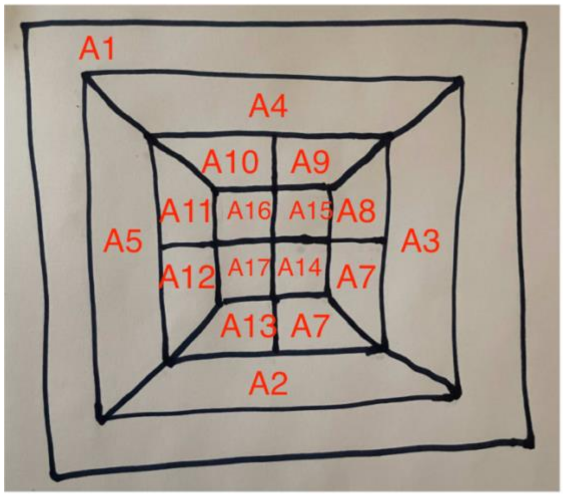
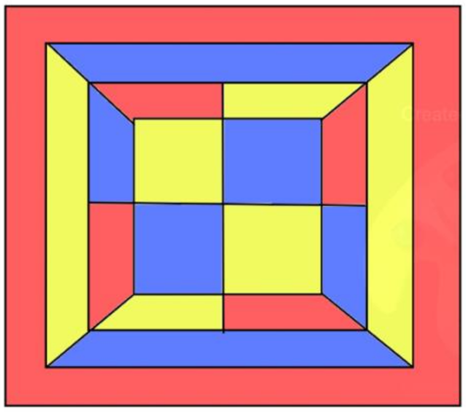
Prolog Programming Assignment : Various Computations

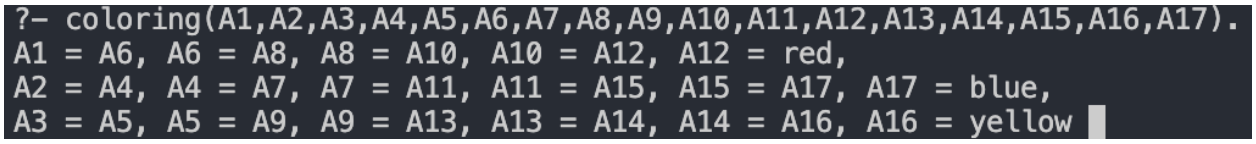
Learning Abstract

This is the only programming assignment on Prolog and it features programming exercises that focus on knowledge representation, search, and list processing in Prolog.

