

**Learning Abstract:** Task 1 involves establishing and interacting with the knowledge base detailed in Prolog Lesson 1, a very simple KB pertaining to colors. Task 2 involves establishing and interacting with a very simple KB which is structurally just like the given KB of Task 1, but which you are asked to piece together yourself, one pertaining to food. Task 3, based on Prolog Lesson 3, is all about solving a map coloring problem. Task 4 involves establishing and interacting with a given KB of a bit more complexity than that featured in the first task. This is the KB about floating shapes, inspired by Terry Winograd's blocks world, that was presented in Prolog Lesson 4. Collectively, these tasks afford an opportunity to get acquainted with the basics of Prolog programming.

### **Task 1 – Colors KB:**

Colors KB Code:

```
% -----  
% File: colors.pro  
% Line: Six color facts, structured into primaries and secondaries  
% -----  
% primary(P) :: P is a primary color  
primary(blue).  
primary(red).  
primary(yellow).  
% -----  
% secondary(S) :: S is a secondary color  
secondary(green).  
secondary(orange).  
secondary(purple).  
% -----
```

% color(C) :: C is a color

color(C) :- primary(C).

color(C) :- secondary(C).

## Colors KB Interaction:

```
SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)
File Edit Settings Run Debug Help
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).

?- primary(blue).
ERROR: Unknown procedure: primary/1 (DWIM could not correct goal)
?- consult('colors.pro')
|
true.

?- primary(blue).
true.

?- primary(red).
true.

?- primary(green).
false.

?- secondary(green).
true.

?- secondary(purple).
true.

?- secondary(yellow).
false.

?- color(blue).
true.

?- color(purple).
true.

?- primary(P).
P = blue ;
P = red ;
P = yellow.

?- secondary(S)
|
S = green ;
S = orange ;
S = purple.

?- color(C).
C = blue ;
C = red ;
C = yellow ;
C = green ;
C = orange ;
C = purple.

?- listing(primary).
primary(blue).
primary(red).
primary(yellow).

true.

?- listing(secondary).
secondary(green).
secondary(orange).
secondary(purple).

true.

?- listing(color).
color(C) :-
    primary(C).
color(C) :-
    secondary(C).

true.


?-
```

## Task 2 –Food KB:

Food KB Code:

```
% -----  
% File:foods.pro  
% Line: Six food facts, structured into fruits and vegetables  
% -----  
% fruit(F) :: F is a fruit  
fruit(grapefruit).  
fruit(avocado).  
fruit(date).  
% -----  
% vegetable(V) :: V is a vegetable  
vegetable(asperagus).  
vegetable(broccoli).  
vegetable(carrot).  
% -----  
% food(F) :: F is a food  
food(F) :- fruit(F).  
food(F) :- vegetable(F).
```

## Food KB Interaction:

 SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)

File Edit Settings Run Debug Help

Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)  
 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).`

```
?- fruit(grapefruit).
ERROR: Unknown procedure: fruit/1 (DWIM could not correct goal)
?- consult('foods.pro').
true.
```

```
?- fruit(grapefruit).
true.
```

```
?- fruit(avocado).
true.
```

```
?- fruit(asperagus).
false.
```

```
?- vegetable(asperagus).
true.
```

```
?- vegetable(broccoli).
true.
```

```
?- vegetable(date).
false.
```

```
?- food(grapefruit).
true.
```

```
?- food(carrot).
true.
```

```
?- fruit(F).
F = grapefruit ;
F = avocado ;
F = date.
```

```
?- vegetable(V).
V = asperagus ;
V = broccoli ;
V = carrot.
```

```
?- food(F).
F = grapefruit ;
F = avocado ;
F = date ;
F = asperagus ;
F = broccoli ;
F = carrot.
```

```
?- listing(fruit).
fruit(grapefruit).
fruit(avocado).
fruit(date).
```

```
true.
```

```
?- listing(vegetable).
vegetable(asperagus).
vegetable(broccoli).
vegetable(carrot).
```

```
true.
```

