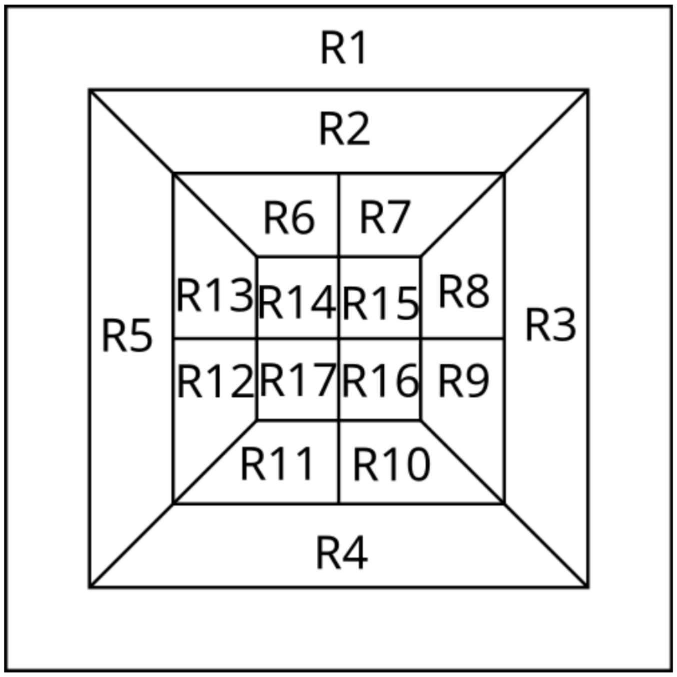
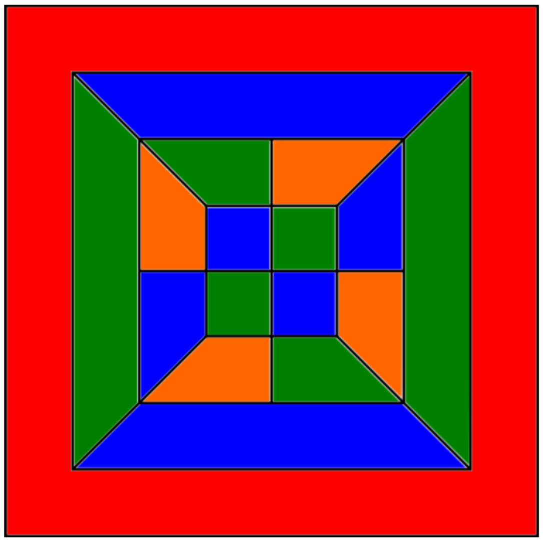
Prolog Assignment #1: Various Computations

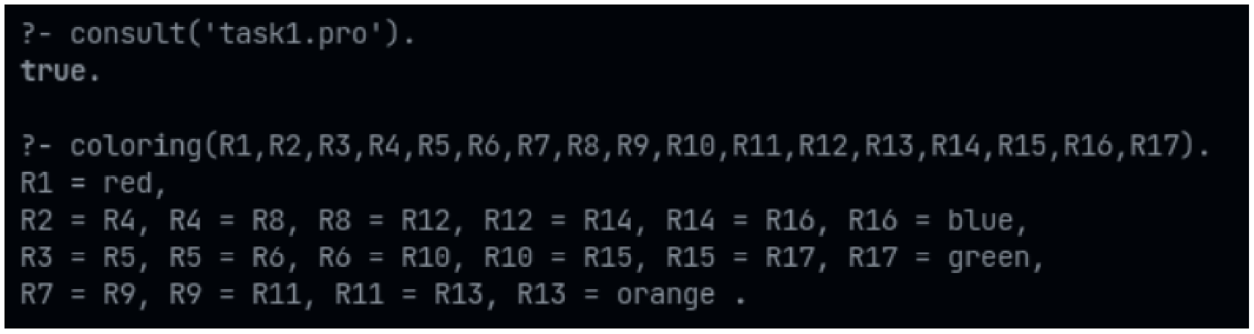
Abstract: Programming exercises that focus on knowledge representation, search, and list processing in Prolog.

Map Coloring:

Map Coloring Demo







```
?- consult('task1.pro').
true.

?- coloring(R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15,R16,R17).
R1 = red,
R2 = R4, R4 = R8, R8 = R12, R12 = R14, R14 = R16, R16 = blue,
R3 = R5, R5 = R6, R6 = R10, R10 = R15, R15 = R17, R17 = green,
R7 = R9, R9 = R11, R11 = R13, R13 = orange .
```

Map Coloring:

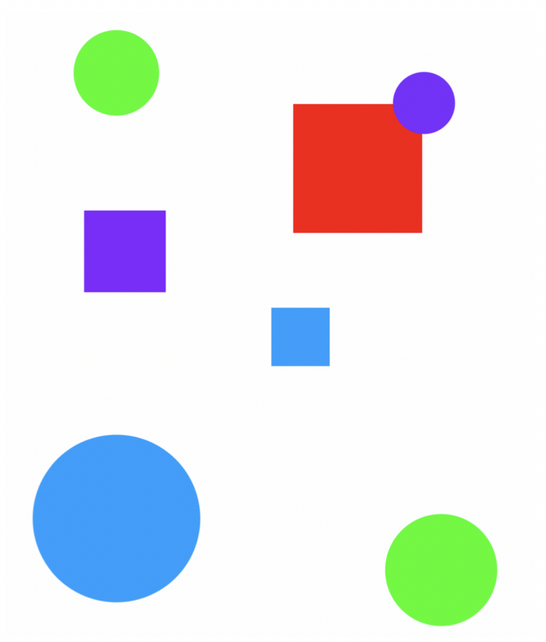
Map Coloring Code

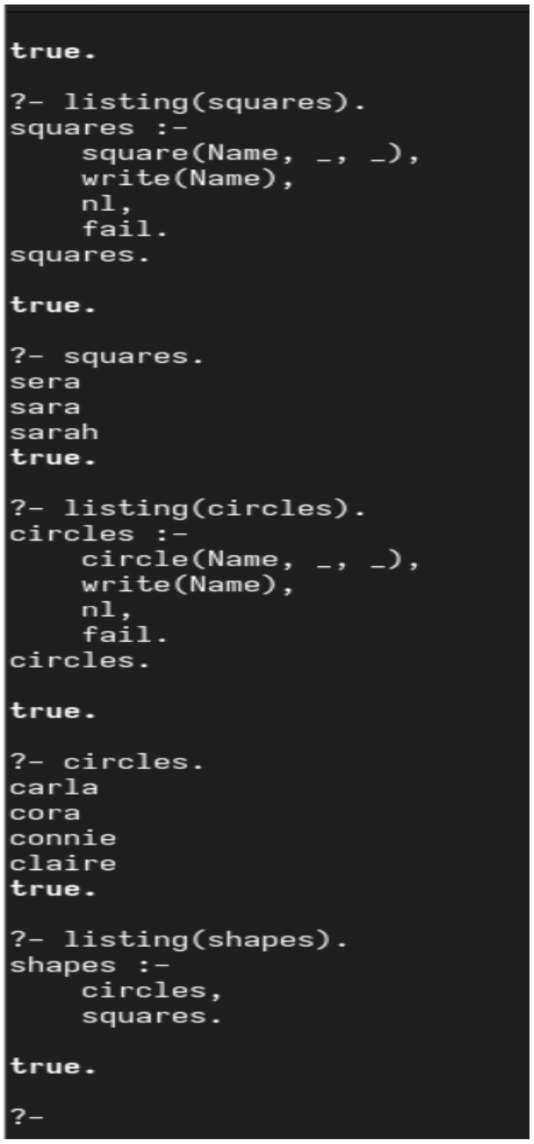
```
% different(X, Y) :: X is not equal to Y

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(R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15,R16,R17) :-
    different(R1,R2),
    different(R1,R3),
    different(R1,R4),
    different(R1,R5),
    different(R2,R3),
    different(R2,R5),
    different(R2,R6),
    different(R2,R7),
    different(R3,R8),
    different(R3,R9),
    different(R3,R4),
    different(R4,R10),
    different(R4,R11),
```

```
    different(R4,R5),
    different(R5,R12),
    different(R5,R13),
    different(R6,R7),
    different(R6,R14),
    different(R6,R13),
    different(R13,R14),
    different(R13,R12),
    different(R12,R17),
    different(R12,R11),
    different(R11,R17),
    different(R11,R10),
    different(R10,R16),
    different(R10,R9),
    different(R9,R16),
    different(R9,R8),
    different(R8,R15),
    different(R8,R7),
    different(R7,R15),
    different(R15,R14),
    different(R15,R16),
    different(R16,R17),
    different(R17,R14).
```

Floating Shapes World:

Floating Shapes World Demo

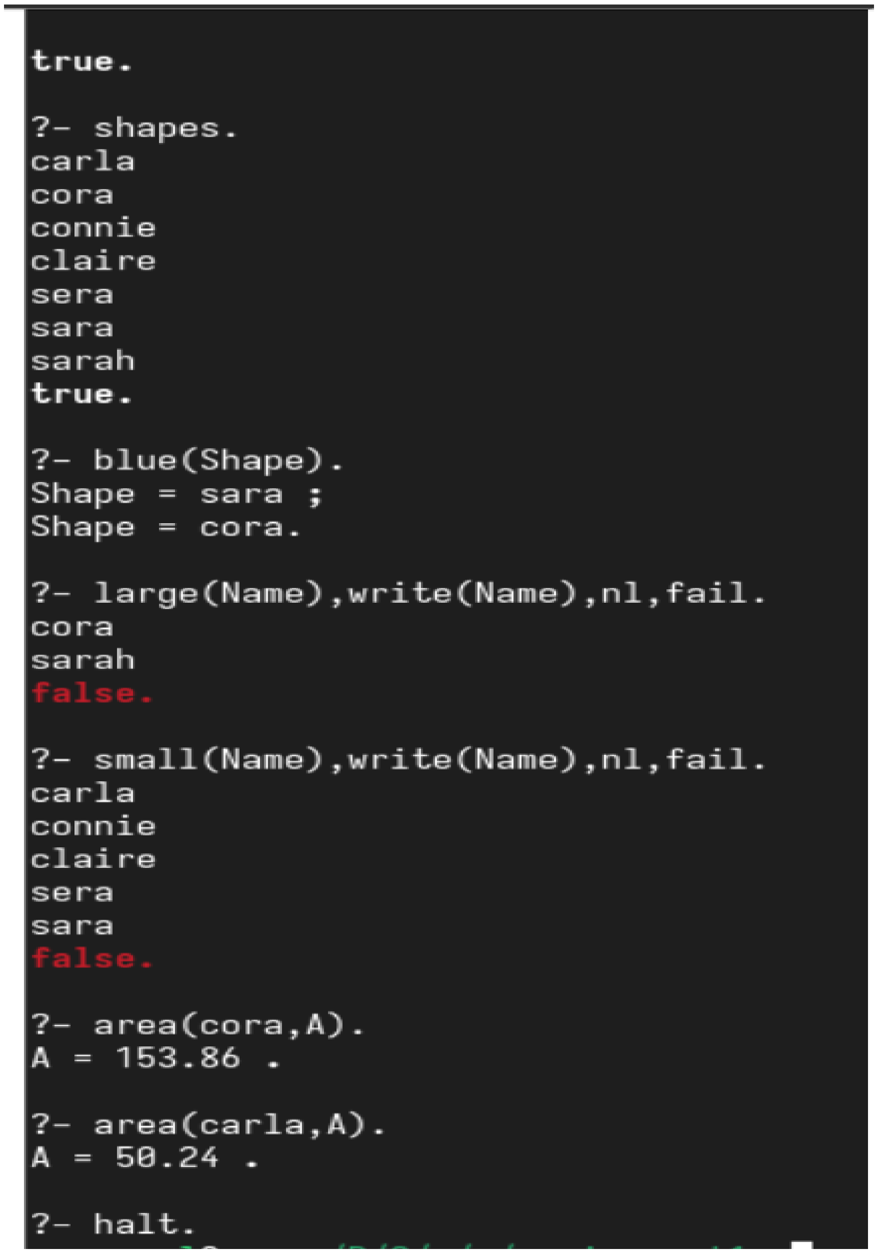




```
true.

?- 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.

?-
```

Floating Shapes World:

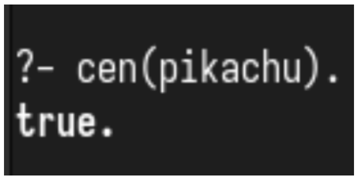
Floating Shapes World Code



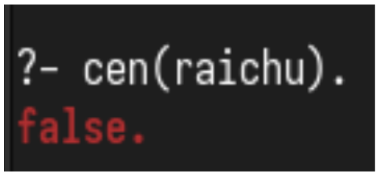
```
true.

?- 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.
```

