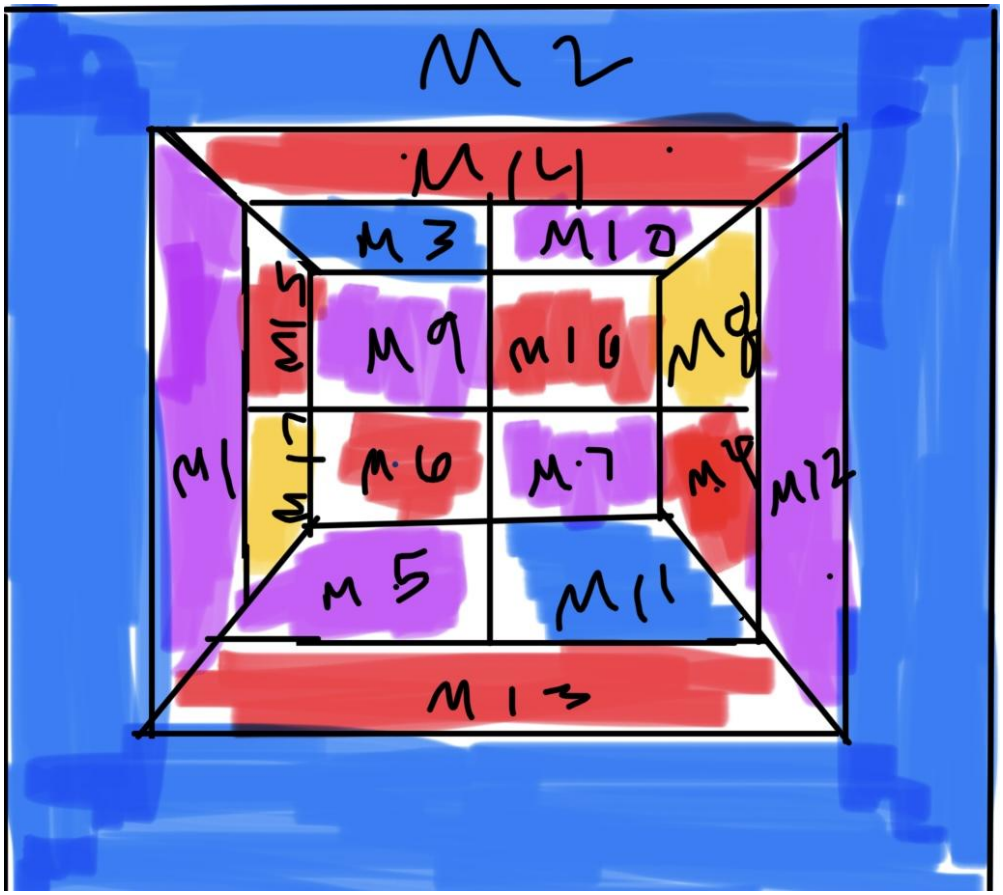
Prolog Programming Assignment #1: Various Computations

Learning Abstract: In this Assignment, we will program exercises in Prolog that focus on knowledge representation, search, and list processing. The first task was working by analogy with a map coloring tool presented in class. Task 2 involves entering the Prolog code for the floating forms world KB that was given in class. Task 3 required us to incorporate the information base on Pokemon trade cards from class into our computational universe. In the last work, we complete the "Head/Tail Referencing Exercises" segment from Lesson 5 on list processing in Prolog.