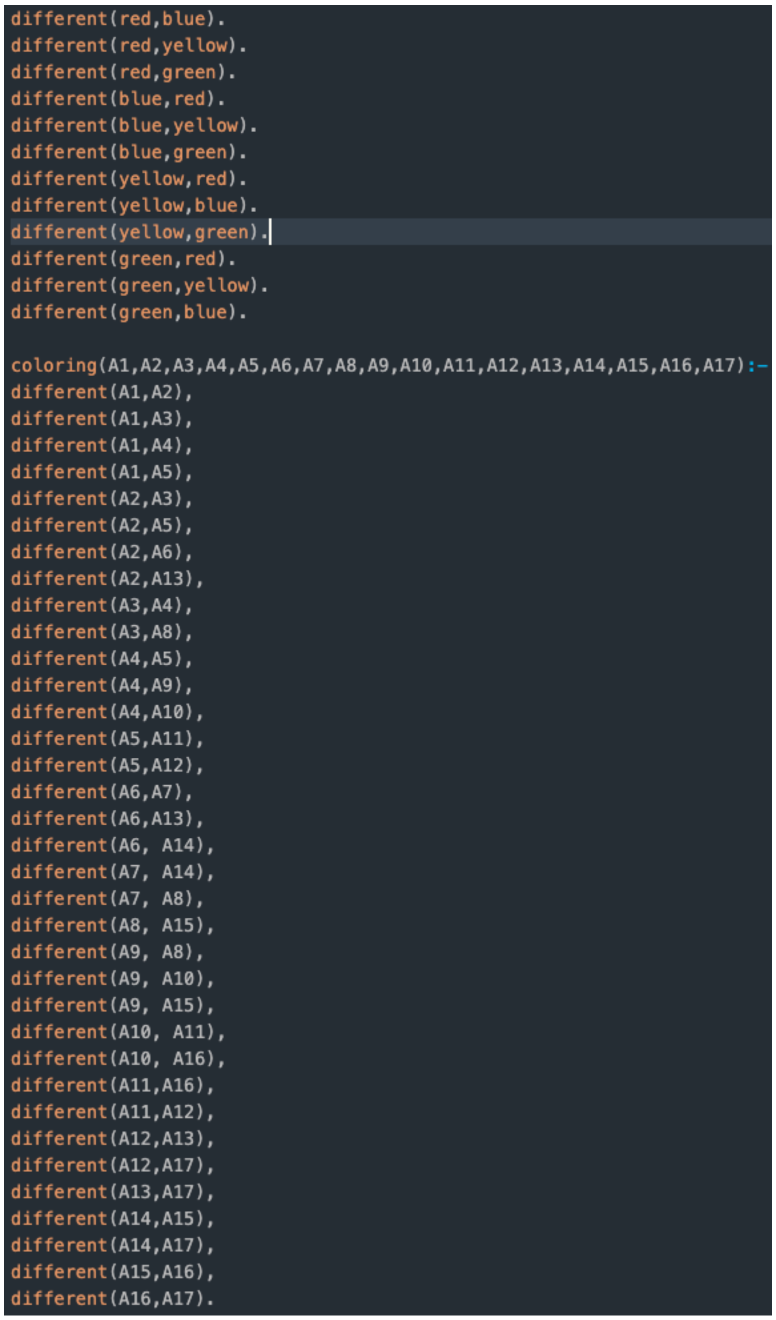
Task 1: Map Coloring





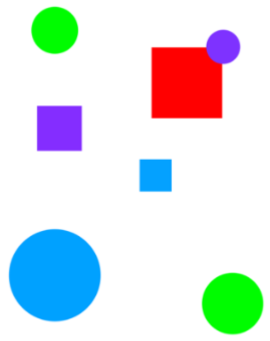


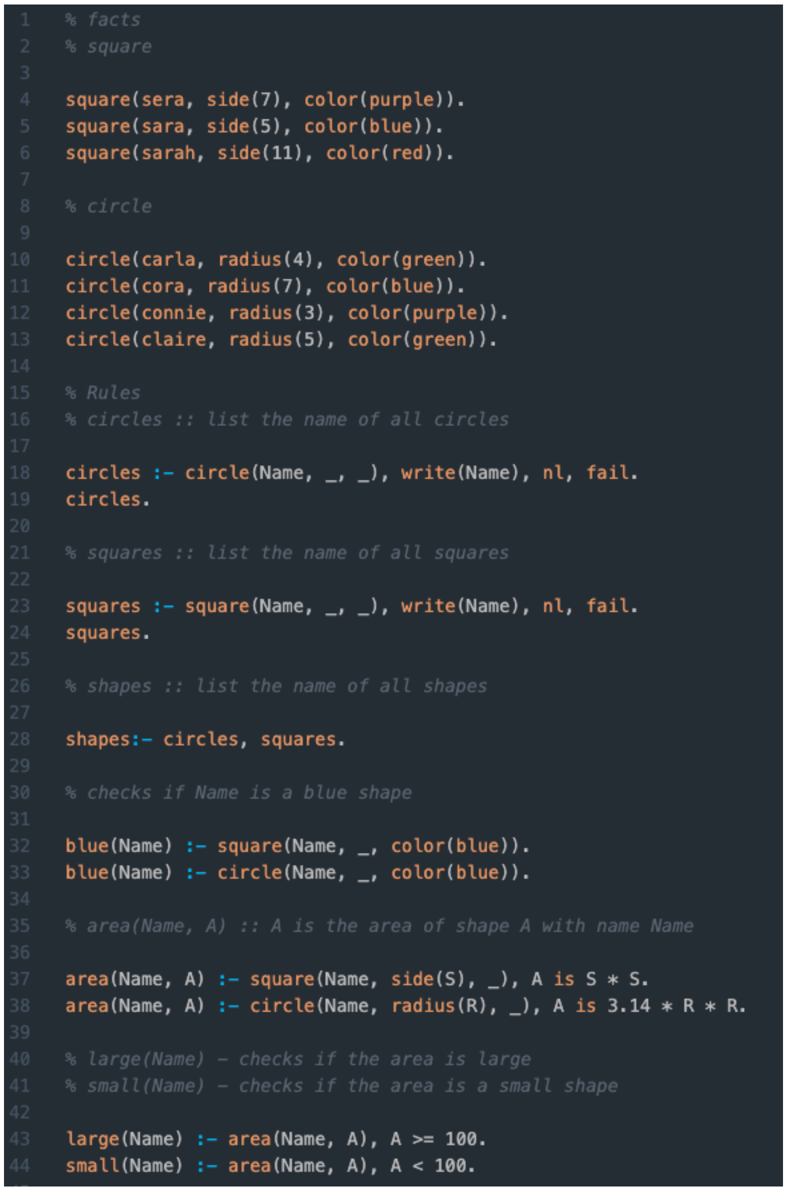
```
?- coloring(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,A15,A16,A17).
A1 = A6, A6 = A8, A8 = A10, A10 = A12, A12 = red,
A2 = A4, A4 = A7, A7 = A11, A11 = A15, A15 = A17, A17 = blue,
A3 = A5, A5 = A9, A9 = A13, A13 = A14, A14 = A16, A16 = yellow
```