Pokemon KB Interaction and Programing:

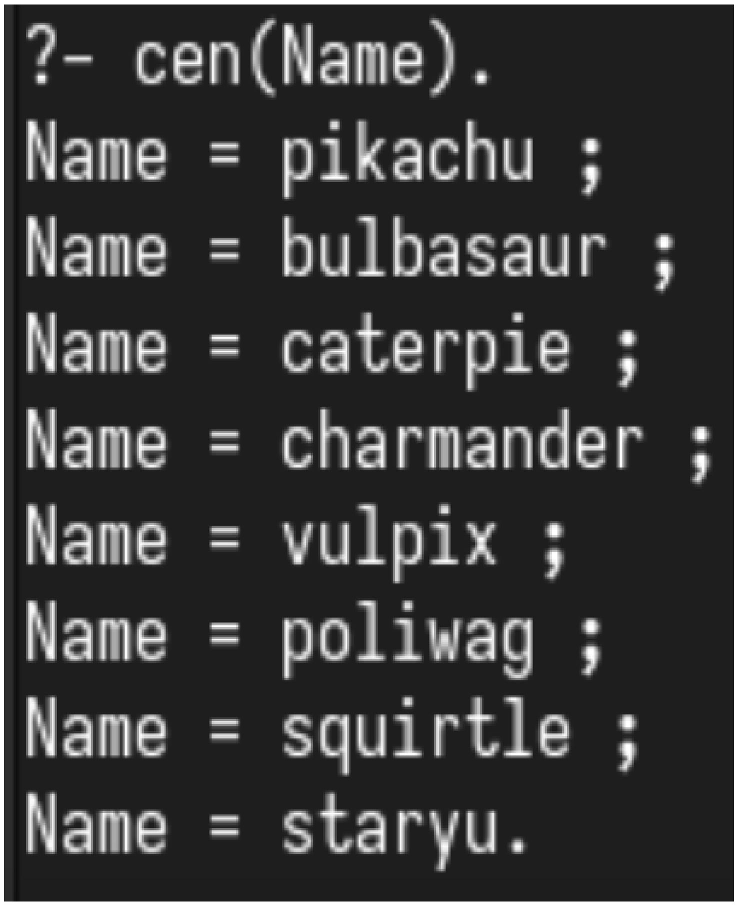
Part 1: Queries (I tried to do one at a time)



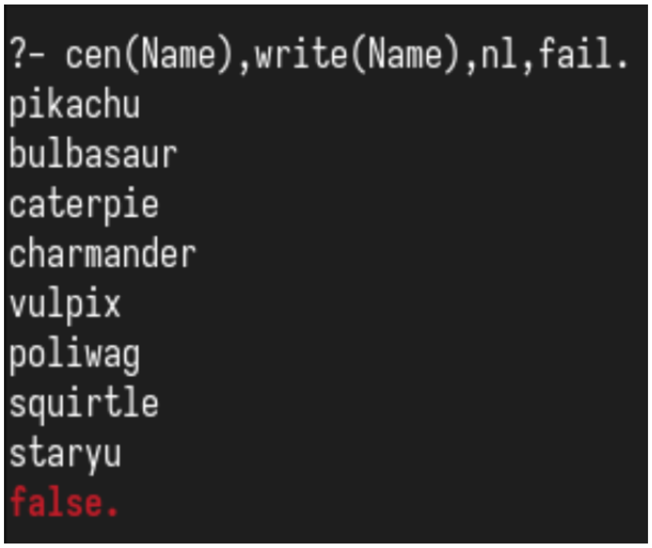
```
?- cen(pikachu).
true.
```



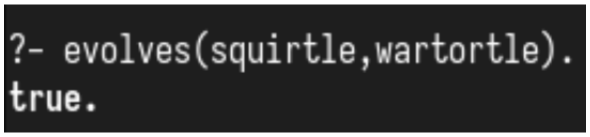
```
?- cen(raichu).
false.
```



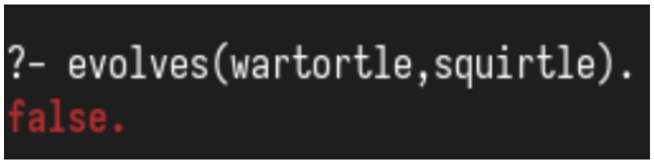
```
?- cen(Name).
Name = pikachu ;
Name = bulbasaur ;
Name = caterpie ;
Name = charmander ;
Name = vulpix ;
Name = poliwag ;
Name = squirtle ;
Name = staryu.
```



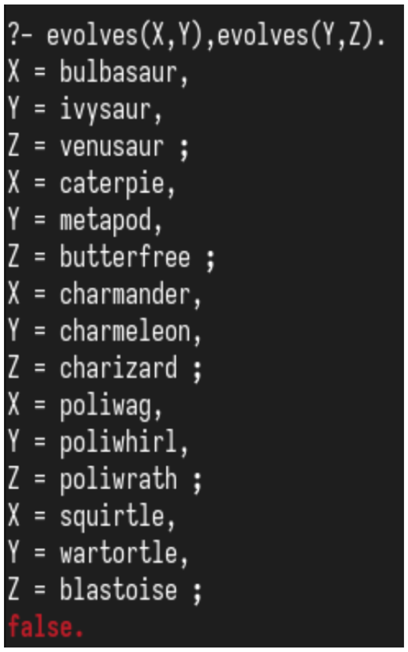
```
?- cen(Name),write(Name),nl,fail.
pikachu
bulbasaur
caterpie
charmander
vulpix
poliwag
squirtle
staryu
false.
```



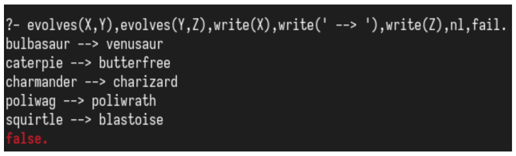
```
?- evolves(squirtle,wartortle).
true.
```