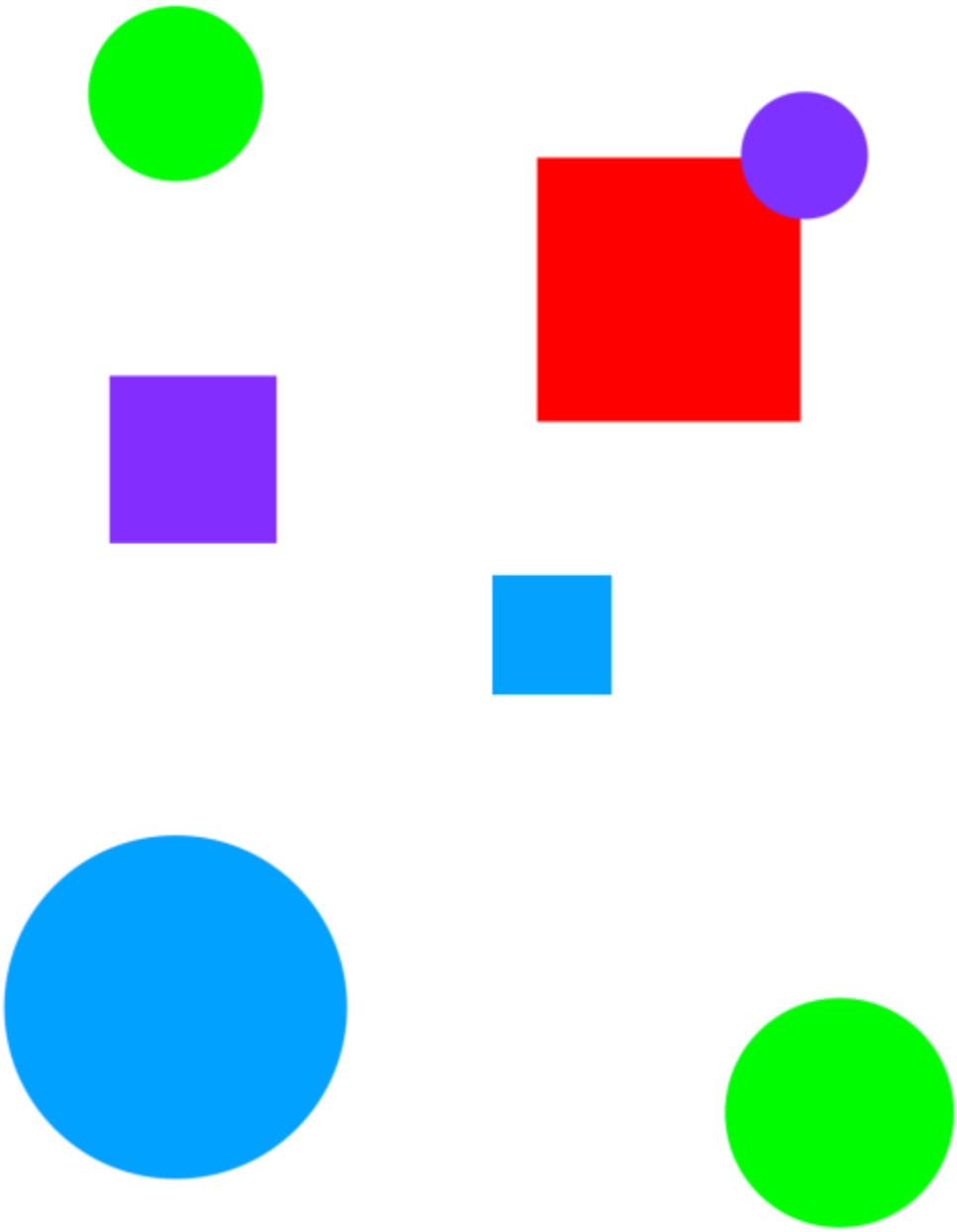
Task 1: Map Coloring



```
% ------------------------------------------------------------------------
% File: switest.pl
% ------------------------------------------------------------------------
% different(X,Y) :: X is not equal to Y
different(red,purple).
different(red,blue).
different(red,yellow).
different(purple,red).
different(purple,blue).
different(purple,yellow).
different(blue,red).
different(blue,purple).
different(blue,yellow).
different(yellow,red).
different(yellow,purple).
different(yellow,blue).
% ------------------------------------------------------------------------
coloring(M1,M2,M3,M4,M5,M6,M7,M8,M9,M10,M11,M12,M13,M14,M15,M16,M17) :-
different(M17, M12),
different(M13, M10),
different(M14, M9),
different(M13, M11),
different(M16, M10),
different(M15, M1),
different(M12, M17),
different(M17, M11),
different(M11, M16),
different(M11, M7),
different(M10, M14),
different(M10, M15),
different(M10, M2),
different(M9, M2),
different(M9, M8),
different(M9, M11),
different(M8, M2),
different(M8, M6),
different(M8, M16),
different(M2, M5),
different(M2, M15),
different(M5, M6),
different(M4, M7),
different(M6, M5),
different(M3, M4),
different(M17, M3),
different(M17, M7),
different(M1, M3),
different(M1, M17).
```

```
?- coloring(M1,M2,M3,M4,M5,M6,M7,M8,M9,M10,M11,M12,M13,M14,M15,M16,M17).
M1 = M5, M5 = M7, M7 = M9, M9 = M10, M10 = M12, M12 = purple,
M2 = M3, M3 = M11, M11 = blue,
M4 = M6, M6 = M13, M13 = M14, M14 = M15, M15 = M16, M16 = M17, M17 = red,
M8 = yellow
```