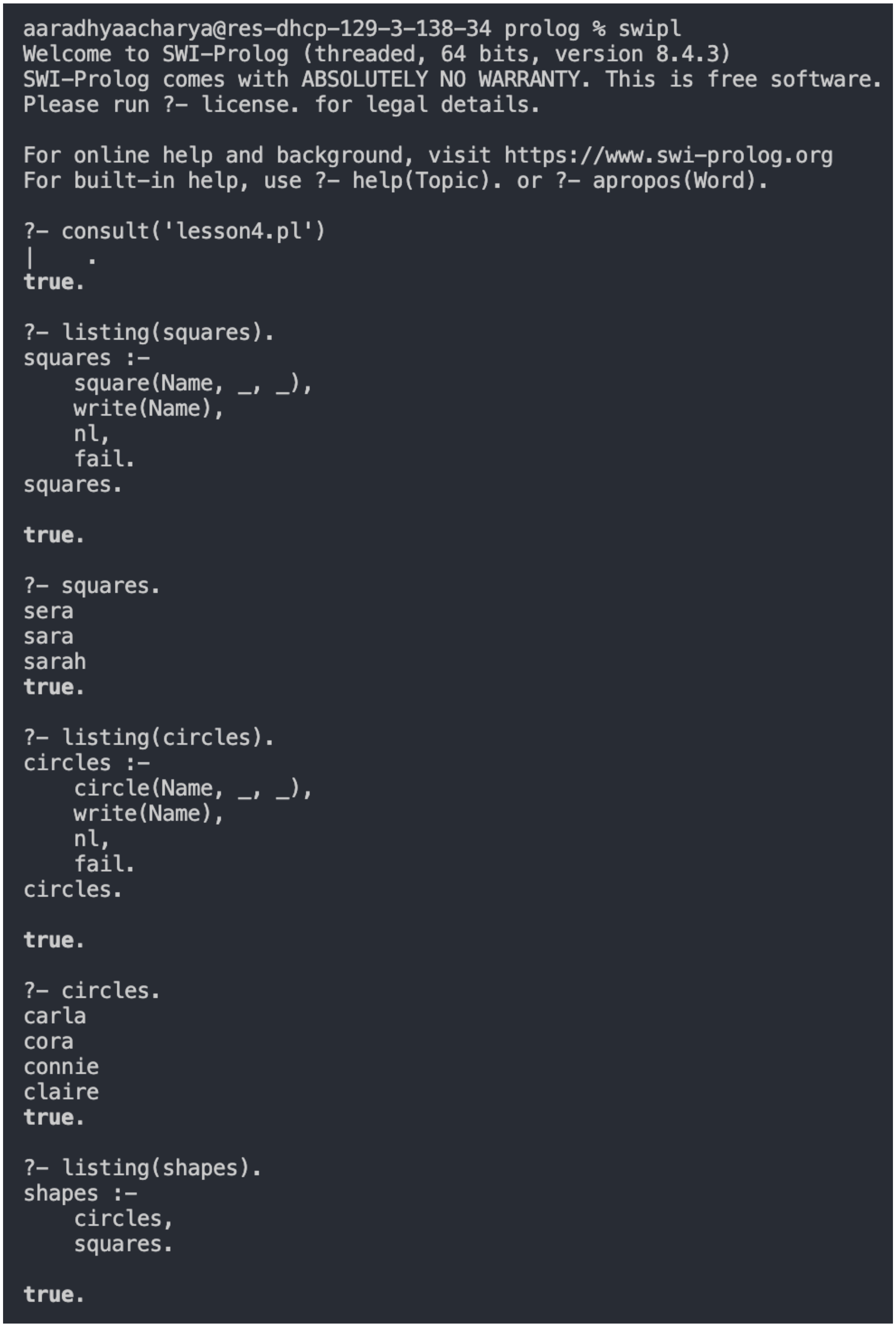
```
different(red,blue).
different(red,yellow).
different(red,green).
different(blue,red).
different(blue,yellow).
different(blue,green).
different(yellow,red).
different(yellow,blue).
different(yellow,green).
different(green,red).
different(green,yellow).
different(green,blue).

coloring(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,A15,A16,A17):-
different(A1,A2),
different(A1,A3),
different(A1,A4),
different(A1,A5),
different(A2,A3),
different(A2,A5),
different(A2,A6),
different(A2,A13),
different(A3,A4),
different(A3,A8),
different(A4,A5),
different(A4,A9),
different(A4,A10),
different(A5,A11),
different(A5,A12),
different(A6,A7),
different(A6,A13),
different(A6, A14),
different(A7, A14),
different(A7, A8),
different(A8, A15),
different(A9, A8),
different(A9, A10),
different(A9, A15),
different(A10, A11),
different(A10, A16),
different(A11,A16),
different(A11,A12),
different(A12,A13),
different(A12,A17),
different(A13,A17),
different(A14,A15),
different(A14,A17),
different(A15,A16),
different(A16,A17).
```

