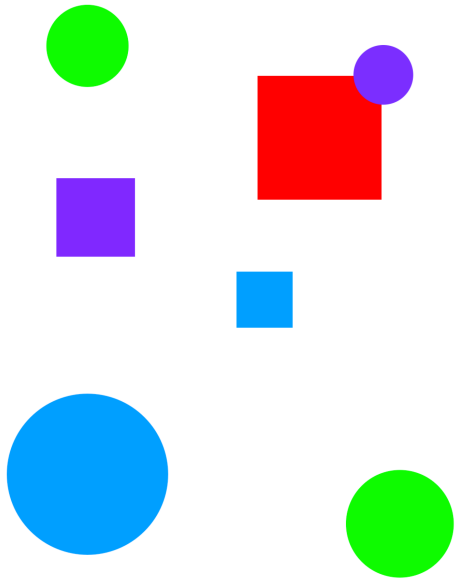
Colors Generated by the Program

```
 ~/Documents/Programming/CSC-344/src/PA5/ [main*] swipl map_coloring.pro
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.2)
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).

?-
coloring(RC1,RC2,RC3,RC4,RC5,RC6,RC7,RC8,RC9,RC10,RC11,RC12,RC13,RC14,RC15,RC16,RC17
). RC1 = purple,
RC2 = RC4, RC4 = RC8, RC8 = RC12, RC12 = RC14, RC14 = RC16, RC16 = blue,
RC3 = RC5, RC5 = RC6, RC6 = RC10, RC10 = RC15, RC15 = RC17, RC17 = green,
RC7 = RC9, RC9 = RC11,RRC11 = RC13, RC13 = orange
```

Source Code

```
% ---------------------------------------------------------------------------
% File: map_coloring.pro
% Line: Program to find a 4 color map rendering for South American coutries.
% More: The colors used will be purple, blue, green orange.
% More: The standard abbrieviations are used to stand for the countries.
% ---------------------------------------------------------------------------
% different(X,Y) :: X is not equal to Y
different(purple,blue).
different(purple,green).
different(purple,orange).
different(green,blue).
different(green,orange).
different(green,purple).
different(blue,green).
different(blue,orange).
different(blue,purple).
different(orange,blue).
different(orange,green).
different(orange,purple).

% ---------------------------------------------------------------------------
% coloring(RC1, RCX, ...) :: RC1, RCX, ... are the colors used in the map.

coloring(RC1,RC2,RC3,RC4,RC5,RC6,RC7,RC8,RC9,RC10,RC11,RC12,RC13,RC14,RC15,RC16,RC17) :-

different(RC1,RC2),
different(RC1,RC3),
different(RC1,RC4),
different(RC1,RC5),
different(RC2,RC3),
different(RC2,RC5),
different(RC2,RC6),
different(RC2,RC7),
different(RC3,RC8),
different(RC3,RC9),
different(RC3,RC4),
different(RC4,RC10),
different(RC4,RC11),
different(RC4,RC5),
different(RC5,RC12),
different(RC5,RC13),
different(RC6,RC7),
different(RC6,RC14),
different(RC6,RC13),
different(RC13,RC14),
different(RC13,RC12),
different(RC12,RC17),
different(RC12,RC11),
different(RC11,RC17),
different(RC11,RC10),
different(RC10,RC16),
different(RC10,RC9),
different(RC9,RC16),
different(RC9,RC8),
different(RC8,RC15),
different(RC8,RC7),
different(RC7,RC15),
different(RC15,RC14),
different(RC15,RC16),
different(RC16,RC17),
different(RC17,RC14).
```