Task 2: The Floating Shapes World



KB in Prolog

```
% ------------------------------------------------------------------------
% ------------------------------------------------------------------------
% --- File: switest.pl
% --- 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
```

```
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.
```

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.0)
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).

?-
% c:/Users/Erica/IntelGraphicsProfiles/Desktop/prolog/switest.pl comp
?-  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(squirtle,wartortle).
true.

?- evolves(wartortle,squirtle).
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(name(N),TYP,_,_),write(nks(N,kind(TYP))),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 ;
false.

?- pokemon(name(N),_,_,attack(poision-powder,_)).
false.

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

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

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

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

```
?- pokemon(name(N),_,_,attack(_,INT)),INT>60,write(N),nl,false.
raichu
venusaur
butterfree
charizard
ninetails
false.

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

Part 2:

```
?- 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(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
false.

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

```
?- families.

pikachu raichu
bulbasaur ivysaurvenusaur

caterpie metapodbutterfree

charmander charmeleoncharizard

vulpix ninetails
poliwag poliwhirlpoliwrath

squirtle wartortleblastoise

staryu starmie
false.

?- lineage(caterpie).
pokemon(name(caterpie),grass,hp(50),attack(gnaw,20))

pokemon(name(pikachu),electric,hp(60),attack(gnaw,10))

pokemon(name(raichu),electric,hp(90),attack(thunder-shock,90))
true .

?- lineage(metapod)
|    .
pokemon(name(metapod),grass,hp(70),attack(stun-spore,20))

pokemon(name(pikachu),electric,hp(60),attack(gnaw,10))

pokemon(name(raichu),electric,hp(90),attack(thunder-shock,90))
true .

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

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

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

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

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

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

dump_kind(TYP) :- pokemon(Name,TYP,HP,A), write(pokemon(Name,TYP,HP,A)), nl, fail.

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

family(CEN) :- evolves(CEN,E), write(CEN), write(" "), write(E), evolves(E, V), write(" "), write(V).

families :- cen(CEN), evolves(CEN,E), nl, write(CEN), write(" "),  write(E), evolves(E, V), write(V), nl, fail.

lineage(Name) :- pokemon(name(Name),T,H,A), write(pokemon(name(Name),T,H,A)), nl, evolves(Name,E),
pokemon(name(X),TT,TH,TA), nl, write(pokemon(name(X),TT,TH,TA)),nl, evolves(X, Y),
pokemon(name(Y),TT2,TH2,TA2), nl, write(pokemon(name(Y),TT2,TH2,TA2)).
```

Task 4: Lisp Processing in Prolog

```
?-  [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)].

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

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

?- est([apple],Rest).
Correct to: "rest([apple],Rest)"? yes
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],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 = peach ,

?-  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] ,
```

```
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).
```