Task 2: The Floating Shapes World

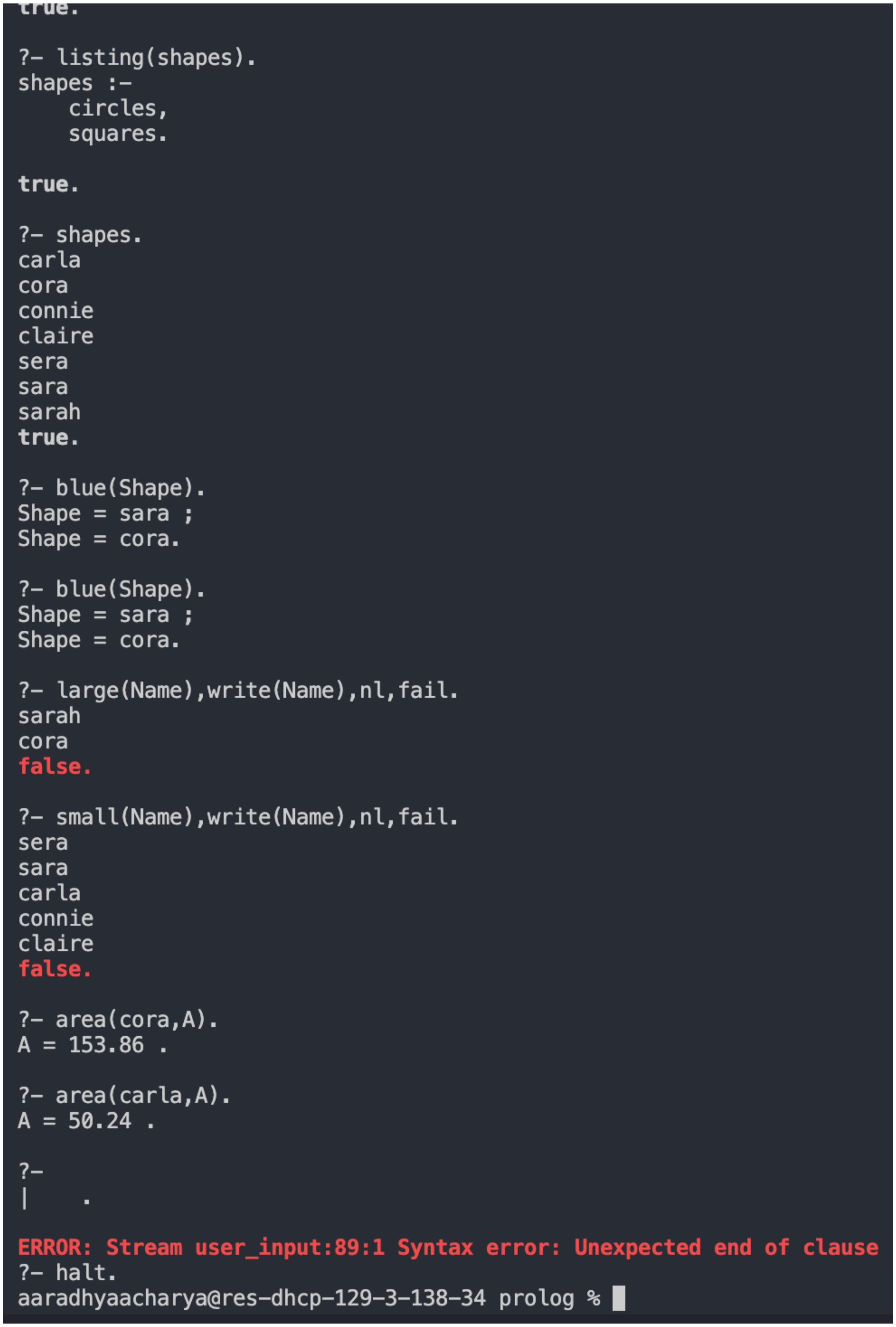




```prolog
% facts
% square

square(sera, side(7), color(purple)).
square(sara, side(5), color(blue)).
square(sarah, side(11), color(red)).

% circle

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 name of all circles

circles :- circle(Name, _, _), write(Name), nl, fail.
circles.

% squares :: list the name of all squares

squares :- square(Name, _, _), write(Name), nl, fail.
squares.

% shapes :: list the name of all shapes

shapes:- circles, squares.

% checks if Name is a blue shape

blue(Name) :- square(Name, _, color(blue)).
blue(Name) :- circle(Name, _, color(blue)).

% area(Name, A) :: A is the area of shape A with name Name

area(Name, A) :- square(Name, side(S), _), A is S * S.
area(Name, A) :- circle(Name, radius(R), _), A is 3.14 * R * R.

% large(Name) - checks if the area is large
% small(Name) - checks if the area is a small shape

large(Name) :- area(Name, A), A >= 100.
small(Name) :- area(Name, A), A < 100.
```