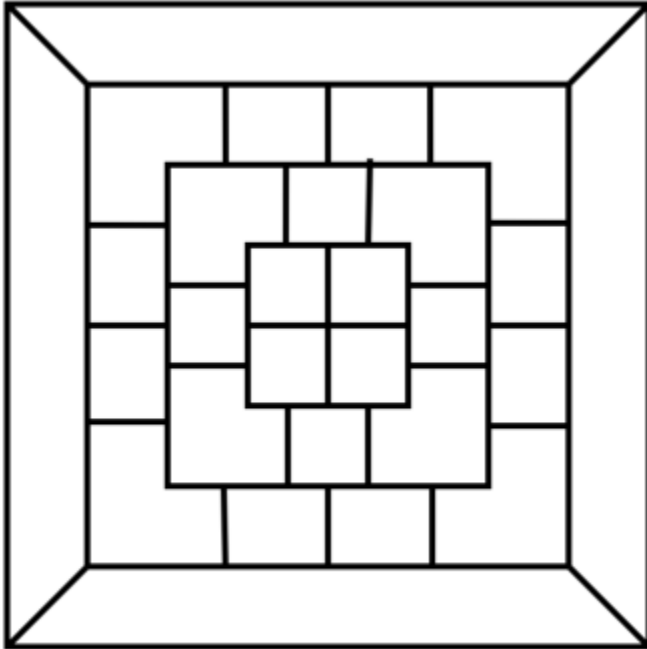
```
?- listing(food).
food(F) :-
    fruit(F).
food(F) :-
    vegetable(F).
```

```
true.
```

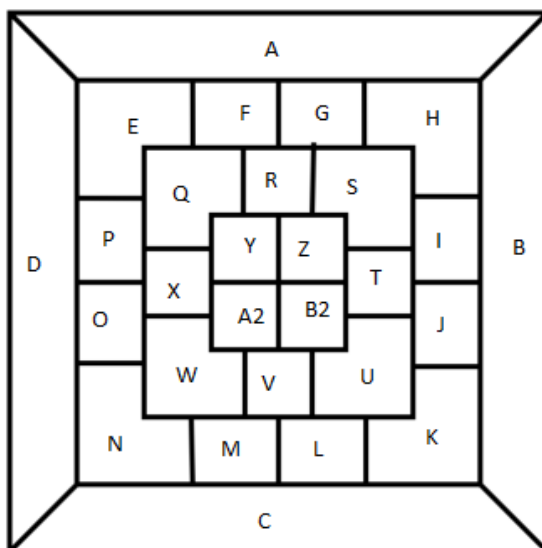
```
?- ■
```

### Task 3 – Map Coloring:

The Given Map:



The Labeled Map:



Code for Coloring the Map:

% -----

```
% File: map_coloring.pro

% Line: Program to find a 4 color map rendering for the given map.

% More: The colors used will be red, blue, green, and orange.

% More: The letters are taken from the labeled map provided

% -----

% 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,green).

different(blue,orange).

different(blue,red).

different(orange,blue).

different(orange,green).

different(orange,red).

% -----

% coloring(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,A2,B2) :: The regions
labeled by their letter are colored

% so that none of the regions sharing a border are the same color.

coloring(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,A2,B2) :-
different(A, B),
different(A, D),
different(A, E),
different(A, F),
different(A, G),
```

different(A, H),  
different(B, H),  
different(B, I),  
different(B, J),  
different(B, K),  
different(B, C),  
different(C, K),  
different(C, L),  
different(C, M),  
different(C, N),  
different(C, D),  
different(D, N),  
different(D, O),  
different(D, P),  
different(D, E),  
different(E, F),  
different(E, P),  
different(E, Q),  
different(F, G),  
different(F, R),  
different(F, Q),  
different(G, H),  
different(G, R),  
different(G, S),  
different(H, I),  
different(H, S),  
different(I, J),

different(I, S),  
different(I, T),  
different(J, K),  
different(J, T),  
different(J, U),  
different(K, L),  
different(K, U),  
different(L, M),  
different(L, U),  
different(L, V),  
different(M, N),  
different(M, V),  
different(M, W),  
different(N, O),  
different(N, W),  
different(O, P),  
different(O, W),  
different(O, X),  
different(P, Q),  
different(P, X),  
different(Q, R),  
different(Q, X),  
different(Q, Y),  
different(R, S),  
different(R, Y),  
different(R, Z),  
different(S, T),



different(S, Z),  
different(T, U),  
different(T, Z),  
different(T, B2),  
different(U, V),  
different(U, B2),  
different(V, W),  
different(V, A2),  
different(V, B2),  
different(W, X),  
different(W, A2),  
different(X, Y),  
different(X, A2),  
different(Y, Z),  
different(Y, A2),  
different(Y, B2),  
different(Z, A2),  
different(Z, B2),  
different(A2, B2).