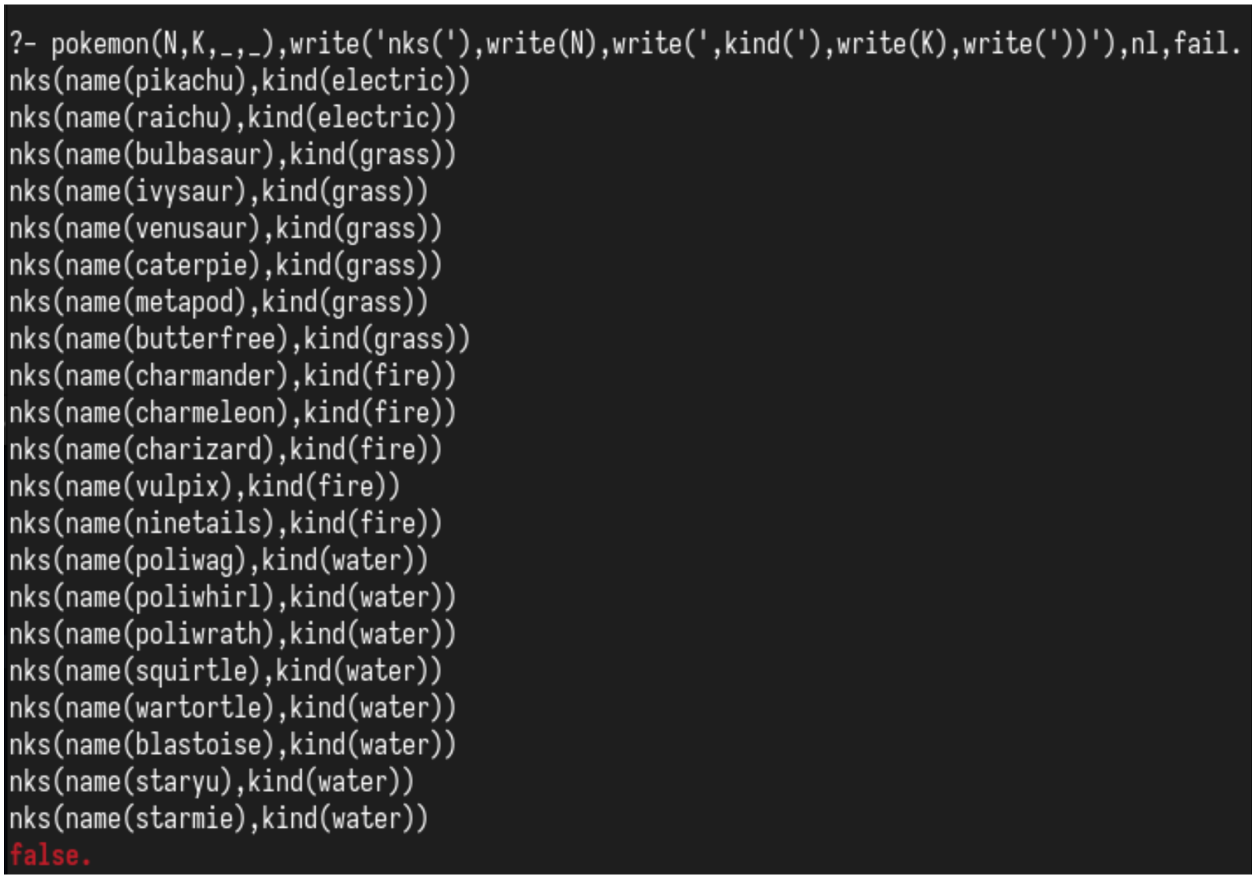
```
?- evolves(wartortle,squirtle).
false.
```



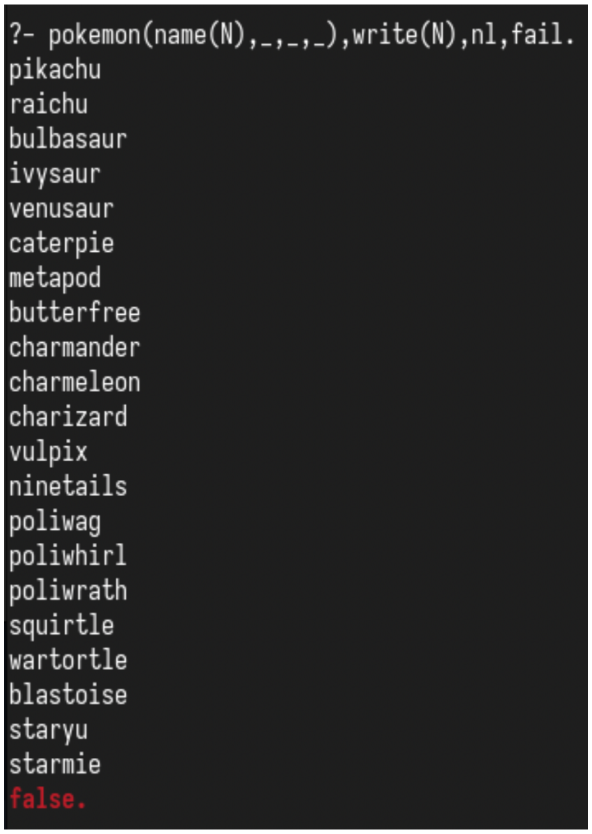
```
?- evolves(X,Y),evolves(Y,Z).
X = bulbasaur,
Y = ivysaur,
Z = venusaur ;
X = caterpie,
Y = metapod,
Z = butterfree ;
X = charmander,
Y = charmeleon,
Z = charizard ;
X = poliwag,
Y = poliwhirl,
Z = poliwrath ;
X = squirtle,
Y = wartortle,
Z = blastoise ;
false.
```



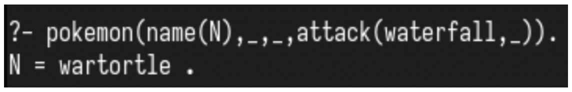
```
?- evolves(X,Y),evolves(Y,Z),write(X),write(' --> '),write(Z),nl,fail.
bulbasaur --> venusaur
caterpie --> butterfree
charmander --> charizard
poliwag --> poliwrath
squirtle --> blastoise
false.
```