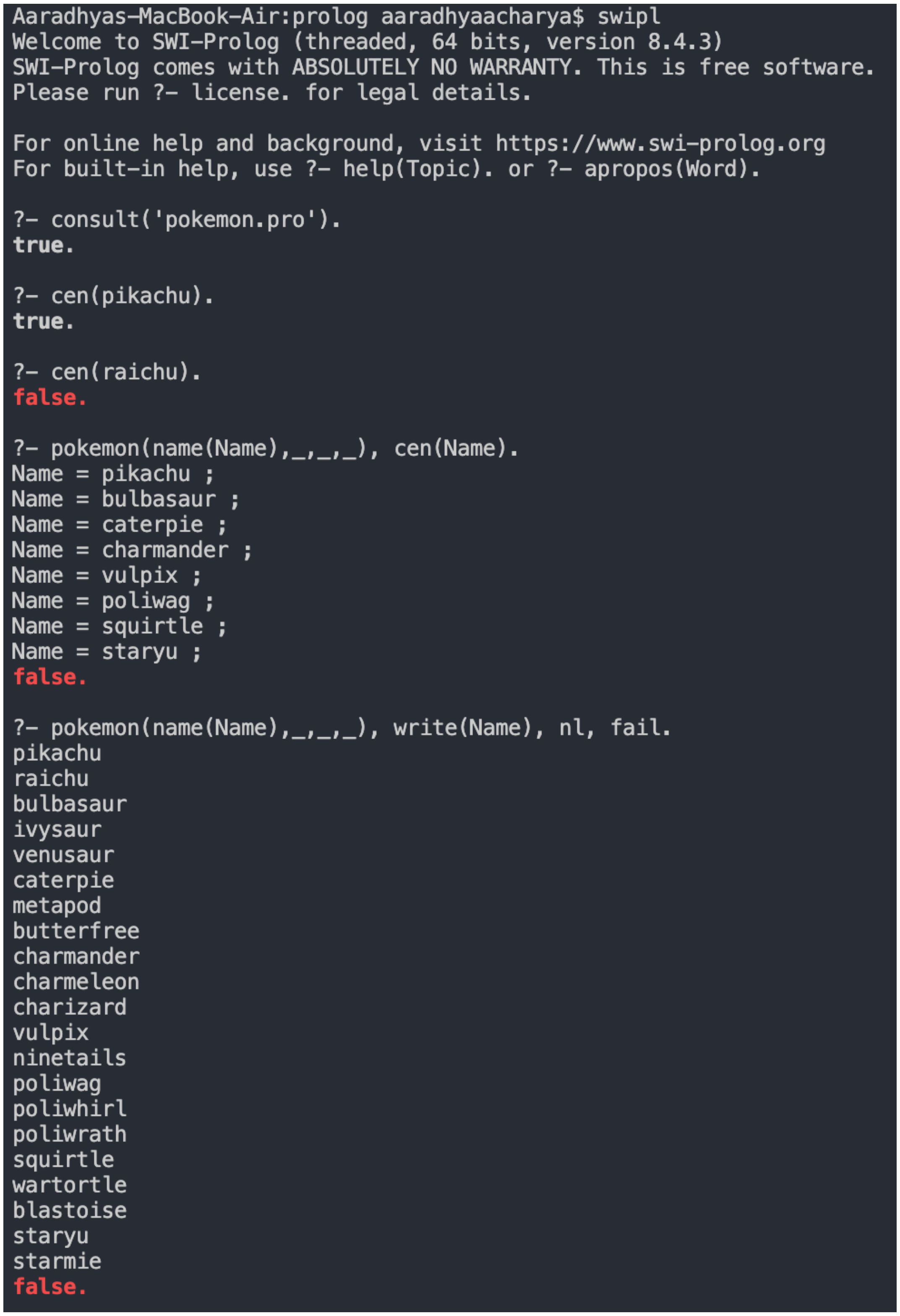


```
aaradhyaacharya@res-dhcp-129-3-138-34 prolog % swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

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

?- consult('lesson4.pl')
|    .
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.
```

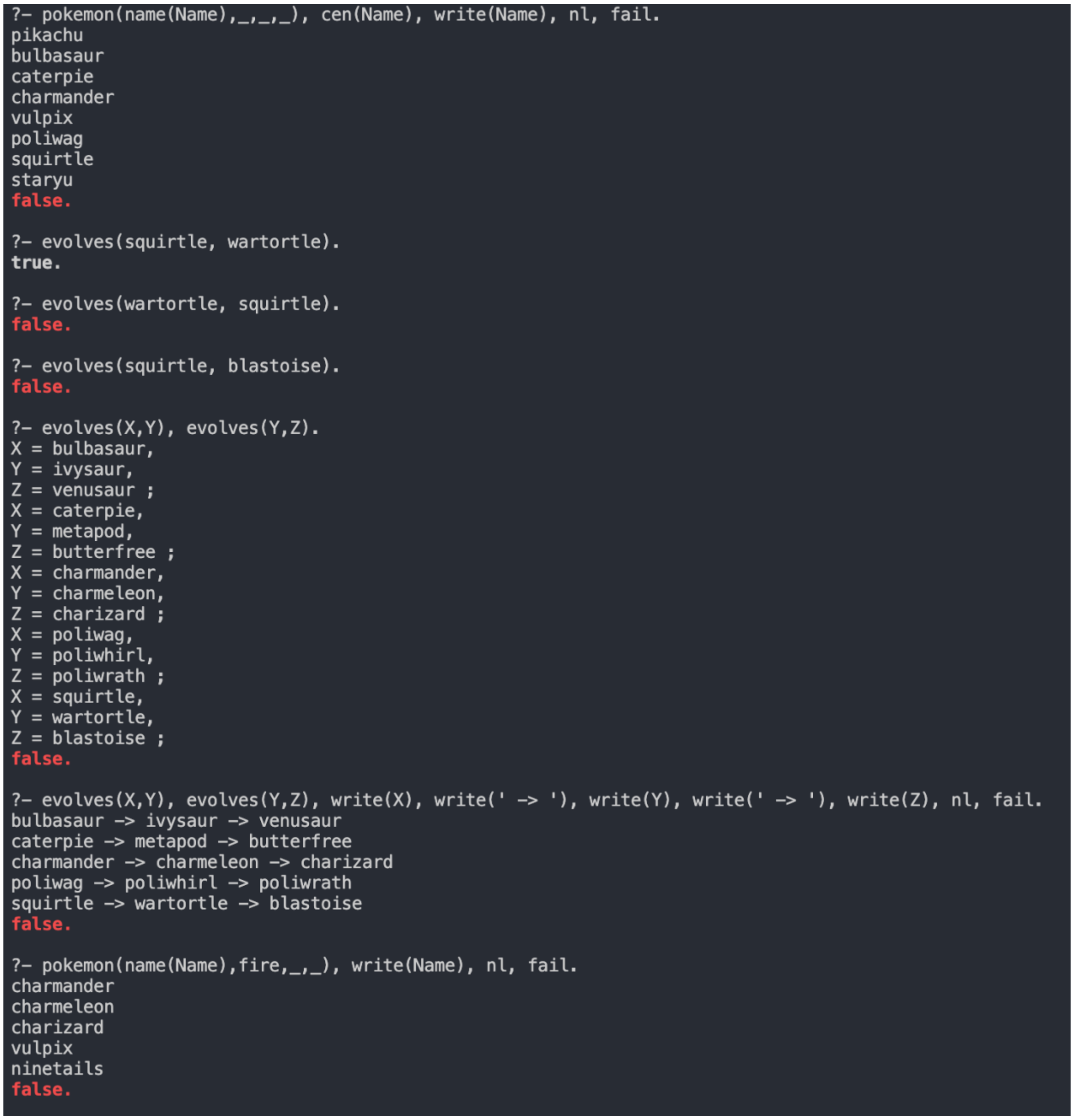


```
true.

?- listing(shapes).
shapes :-
    circles,
    squares.

true.

?- shapes.
carla
cora
connie
claire
sera
sara
sarah
true.

?- blue(Shape).
Shape = sara ;
Shape = cora.

?- blue(Shape).
Shape = sara ;
Shape = cora.

?- large(Name),write(Name),nl,fail.
sarah
cora
false.

?- small(Name),write(Name),nl,fail.
sera
sara
carla
connie
claire
false.

?- area(cora,A).
A = 153.86 .

?- area(carla,A).
A = 50.24 .

?-
|    .

ERROR: Stream user_input:89:1 Syntax error: Unexpected end of clause
?- halt.
aaradhyaacharya@res-dhcp-129-3-138-34 prolog % 
```

