Prolog Programming Assignment #1: Various Computations

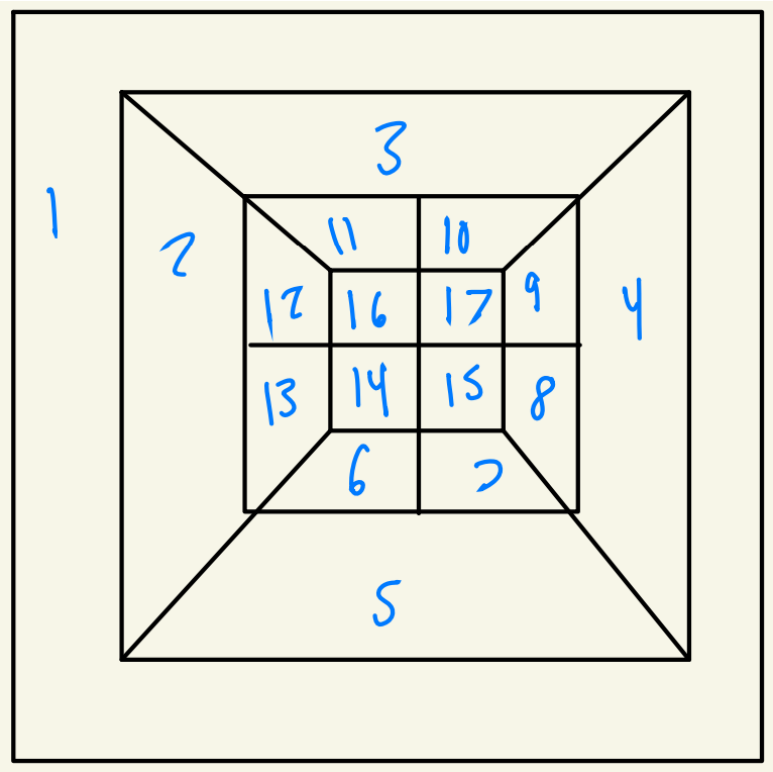
Bryan McLean

Learning Abstract:

This assignment helps students become familiar with Prolog by using knowledge representation, search, and list processing.

Task 1: Map Coloring

Map:



Source Program:

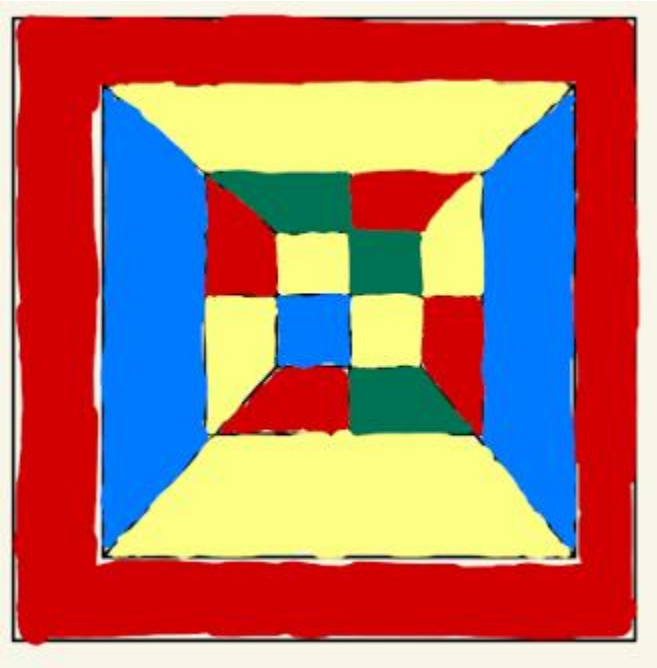
```
%----------------------------------------------------
% File: map_coloring.pro
% More: The colors will be red, blue, green, and yellow.
% More: The standard abbrieviations are used to stand for the countries.
% ----------------------------------------------------
% different(X,Y) :: X is not equal to Y

different(red,blue).
different(red,yellow).
different(red,green).
different(blue,red).
different(blue,yellow).
different(blue,green).
```

```
different(yellow,red).
different(yellow,green).
different(yellow,blue).
different(green,red).
different(green,blue).
different(green,yellow).

%-------------------------------------------------------
% coloring (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17) :: the sections
% of the map
% Sections represented by numbers are colored so that none of the
% sections sharing a side are the same color.

coloring(S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,S11,S12,S13,S14,S15,S16,S17) :-
    different(S1,S2),
    different(S1,S3),
    different(S1,S4),
    different(S1,S5),
    different(S2,S12),
    different(S2,S13),
    different(S2,S3),
    different(S2,S5),
    different(S3,S4),
    different(S3,S10),
    different(S3,S11),
    different(S4,S8),
    different(S4,S9),
    different(S5,S6),
    different(S5,S7),
    different(S6,S7),
    different(S6,S13),
    different(S6,S14),
    different(S7,S15),
    different(S7,S8),
    different(S8,S15),
    different(S8,S9),
    different(S9,S17),
    different(S9,S10),
    different(S10,S11),
    different(S10,S17),
    different(S11,S12),
    different(S11,S16),
    different(S12,S13),
    different(S12,S16),
    different(S13,S14),
    different(S14,S15),
    different(S14,S16),
    different(S15,S17),
    different(S16,S17).
```