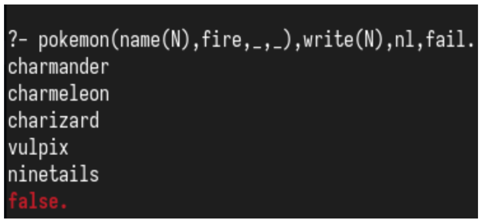
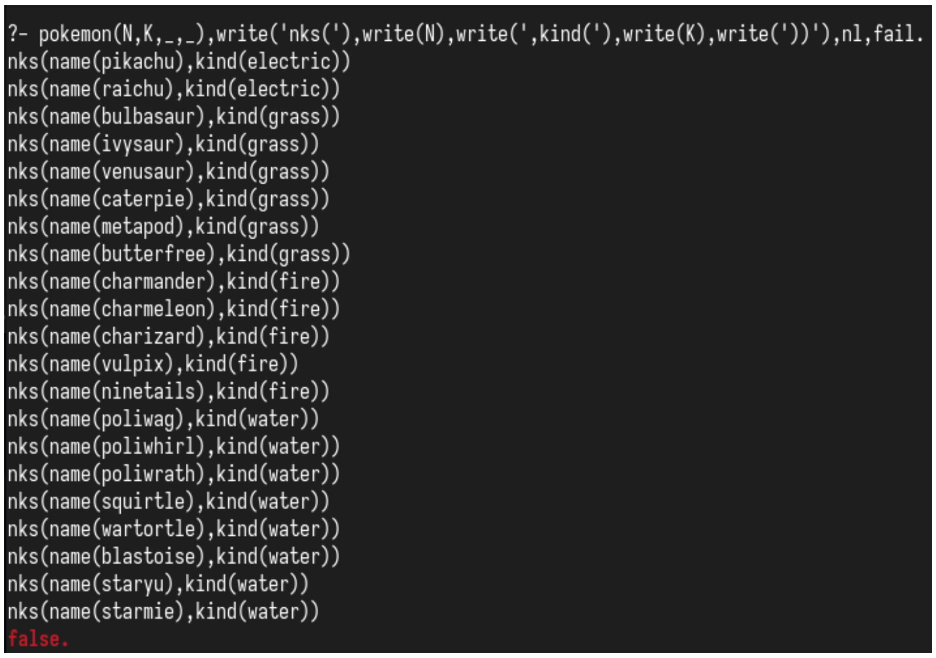
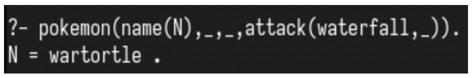
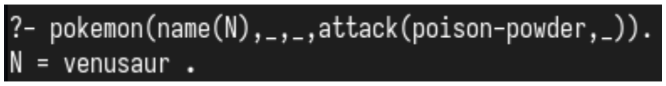
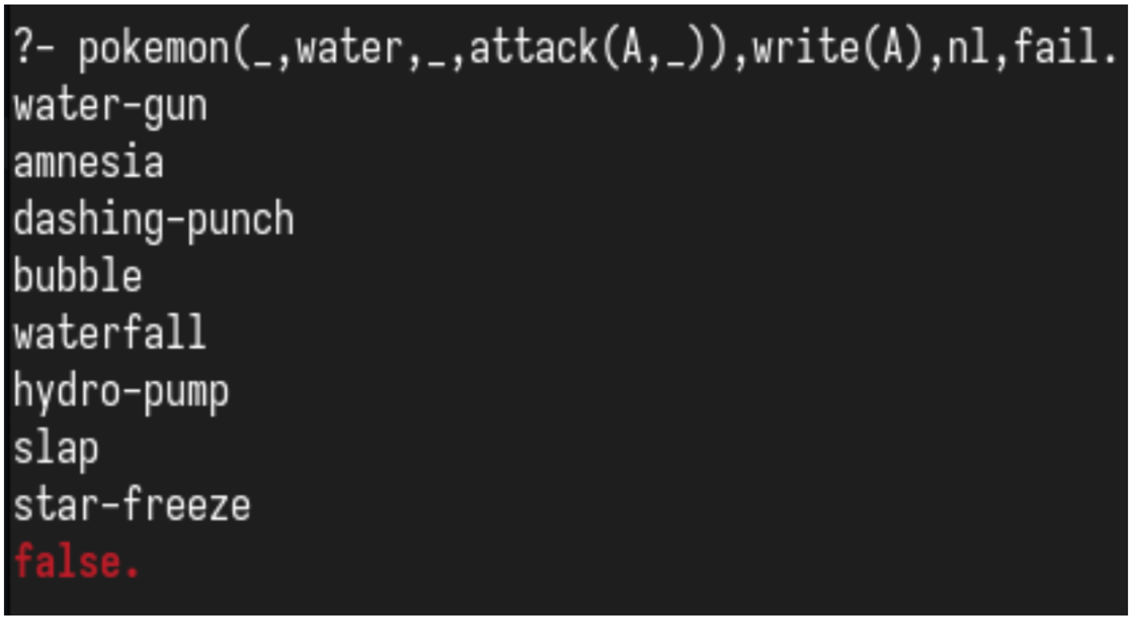
```
?- pokemon(N,K,_,_),write('nks('),write(N),write(',kind('),write(K),write('))'),nl,fail.
nks(name(pikachu),kind(electric))
nks(name(raichu),kind(electric))
nks(name(bulbasaur),kind(grass))
nks(name(ivysaur),kind(grass))
nks(name(venusaur),kind(grass))
nks(name(caterpie),kind(grass))
nks(name(metapod),kind(grass))
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.
```

