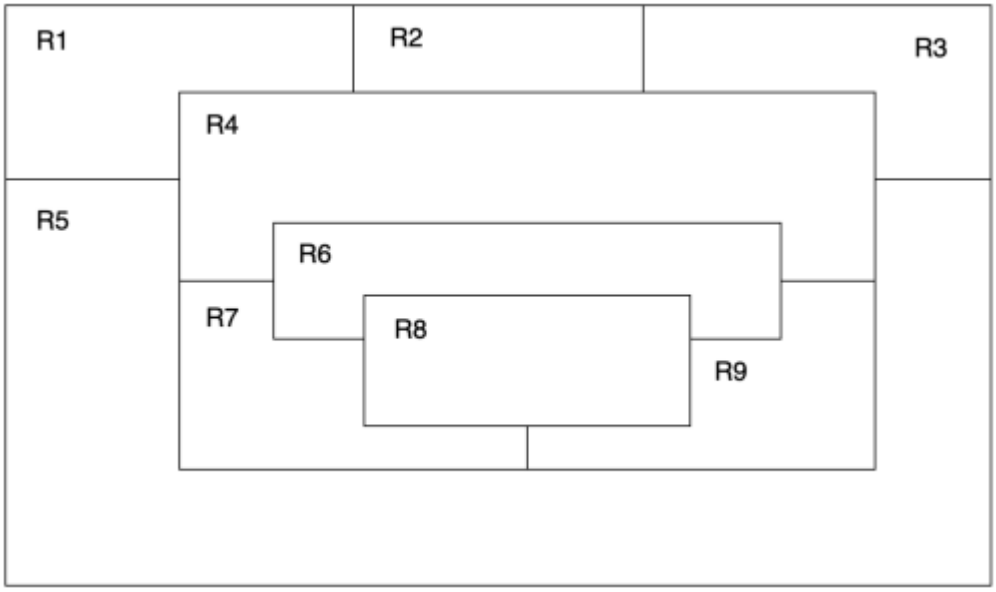
Abstract:
In this assignment we learned list manipulation, implications, unification, and how to write recursive programs in prolog using backtracking.

Task 1: Map Coloring

Blank Image:



Source:

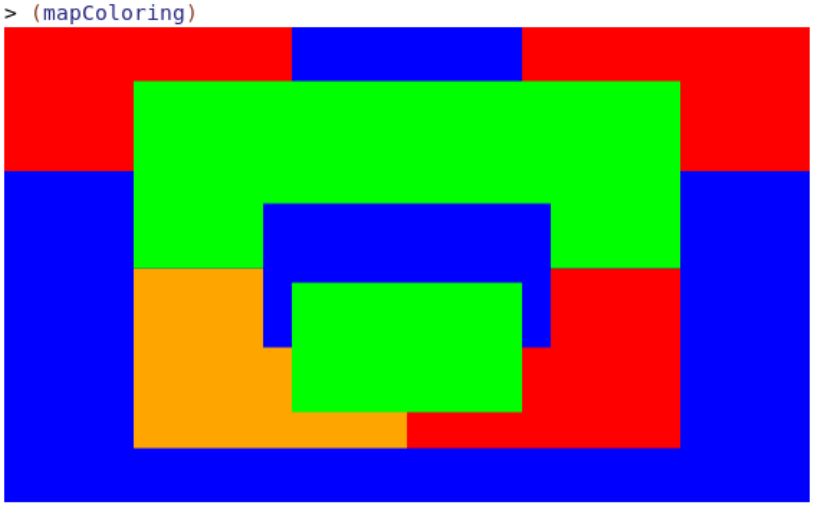
Demo:

```
%-----------------------------------------------------------------------------------------
% File PA1T1.pro
% Line: Program to find a 4 color map rendering for R Map
% More: The colors used will be red, blue, green and orange.

% different(X,Y) :: X is not equal to Y

different(red,blue).
different(red,green).
different(red,orange).
different(green,blue).
different(green,orange).
different(green,red).
different(blue,red).
different(blue,green).
different(blue,orange).
different(orange,blue).
different(orange,green).
different(orange,red).


%-----------------------------------------------------------------------------------------
% Coloring(R1,R2,R3,R4,R5,R6,R7,R8,R9) :: The Rs represent places on a map they are colored so that
none of the Rs that share a border are of the same color.

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).
```

```
jchi@jchi-Predator-G9-793:~/PrologProjects$ swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.5.0-75-g684c117c6)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

    CMake built from "/home/jchi/swipl-devel/build"

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

?- consult("PA1T1.pro").
true.

?- coloring(R1,R2,R3,R4,R5,R6,R7,R8,R9).
R1 = R3, R3 = R9, R9 = red,
R2 = R5, R5 = R6, R6 = blue,
R4 = R8, R8 = green,
R7 = orange .

?-
```