Demo:

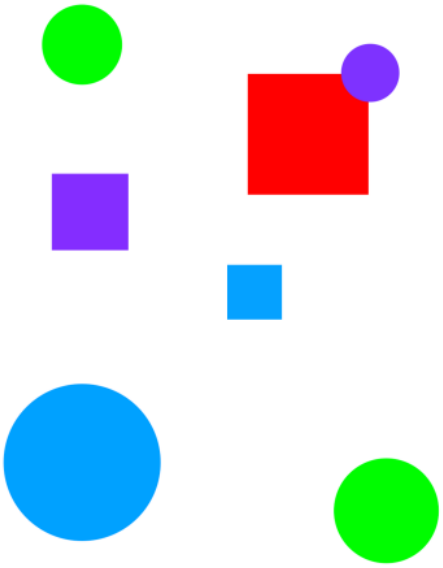
```
?- consult('map_coloring').
true.

?- coloring(S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,S11,S12,S13,S14,S15,S16,S17).
S1 = S6, S6 = S8, S8 = S10, S10 = S12, S12 = red,
S2 = S4, S4 = S14, S14 = blue,
S3 = S5, S5 = S9, S9 = S13, S13 = S15, S15 = S16, S16 = yellow,
S7 = S11, S11 = S17, S17 = green
```

Colored Map:



Task 2: The Floating Shapes World

Image:



Source:

```
%-------------------------------------------
%-------------------------------------------
%---File: shapes_world_1
%---Line: Loosely represented 2-D 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.
```

```
% ---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.
```

Demo:

```
?- consult('shapes_world_1').
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 .
```

Task 3: Pokemon KB Interaction and Programming

Part 1 Demo:

```
?- 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(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,T,_,_),write("(nks("),write(N),write(',kind('),write(T),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 ;
false.
?- pokemon(name(N),_,_,attack(poison-powder,_)).
N = venusaur ;
false.

?- pokemon(_,water,_,attack(Attack,_)),write(Attack),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(name(N),_,_,attack(_,H)),H>60,write(N),nl,fail.
raichu
venusaur
butterfree
charizard
ninetails
false.

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

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

Part 2 Programs:

```
%%_____________________________________________________

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


%%_____________________________________________________

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

%%_____________________________________________________

powerful(N) :-
    pokemon(name(N),_,_,attack(_,D)),
    D>55.

%%_____________________________________________________

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

%%_____________________________________________________
```

```
type(N,T) :-
    pokemon(name(N),T,_,_).

%%_____________________________________________________________

dump_kind(T) :-
    pokemon(name(N),T,hp(H),attack(A,D)),write(pokemon(name(N), T, hp(H), attack(A,D))),nl,fail.

%%_____________________________________________________________

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

%%_____________________________________________________________

family(N1) :-
    evolves(N1,N2), write(N1), write(' '), write(N2), evolves(N2,N3), write(' '), write(N3).

%%_____________________________________________________________

families :-
    cen(N1),evolves(N1,N2),nl, write(N1), write(' '), write(N2), evolves(N2,N3), write(' '), write(N3),fail.