## Map Coloring Interaction:



SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)

File Edit Settings Run Debug Help

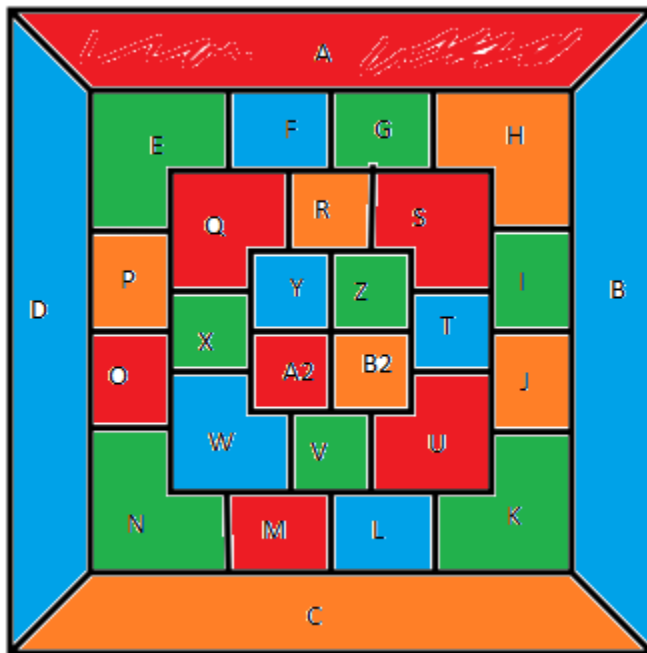
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)  
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.  
Please run `?- license.` for legal details.

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

`?- consult('map_coloring.pro').`  
**true.**

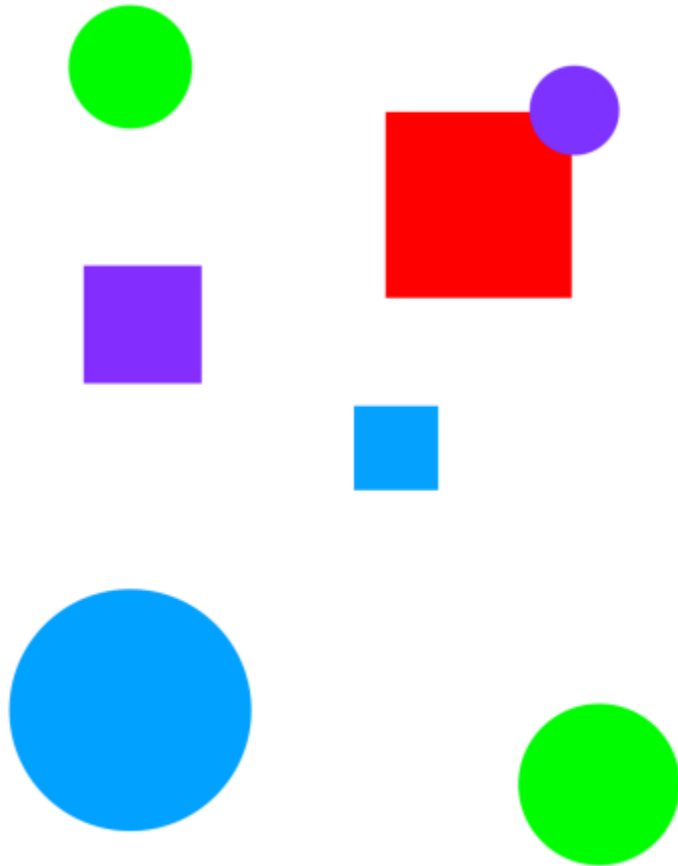
`?- coloring(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,A2,B2).`  
A = M, M = O, O = Q, Q = S, S = U, U = A2, A2 = red,  
B = D, D = F, F = L, L = T, T = W, W = Y, Y = blue,  
C = H, H = J, J = P, P = R, R = B2, B2 = orange,  
E = G, G = I, I = K, K = N, N = V, V = X, X = Z, Z = green ■

## The Colored Map:



## Task 4 –Floating Shapes World KB:

Floating Shapes World Image:



---


Floating Shapes World KB Code:

```
% -----  
% -----  
% --- File: shapes_world.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 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.
```

Floating Shapes World KB Interaction:

 SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)

File Edit Settings Run Debug Help

Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)  
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.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 ;
false.
```

`?- area(carla,A).`

```
A = 50.24 ;
false.
```

`?- ■`