Task 2: The Floating Shapes World



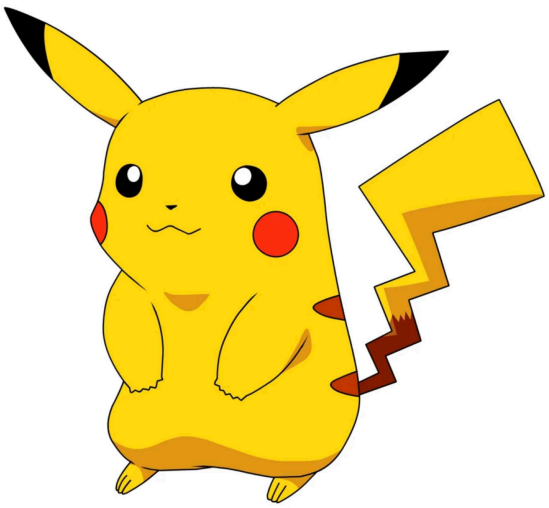
Source Code

```
% ----------------------------------------------------------------------
% ----------------------------------------------------------------------
% --- File: shapes_world_1.pro
% --- Line: Loosely represented 2-D shapes world (simple take on SHRDLU)
% ----------------------------------------------------------------------
% ----------------------------------------------------------------------
% --- Facts ...
% ----------------------------------------------------------------------
% ----------------------------------------------------------------------
% --- 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.
% ----------------------------------------------------------------------
% --- 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.
```

Interactions (Zoom in)

```
 ~/Documents/Programming/CSC-344/src/PA5/ [main*] clear
 ~/Documents/Programming/CSC-344/src/PA5/ [main*] swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.2)
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('shapes_world_1.pro').
true.

?- 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 ;
false
?-
```

Task 3: Pokemon KB Interaction and Programming



Part 1: Queries

```
 ~/Documents/Programming/CSC-344/src/PA5/ [main*] swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

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

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

?- 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(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(N,K,_,_),write('nks('),write(N),write(',kind('),write(K),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 .

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

?- 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(_,_,_,attack(N,P)),P > 60,write(N),nl,fail.
thunder-shock
poison-powder
whirlwind
royal-blaze
fire-blast
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 2: Programs

```
 ~/Documents/Programming/CSC-344/src/PA5/ [main*] swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

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

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

?- 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
true.

?- family(squirtle).
squirtle wartortle blastoise
true .

?- familes.
Correct to: "families"? yes
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))
true .

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

?-
```

Source Code

```
% ------------------------------------------------------------------------
% --- 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(N,_)),
     write(N),
      nl,
      fail.

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

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

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

  dump_kind(Type) :-
      pokemon(name(N),Type,H,A),
     write(pokemon(name(N),Type,H,A)),  nl,
      fail.


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

  family(N) :-
      cen(N),
           evolves(N,X),
     evolves(X,Y),
     write(N),
       write('   '),
     write(X),
     write('      '),
     write(Y).

  family(N) :-
      cen(N),
     evolves(N,X),  \
     +evolves(X,_),
        write(N),
     write('      '),
     write(X).

  families :-
      cen(N),
      family(N),
      nl,
      fail.

  lineage(N) :-
           evolves(N,X),
     evolves(X,Y),
        pokemon(name(N),TN,HN,AN),
pokemon(name(X),TX,HX,AX),
```

```
    pokemon(name(Y),TY,HY,AY),
    write(pokemon(name(N),TN,HN,AN)),
    nl,
    write(pokemon(name(X),TX,HX,AX)),
    nl,
    write(pokemon(name(Y),TY,HY,AY)).


lineage(N) :-
    evolves(N,X),
    pokemon(name(N),TN,HN,AN),
    pokemon(name(X),TX,HX,AX),
    write(pokemon(name(N),TN,HN,AN)),nl,write(pokemon(name(X),TX,HX,AX)).

lineage(N) :-
    pokemon(name(N),TN,HN,AN),
    write(pokemon(name(N),TN,HN,AN)).
```

Task 4: Lisp Processing in Prolog

HT Demo

```
 ~/Documents/Programming/CSC-344/src/PA5/ [main*] clear
 ~/Documents/Programming/CSC-344/src/PA5/ [main*] swipl list_processors.pro
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.2)
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)].

?-
```

```
 ~/Documents/Programming/CSC-344/src/PA5/ [main*] swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

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

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

?- first([apple],First).
First = apple.