%%_____________________________________________________________

lineage(N) :-
    pokemon(name(N),_,_,_), listing(pokemon(name(N),_,_,_)),
    evolves(N,N2),listing(pokemon(name(N2),_,_,_)),
    evolves(N2,N3), listing(pokemon(name(N3),_,_,_)).
```

Part 2 Demo:

```
?- 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
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
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: Lisp Processing in Prolog

Demo:

```
?- [H|T] = [red, yellow, blue, green]
|    .
H = red,
T = [yellow, blue, green].

?- [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)].
```

Prolog File:

```
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).
```

```
product([],1).
product([Head|Tail],Product) :-
  product(Tail,ProductOfTail),
  Product is Head * ProductOfTail.

%-----------------------------------

factorial(0,0).
factorial(X,R) :-
  iota(X,Iota),
  product(Iota,R).

%-----------------------------------

make_list(0,_,[]).
make_list(X,D,L):-
  K is X - 1,
  make_list(K,D,ListK),
  add_last(D,ListK,L).

%-----------------------------------

but_first([_],[]).
but_first([_|R],R).
```

```
but_last([_],[]).
but_last([H|T], R) :-
  reverse(T, [_|X]), reverse(X, Rlist), add_first(H,Rlist,R).

%-----------------------------------------------------------------

is_palindrome([]).
is_palindrome([_]).
is_palindrome(L) :-
  first(L,First),
  last(L,Last),
  First = Last,
  but_first(L,F),
  but_last(F,R),
  is_palindrome(R).

%-----------------------------------------------------------------

noun_phrase(NP) :-
  pick([tall, loud, cold, dirty, bright, lovely], Adjective),
  pick([desk, computer, ipad, fan, bed, soap, poster, shirt], Noun),
  NP = [the, Adjective, Noun].

%-----------------------------------------------------------------
```

```
sentence(S) :-
  noun_phrase(X),
  noun_phrase(Y),
  pick([drank, listened, texted, washed, slept, worked, jumped], Verb),
  append(X, [Verb],L),
  append(L, Y, S).
```

Example List Processors Demo:

```
?- 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],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(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 = apple .

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

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

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

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

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

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

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

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

?- 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] .
```

List Processing Exercises Demo:

```
?- 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_palindrome([x]).
true .

?- is_palindrome([a,b,c]).
false.

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

?- is_palindrome([1,2,3,4,5,4,3,2,1]).
true .

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

?- noun_phrase(NP).
NP = [the, lovely, fan] .

?- noun_phrase(NP).
NP = [the, cold, computer] .

?- noun_phrase(NP).
NP = [the, bright, soap] .

?- noun_phrase(NP).
NP = [the, tall, ipad] .

?- noun_phrase(NP).
NP = [the, lovely, ipad] .

?- sentence(S).
S = [the, loud, desk, drank, the, dirty, ipad] .

?- sentence(S).
S = [the, loud, bed, drank, the, cold, shirt] .

?- sentence(S).
S = [the, tall, poster, drank, the, dirty, desk] .

?- sentence(S).
S = [the, bright, soap, listened, the, tall, poster] .

?- sentence(S).
S = [the, loud, computer, texted, the, bright, poster] .

?- sentence(S).
S = [the, cold, shirt, worked, the, lovely, ipad] .

?- sentence(S).
S = [the, lovely, bed, slept, the, lovely, bed] .

?- sentence(S).
S = [the, bright, ipad, washed, the, tall, soap] .

?- sentence(S).
S = [the, loud, fan, worked, the, bright, soap] .

?- sentence(S).
S = [the, lovely, fan, listened, the, loud, fan] .

?- sentence(S).
S = [the, loud, bed, listened, the, cold, fan] .

?- sentence(S).
S = [the, loud, shirt, washed, the, bright, desk] .

?- sentence(S).
S = [the, loud, fan, listened, the, lovely, shirt] .

?- sentence(S).
S = [the, cold, ipad, texted, the, bright, shirt] .

?- sentence(S).
S = [the, bright, shirt, drank, the, cold, fan] .

?- sentence(S).
S = [the, lovely, ipad, worked, the, tall, poster] .

?- sentence(S).
S = [the, bright, ipad, jumped, the, lovely, desk] .