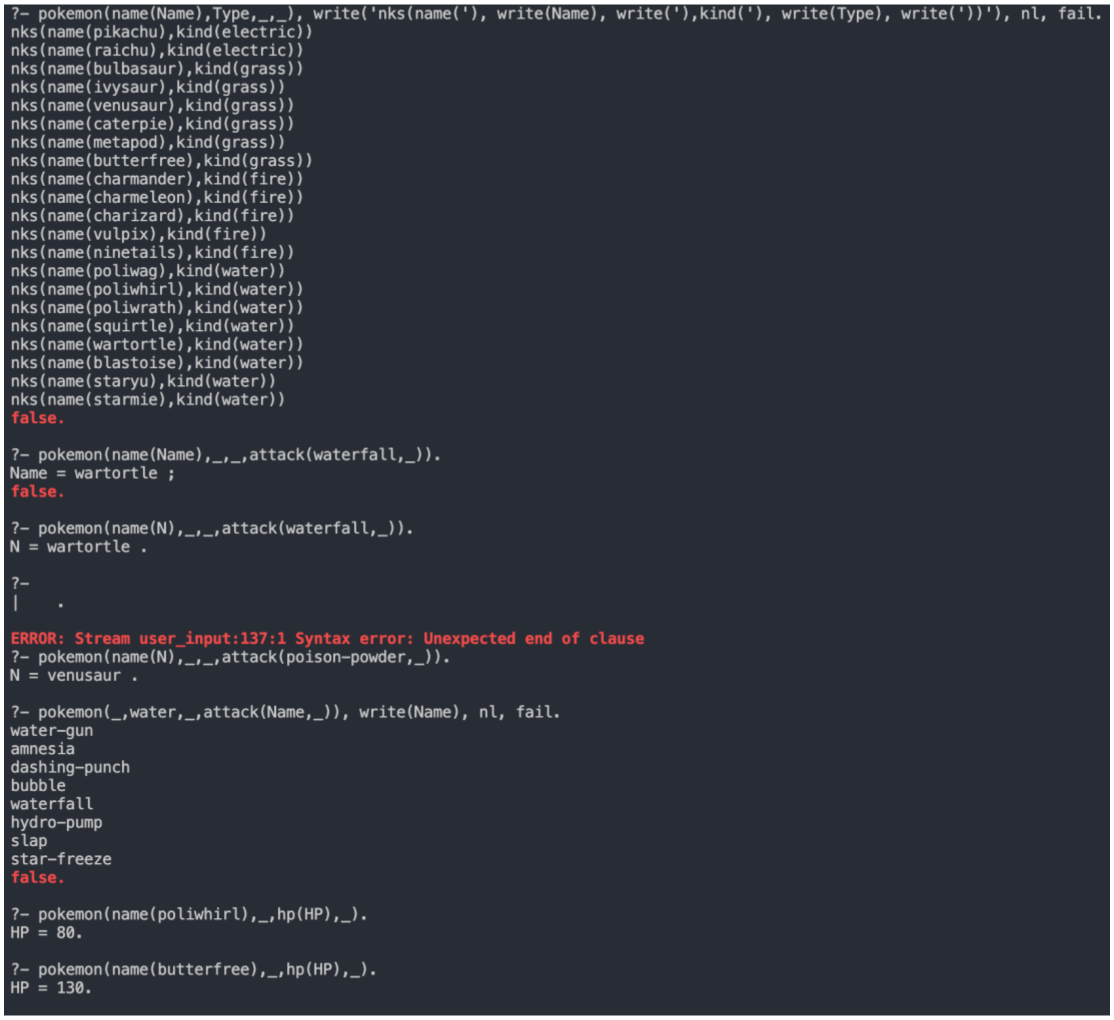
Task 3: Pokemon KB Interaction and Programming