?- first([c,d,e,f,g,a,b],P).
P = c.

?- rest([apple],Rest).
Rest = [].

?- rest([c,d,e,f,g,a,b],Rest).
Rest = [d, e, f, g, a, b].

?- last([peach],Last).
Last = peach .

?- last([c,d,e,f,g,a,b],Last).
Last = b .

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

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

```
 ~/Documents/Programming/CSC-344/src/PA5/ [main*] swipl list_processors.pro
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.2)
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).

?- write_list([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 = cherry .

?- 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 = cherry .

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

?- 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] .

?- 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] ;
```

```
?- 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([1,2,3,4,5,4,2,3,1]).
false.

?- is_palindrome([a,b,b,a]).
true .

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

?- noun_phrase(NP).
NP = [the, available, development] .

?- noun_phrase(NP).
NP = [the, helpful, ai] .

?- noun_phrase(NP).
NP = [the, helpful, goob] .

?- noun_phrase(NP).
NP = [the, helpful, terminator] .

?- noun_phrase(NP).
NP = [the, available, terminator] .

?- sentence(S).
S = [the, available, goob, ran-over, the, acidic, pizza]

?- sentence(S).
S = [the, upset, departure, punched, the, available, ai] .

?- sentence(S).
S = [the, helpful, ai, distracted, the, beneficial, ai] .

?- sentence(S).
S = [the, upset, ai, punched, the, beneficial, iphone] .

?- sentence(S).
S = [the, available, system, distracted, the, beneficial, development] .

?- sentence(S).
S = [the, available, system, ran-over, the, helpful, system] .

?- sentence(S).
S = [the, panoramic, ai, ran-over, the, helpful, iphone] .

?- sentence(S).
S = [the, upset, system, ran-over, the, available, goob] .

?- sentence(S).
S = [the, panoramic, departure, fought, the, upset, terminator] .

?- sentence(S).
S = [the, available, goob, punched, the, panoramic, system] .

?- sentence(S).
S = [the, available, departure, enacted, the, beneficial, terminator] .

?- sentence(S).
S = [the, available, system, enacted, the, acidic, departure] .

?- sentence(S).
S = [the, beneficial, departure, ran-over, the, upset, departure] .

?- sentence(S).
S = [the, acidic, goob, punched, the, acidic, iphone] .

?- sentence(S).
S = [the, upset, development, calculated, the, acidic, goob] .

?- sentence(S).
S = [the, helpful, development, punched, the, helpful, goob] .

?-
```

```
% ----------------------------------------------------------------------
% File: 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).

write_list([]).

write_list([H|T]) :-
    write(H),nl,
    write_list(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,F) :-
    iota(N,I), product(I,F).

make_list(0,_,[]).
make_list(N,Item,List) :-
    K is N - 1,
    make_list(K,Item,Tail),
    add_last(Item,Tail,List).
```

```
but_first([],[]).
but_first([_|X],X).
but_last([], []).
but_last([_], []).
but_last(L,A) :-
    reverse(L,L1),but_first(L1,L2),reverse(L2,A).

is_palindrome([]) :- true.
is_palindrome([_]) :- true.
is_palindrome(L) :-
    first(L,A),
    last(L,B),
    A == B,
    but_first(L,L1),
    but_last(L1,L2),
    is_palindrome(L2).

adjectives([acidic,available,helpful,upset,beneficial,panoramic]).

nouns([departure,terminator,development,pizza,iphone,goob,ai,system]).

past_tense([fought,distracted,enacted,ran-over,punched,jumped,calculated]).

noun_phrase([the,Adjective,Noun]) :-
    adjectives(A),
    nouns(N),
    pick(A, Adjective),
    pick(N, Noun).

sentence(S) :-
    noun_phrase(NP1),
    noun_phrase(NP2),
    past_tense(PT),
    pick(PT, PastTense),
    append(NP1, [PastTense], S1),
    append(S1, NP2, S).
```