Colored Image:

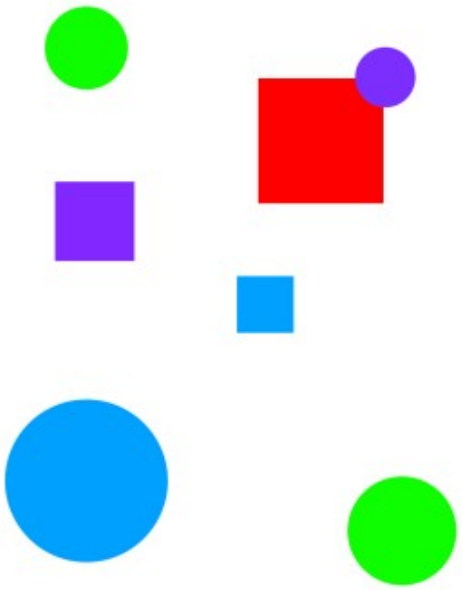
```
Welcome to DrRacket, version 8.2 [cs].
Language: racket, with debugging and profiling [custom]; memory limit: 1000 MB.
> (mapColoring)
```



```
>
```

Task 2: The Floating Shapes World

Image:



The Prolog KB:

```
%-------------------------------------------------------------------------------------------------------
%-------------------------------------------------------------------------------------------------------
%--- 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.

%-------------------------------------------------------------------------------------------------------
%--- shapes :: list the names of the shapes

shapes :- circles,squares.

 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.
```

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.5.0-75-g684c117c6)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

    CMake built from "/home/jchi/swipl-devel/build"

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

?- consult("PA1T2.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 .

?- halt.
```

Task 3: Pokemon KB Interaction and Programming

Demo P1:

```
jchi@jchi-Predator-G9-793:~/PrologProjects$ swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.5.0-75-g684c117c6)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

    CMake built from "/home/jchi/swipl-devel/build"

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

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

?- evolves(squirtle,wartortle).
true.

?- evolves(wartortle,squirtle).
false.

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

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

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

?- pokemon(name(PN),_,_,_),write(PN),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(poliwhirl),_,hp(X),_).
X = 80.

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

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

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

?- cen(P),pokemon(name(P),_,hp(X),_),write(P),write(":  "),write(X),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(PN),fire,_,_),write(PN),nl,fail.
charmander
charmeleon
charizard
vulpix
ninetails
false.

?- pokemon(name(PN),T,_,_),write(nks(name(PN),kind(T))),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(PN),_,_,attack(waterfall,_)).
PN = wartortle .

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

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

Extended Knowledge base:

```
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() :: Displays all pokemon names

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

%------------------------------------------------------------------------
% --- display_attacks :: Display all pokemons attacks

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

%------------------------------------------------------------------------
% --- powerful(Name). :: There is a pokemon with N name that is powerful
% --- ( dmg > 55 )

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

%------------------------------------------------------------------------
% --- tough(name(Name)). :: There is a pokemon with Name that is tough
% --- (HP > 100)

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

%------------------------------------------------------------------------
% --- tyoe(Name,Type) :: There is a pokemon with Name and Type.

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

```
%------------------------------------------------------------------------
% --- dump_kind(Type). :: Dumps all Pokemon and attributes of those pokemon

dump_kind(Type):- pokemon(N,Type,HP,Attack), write(pokemon(N,Type,HP,Attack)), nl,fail.

%------------------------------------------------------------------------
% --- display_cen() :: displays all of the "creati ex nihlo" pokemon

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

%------------------------------------------------------------------------
% --- family(CenP) :: takes in a "creati ex nihilo" pokemon and deiplays evolution tree

family(X):- evolves(X,Y),X\==Y,\+evolves(Y,_),write(X),write(" "),write(Y).
family(X):- evolves(X,Y),evolves(Y,Z),X\==Y, write(X),write(" "),write(Y),write(" "),write(Z).

%------------------------------------------------------------------------
% --- families() :: displays all families of pokemon.


families() :- family(_), nl , fail.
families().

%------------------------------------------------------------------------
% --- linage(X) :: takes in a pokemon and outputs all info on subsequest evolutions

%linage(X):- evolves(X,Y)
```

Demo

```
?- 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
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.
```

Demo Cont.

```
?- 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.
```

Demo cont.