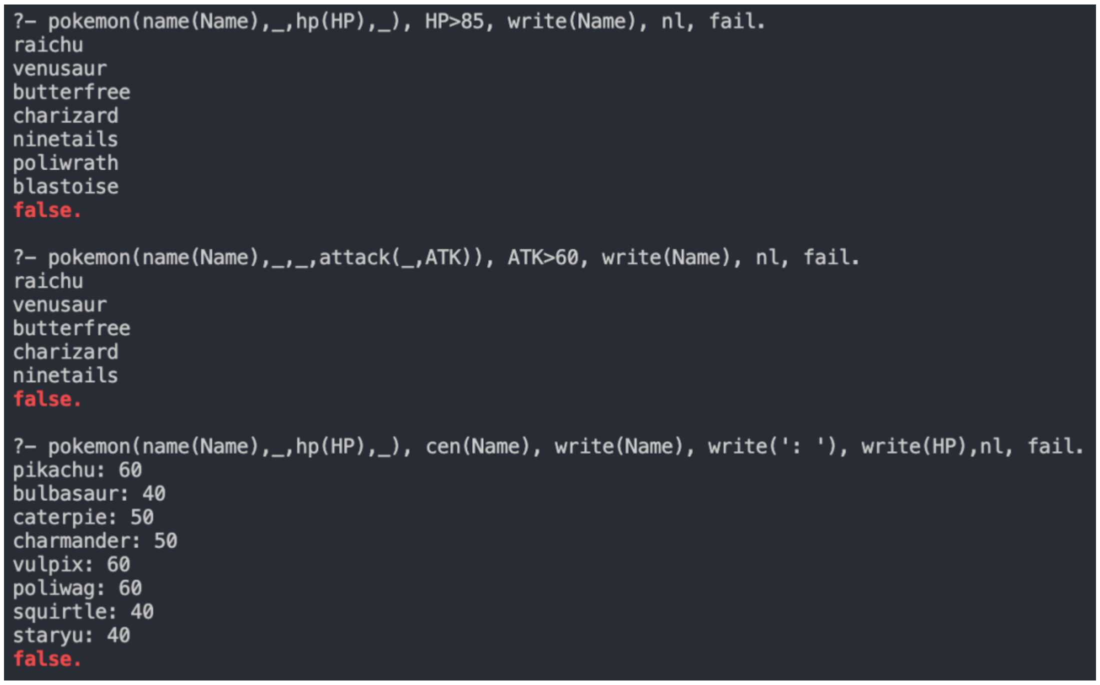
```
Aaradhyas-MacBook-Air:prolog aaradhyaacharya$ swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

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

?- consult('pokemon.pro').
true.

?- cen(pikachu).
true.

?- cen(raichu).
false.

?- pokemon(name(Name),_,_,_), cen(Name).
Name = pikachu ;
Name = bulbasaur ;
Name = caterpie ;
Name = charmander ;
Name = vulpix ;
Name = poliwag ;
Name = squirtle ;
Name = staryu ;
false.

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



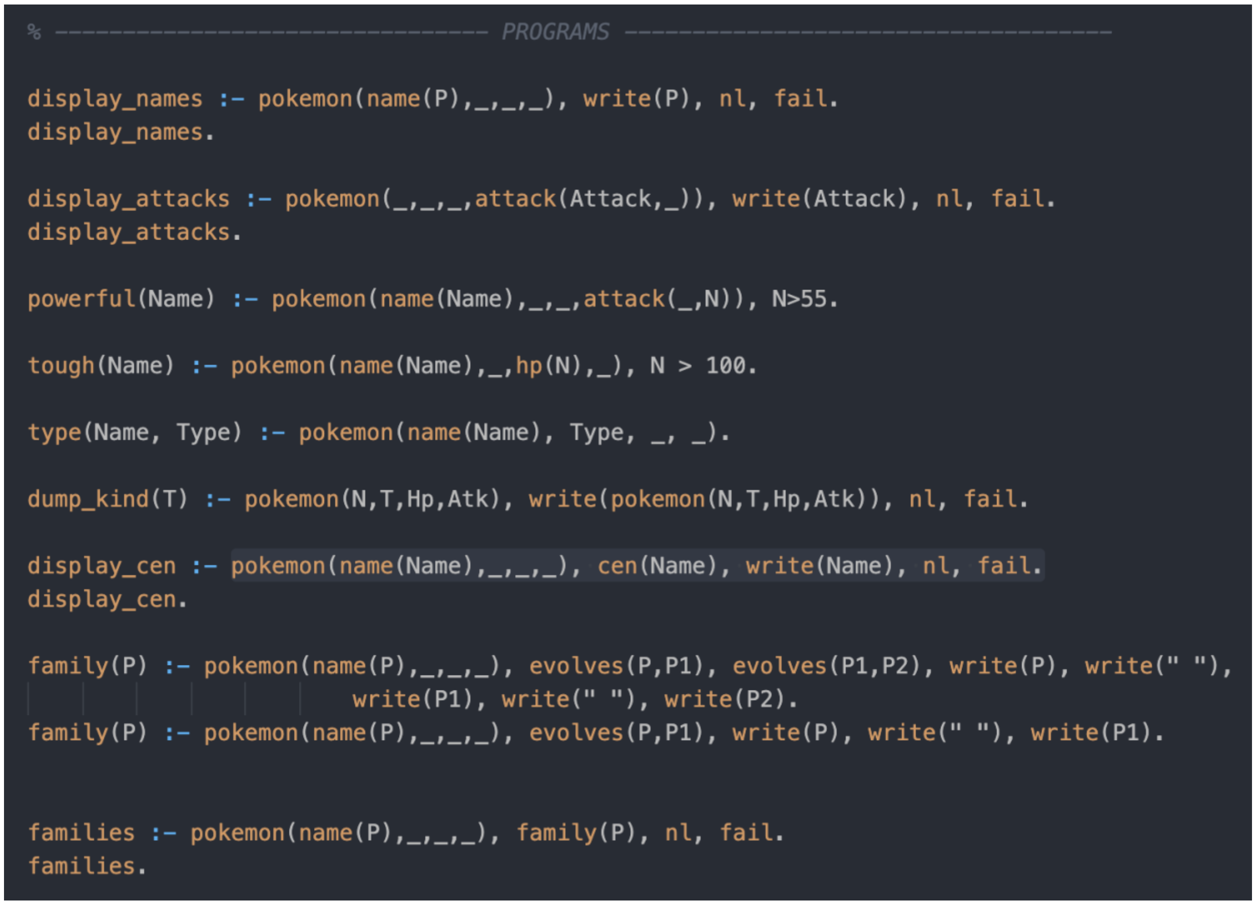
```
?- pokemon(name(Name),_,_,_), cen(Name), write(Name), nl, fail.
pikachu
bulbasaur
caterpie
charmander
vulpix
poliwag
squirtle
staryu
false.