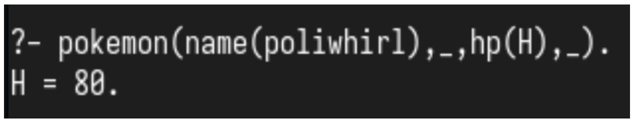
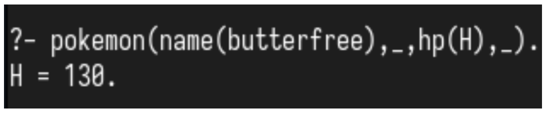
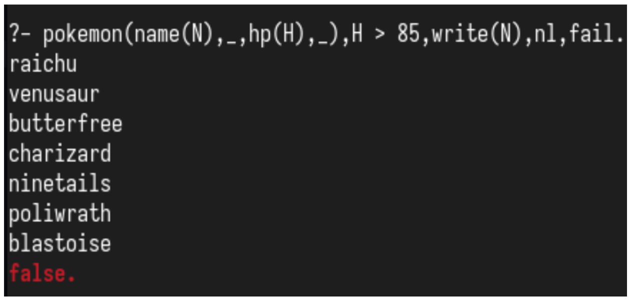
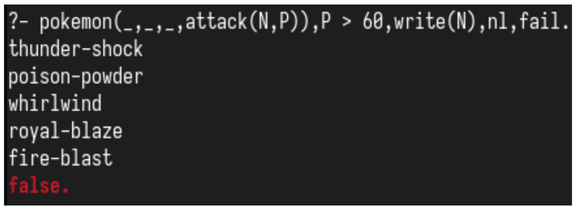
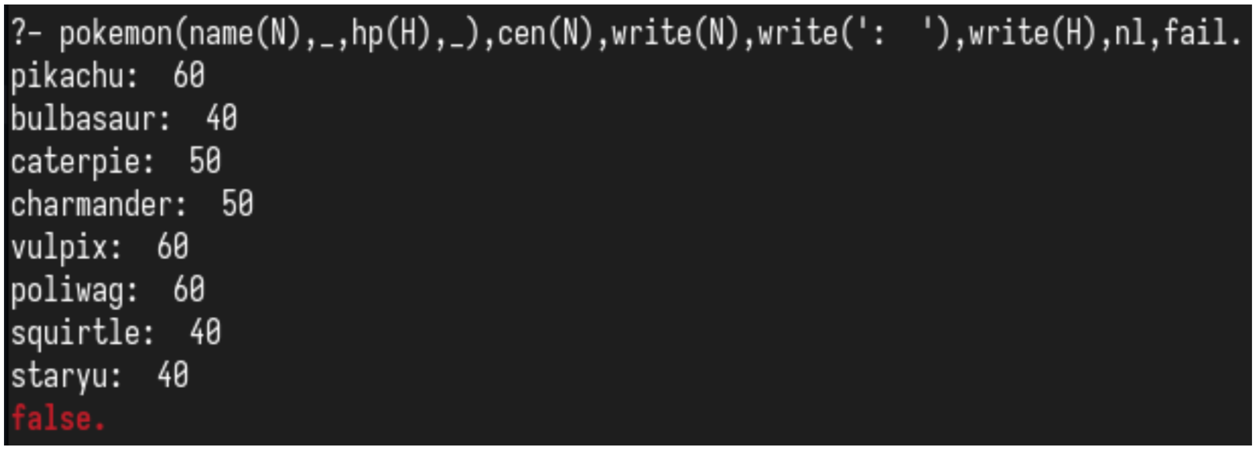


```
?- pokemon(name(N),_,_,_),write(N),nl,fail.
pikachu
raichu
bulbasaur
ivysaur
venusaur
caterpie
metapod
butterfree
charmander
charmeleon
charizard
vulpix
ninetails
poliwag
poliwhirl
poliwrath
squirtle
wartortle
blastoise
staryu
starmie
false.
```



```
?- pokemon(name(N),_,_,attack(waterfall,_)).
N = wartortle .
```


```
?- 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.
```

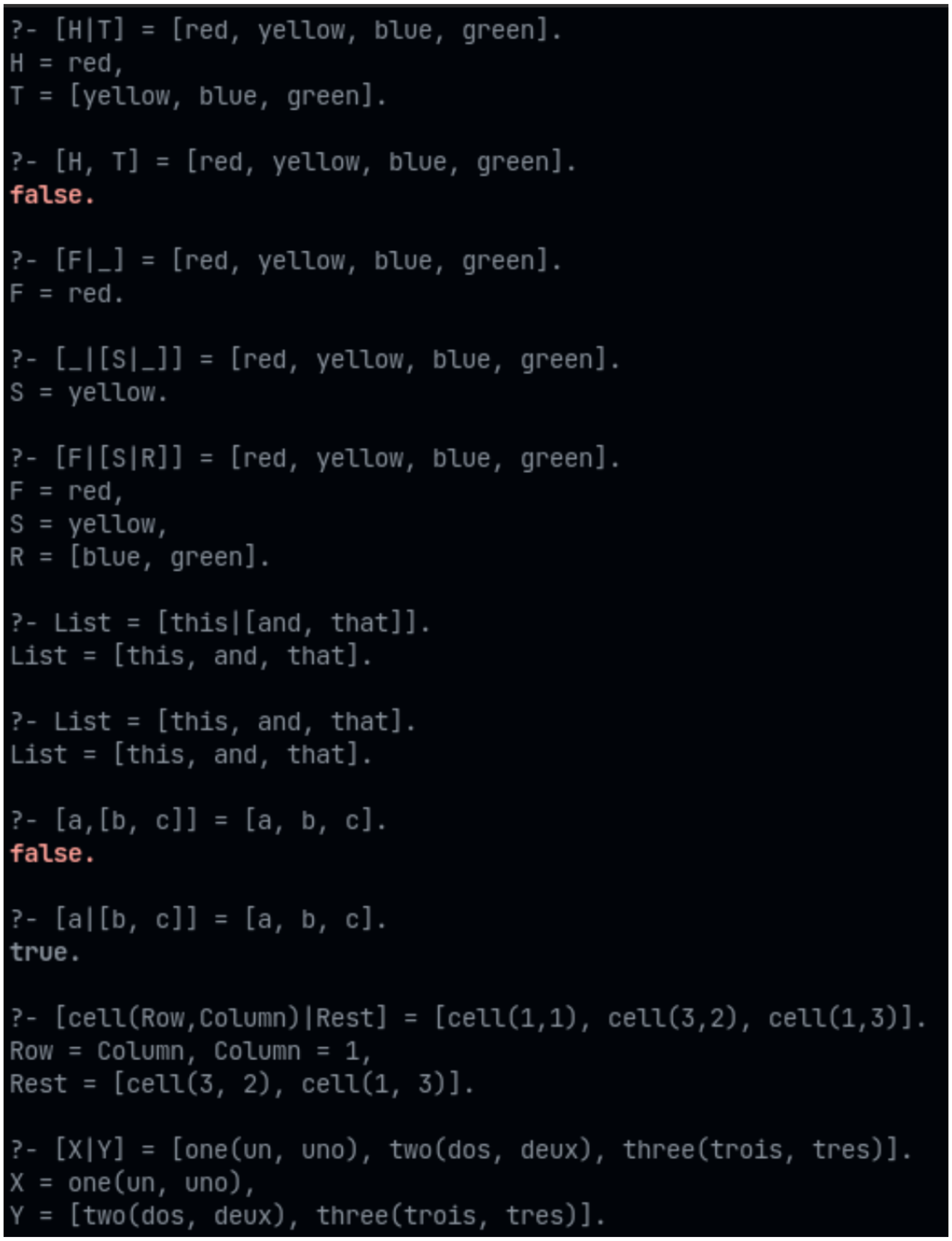
Pokemon KB Interaction and Programing:

Part 2: Extended Knowledge Base

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

% assignment additions

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

Lisp Processing in Prolog:

Lisp Processing in Prolog Demo:



```
?- [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)].
```

Lisp Processing in Prolog:

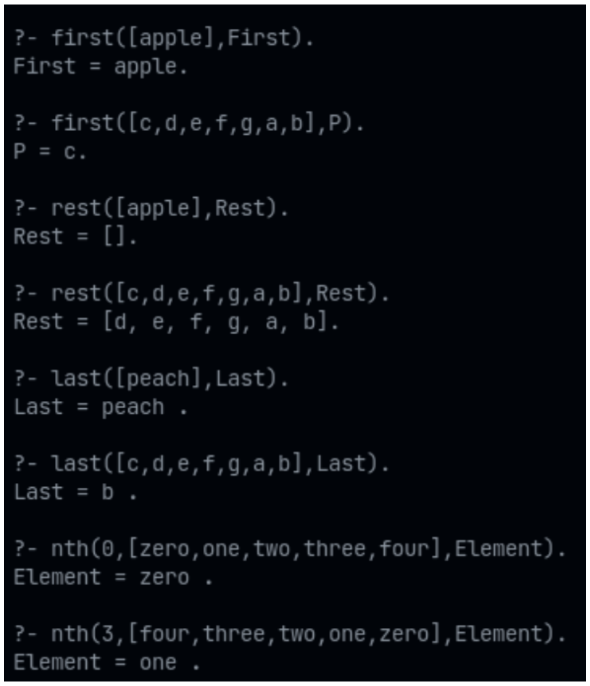
Lisp Processing in Prolog Code:

```
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).

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).
```

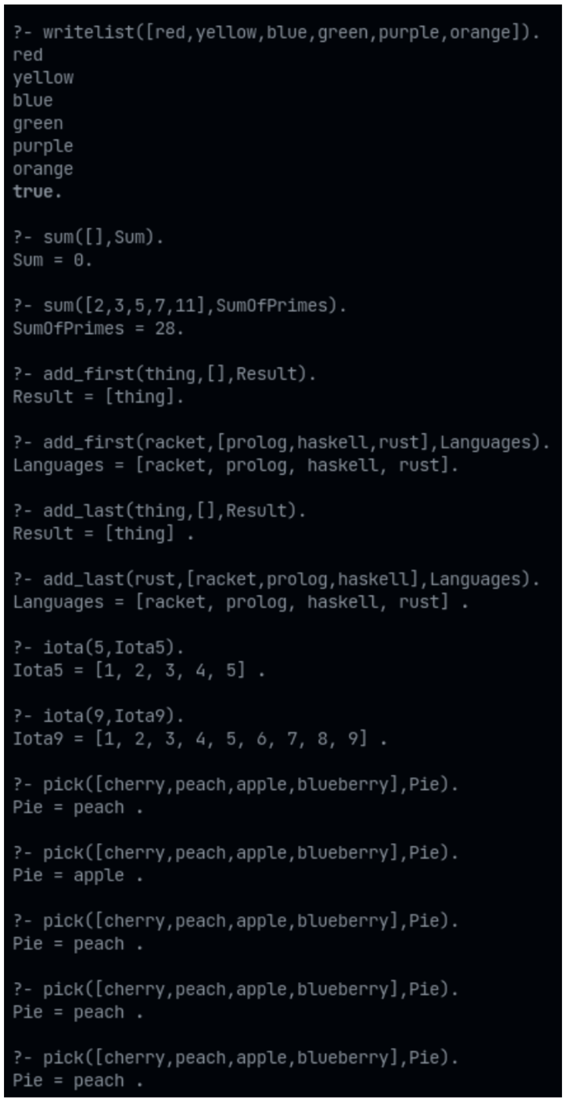
```
product([], 1).
product([Head|Tail], Product) :-
    product(Tail,ProductOfTail),
    Product is Head * ProductOfTail.

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([crazy,inane,odd,wacky,confused,tired]).

nouns([robot,terminator,alien,lamppost,phone,blob,ai,fan]).

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).
```

Lisp Processing in Prolog:

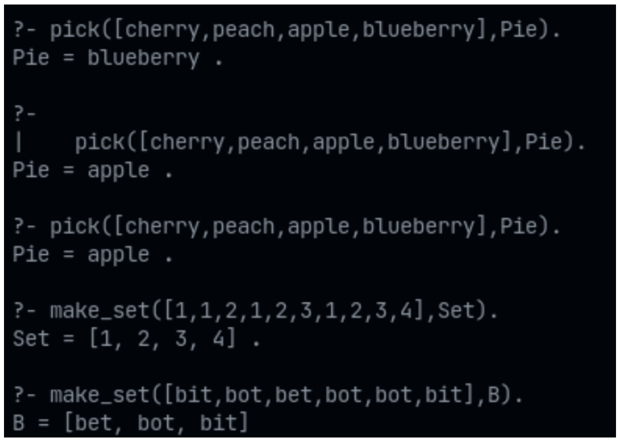
Example Lisp processors and List Processing Exercises:



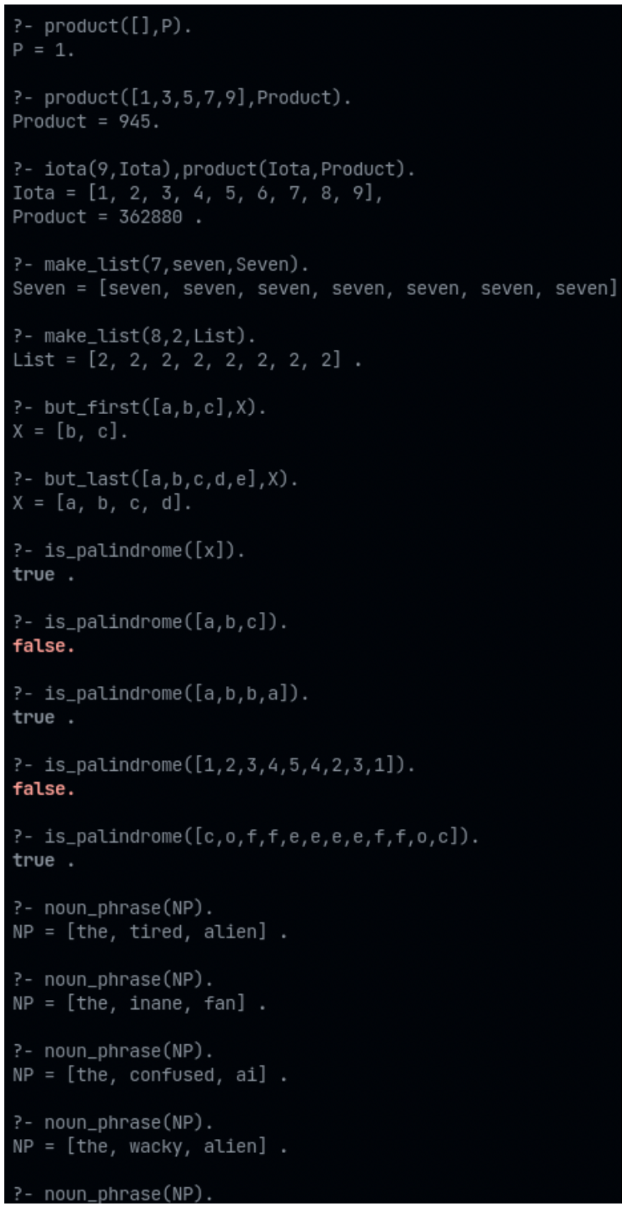
```
?- first([apple],First).
First = apple.

?- 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 .
```