```
?- 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(fi).
file                      file_no                   find_in_history
fileExtension             file_prefix               find_library
file_alias_path           file_search               find_nth0
file_auto_import          file_search_cache_time    find_predicate
file_autoload_directives  file_search_path           find_predicate_
file_base_name            file_sha1                 find_sim_pred
file_char                 file_type                 find_subgoal
file_char_count           fileerrors                findall
file_chars                filepos_line              findall_loop
file_directory_name       files                     findnsols
... skipped 8 rows
?- 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
```

Demo cont.

```
?- family(pikachu).
pikachu raichu
true .

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

?- families.
pikachu raichu
ivysaur venusaur
metapod butterfree
charmeleon charizard
vulpix ninetails
poliwhirl poliwrath
wartortle blastoise
staryu starmie
bulbasaur ivysaur venusaur
caterpie metapod butterfree
charmander charmeleon charizard
poliwag poliwhirl poliwrath
squirtle wartortle blastoise
true.

?- linage(caterpie).
pokemon(caterpie,grass,hp(50),attack(gnaw,20))
pokemon(metapod,grass,hp(70),attack(stun-spore,20))
pokemon(butterfree,grass,hp(130),attack(whirlwind,80))
pokemon(metapod,grass,hp(70),attack(stun-spore,20))
pokemon(metapod,grass,hp(70),attack(stun-spore,20))
pokemon(butterfree,grass,hp(130),attack(whirlwind,80))
false.

?- linage(metapod).
pokemon(metapod,grass,hp(70),attack(stun-spore,20))
pokemon(butterfree,grass,hp(130),attack(whirlwind,80))
false.

?- linage(butterfree).
false.
```

Task 4: Lisp Processing in Prolog

Head/Tail Exercises

```
jchi@jchi-Predator-G9-793:~/PrologProjects$ swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.5.0-75-g684c117c6)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

    CMake built from "/home/jchi/swipl-devel/build"

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.

?- 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],SumOfPrime).
SumOfPrime = 28.
```

Source

```
%-------------------------------------------------------------------------------------------------------
% --- File: list_processors.pro -----------------------------------------------------------------------
%-------------------------------------------------------------------------------------------------------
%-------------------------------------------------------------------------------------------------------
% Code First
%-------------------------------------------------------------------------------------------------------
% --- Fist(X,Y) :: takes in list in X and returns the head of a list in Y

first([H|_], H).

%-------------------------------------------------------------------------------------------------------
% Code Rest
%-------------------------------------------------------------------------------------------------------
% --- rest(X,Y) :: takes in a list X and returns tail of list in Y
rest([_|T],T).

%-------------------------------------------------------------------------------------------------------
% Code Last
%-------------------------------------------------------------------------------------------------------
% --- last(X,Y) ::  recusrsivly looks at the tail of the list until tail is empty then return head.
last([H|[]],H).
last([_|T],Result):- last(T,Result).

%-------------------------------------------------------------------------------------------------------
% Code Nth
%-------------------------------------------------------------------------------------------------------
% --- nth(X,Y) ::  recursivly shrinks down list looking at tail until we reach the index we are look
ing%for this is done by subtracting the N until it is zero this leaves us to the desired index
nth(0, [H|_],H).
nth(N, [_|T],E) :- K is N-1, nth(K,T,E).

%-------------------------------------------------------------------------------------------------------
% Code Writelist
%-------------------------------------------------------------------------------------------------------
% --- writelist(X) :: takes in a List and recursivly writes it to the screen until list is empty.
writelist([]).
writelist([H|T]) :- write(H), nl, writelist(T).

%-------------------------------------------------------------------------------------------------------
% Code Sum
%-------------------------------------------------------------------------------------------------------
% --- sum(X,Y) :: takes in a list X runs itslef on the tail of the rest of the list and assigns the
%Head to the first of tail until list is empty while assigning the last tail to Sum of tail then add
%ing it up from 0 to the the head brfore list was empty and so on.
sum([],0).
sum([Head|Tail],Sum):-
  sum(Tail,SumOfTail),
  Sum is Head + SumOfTail.

%-------------------------------------------------------------------------------------------------------
% Code Add_first
```

Cont

```
%
%-----------------------------------------------------------------------------------------------------
% Code Product
%-----------------------------------------------------------------------------------------------------
% --- product(X,Y) :: takes in a list at X produces proudct of mememberS of list as output.
product([],1).
product([Head|Tail],Product) :-
  product(Tail,ProductOfTail),
  Product is Head * ProductOfTail.

%-----------------------------------------------------------------------------------------------------
% Code Factorial
%-----------------------------------------------------------------------------------------------------
% --- factorial(N,Y):: takes in a positive number N and returns the factorial in Y
factorial(1,1).
factorial(N,Y) :-
  iota(N, ListOFNumbers),
  product(ListOFNumbers,Y).