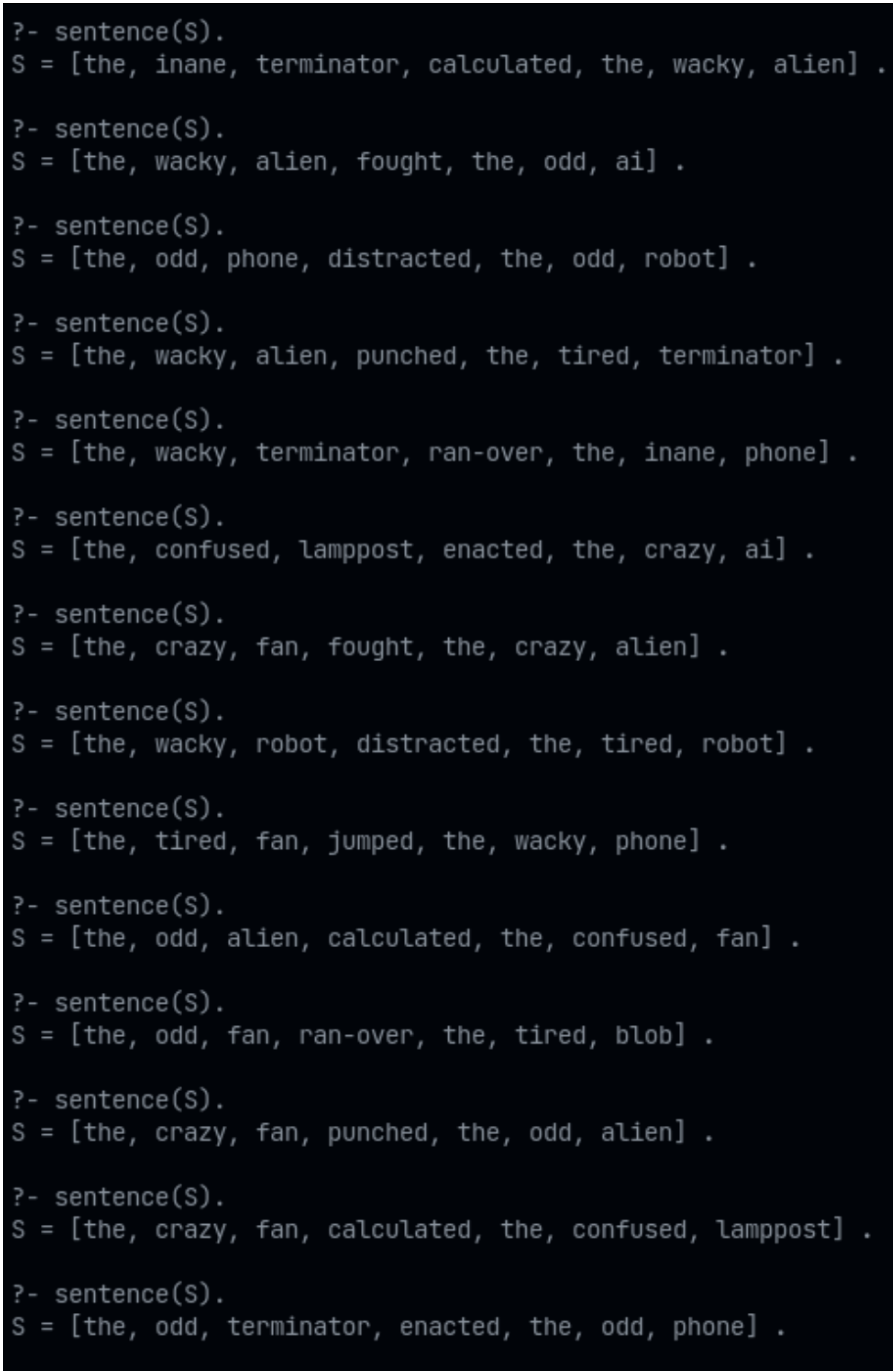
```
?- writelist([red,yellow,blue,green,purple,orange]).
red
yellow
blue
green
purple
orange
true.

?- 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 = peach .

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

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

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

?- 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]
```



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

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

?- noun_phrase(NP).
NP = [the, tired, alien] .

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

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

?- noun_phrase(NP).
NP = [the, wacky, alien] .

?- noun_phrase(NP).
```



```
?- sentence(S).
S = [the, inane, terminator, calculated, the, wacky, alien] .

?- sentence(S).
S = [the, wacky, alien, fought, the, odd, ai] .

?- sentence(S).
S = [the, odd, phone, distracted, the, odd, robot] .

?- sentence(S).
S = [the, wacky, alien, punched, the, tired, terminator] .

?- sentence(S).
S = [the, wacky, terminator, ran-over, the, inane, phone] .

?- sentence(S).
S = [the, confused, lamppost, enacted, the, crazy, ai] .

?- sentence(S).
S = [the, crazy, fan, fought, the, crazy, alien] .

?- sentence(S).
S = [the, wacky, robot, distracted, the, tired, robot] .

?- sentence(S).
S = [the, tired, fan, jumped, the, wacky, phone] .

?- sentence(S).
S = [the, odd, alien, calculated, the, confused, fan] .

?- sentence(S).
S = [the, odd, fan, ran-over, the, tired, blob] .

?- sentence(S).
S = [the, crazy, fan, punched, the, odd, alien] .

?- sentence(S).
S = [the, crazy, fan, calculated, the, confused, lamppost] .

?- sentence(S).
S = [the, odd, terminator, enacted, the, odd, phone] .
```