%-----------------------------------------------------------------------------------------------------
% Code make_list
%-----------------------------------------------------------------------------------------------------
% --- make_list(X,Y,Z) :: takes a postive integer as its first parmeter a data item as Y and a list
% with the data object being repreated X times Constraint: recursive.
make_list(0,_,[]).
make_list(X,Y,[H|T]):-
  X>0,
  H is Y ,
  XX is X-1,
  make_list(XX,Y,T).

%-----------------------------------------------------------------------------------------------------
% Code but_first
%-----------------------------------------------------------------------------------------------------
% ---but_first([H|T],X) ::  take in a non-empty list and returns tail of list
but_first([_|T],T).

%-----------------------------------------------------------------------------------------------------
% Code but_last
%-----------------------------------------------------------------------------------------------------
% --- but_last([H|T],X) :: takes in nonempty list as first arg and returns rdc as X
but_last([H|T],X):-
  reverse([H|T],Y),
  but_first(Y,Z),
  reverse(Z,X).

%-----------------------------------------------------------------------------------------------------
% Code is_palindrome
%-----------------------------------------------------------------------------------------------------
% --- is_palindrome(X) ::  takes in a list as its parameter and succueds if list is a palendrome
is_palindrome([]).
is_palindrome([_|T]):- T == [].
```

Source Cont.

Source Cont.

```
is_palindrome(X):-
  last(X,Z),first(X,O), O==Z,
  but_last(X,Q),but_first(Q,V),
  is_palindrome(V).

%-----------------------------------------------------------------------------------------------------
% Code noun_phrase
%-----------------------------------------------------------------------------------------------------
% --- noun_phrase :: takes in a variable and produces a sentence Format:"The"  adj Noun
noun_phrase(S):-
  add_first(the,[],X),
  pick([omniscient,crowded,serious,depressed,lethal,victorious],Adj),
  pick([quality,polish,drink,manager,cloth,sky,change,punsihment],N),
  add_last(Adj,X,Z),
  add_last(N,Z,S).

%-----------------------------------------------------------------------------------------------------
% Code Sentence
%-----------------------------------------------------------------------------------------------------
% --- sentence :: takes in a variable and produces a sentence witha NP followed by a past tnese verb
% followed by a NP
sentence(X):-
  noun_phrase(Y),
  pick([was,beat,became,began,came,cost,found,fought], Past),
  noun_phrase(U),
  add_last(Past,Y,V),
  first(U,F),
  rest(U,REST),
  first(REST,K),
  rest(REST,WE),
  first(WE,L),
  add_last(F,V,Z),
  add_last(K,Z,M),
  add_last(L,M,X).
```

Example list processing

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

?- 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 = peach .

?- 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 = apple .

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

?- make_set([1,1,2,1,2,3,1,2,3,4],Set).
Set = [1, 2, 3, 4] .

?- make_set([bit,bot,bet,bot,bot,bit],B).
B = [bet, bot, bit] .
```

Demo

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

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

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

?- iota(9,Iota),product(Iota,Product).
Iota = [1, 2, 3, 4, 5, 6, 7, 8, 9],
Product = 362880 .

?- make_list(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_pa
is_palindrome  is_paren
?- 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, omniscient, drink] .

?- noun_phrase(NP).
NP = [the, omniscient, polish]
```

Cont.

```
?- noun_phrase(NP).
NP = [the, victorious, punsihment] .

?- noun_phrase(NP).
NP = [the, victorious, polish] .

?- sentence(S).
S = [the, serious, sky, beat, the, victorious, punsihment] .

?- sentence(S).
S = [the, victorious, polish, fought, the, omniscient, change] .

?- sentence(S).
S = [the, crowded, punsihment, beat, the, victorious, drink] .

?- sentence(S).
S = [the, victorious, polish, found, the, crowded, sky] .

?- sentence(S).
S = [the, crowded, sky, cost, the, crowded, cloth] .

?- sentence(S).
S = [the, depressed, quality, cost, the, serious, sky] .

?- sentence(S).
S = [the, omniscient, drink, came, the, lethal, change] .

?- sentence(S).
S = [the, serious, change, fought, the, victorious, quality] .

?- sentence(S).
S = [the, victorious, drink, came, the, victorious, sky] .

?- sentence(S).
S = [the, lethal, change, fought, the, omniscient, manager] .

?- sentence(S).
S = [the, crowded, cloth, began, the, serious, drink] .

?- sentence(S).
S = [the, lethal, quality, began, the, omniscient, quality] .
```