?- 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(Y), write(' -> '), write(Z), nl, fail.
bulbasaur -> ivysaur -> venusaur
caterpie -> metapod -> butterfree
charmander -> charmeleon -> charizard
poliwag -> poliwhirl -> poliwrath
squirtle -> wartortle -> blastoise
false.

?- pokemon(name(Name),fire,_,_), write(Name), nl, fail.
charmander
charmeleon
charizard
vulpix
ninetails
false.
```



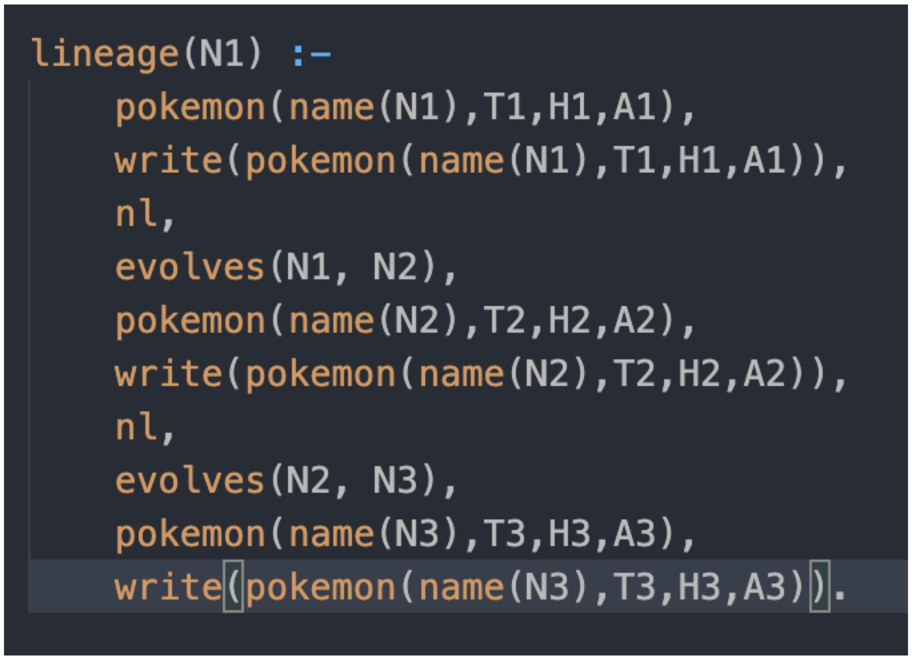
```
?- pokemon(name(Name),Type,_,_), write('nks(name('), write(Name), write('),kind('), write(Type), write('))'), nl, fail.
nks(name(pikachu),kind(electric))
nks(name(raichu),kind(electric))
nks(name(bulbasaur),kind(grass))
nks(name(ivysaur),kind(grass))
nks(name(venusaur),kind(grass))
nks(name(caterpie),kind(grass))
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(Name),_,_,attack(waterfall,_)).
Name = wartortle ;
false.

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

?-
|    .

ERROR: Stream user_input:137:1 Syntax error: Unexpected end of clause
?- pokemon(name(N),_,_,attack(poison-powder,_)).
N = venusaur .

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

?- pokemon(name(poliwhirl),_,hp(HP),_).
HP = 80.

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



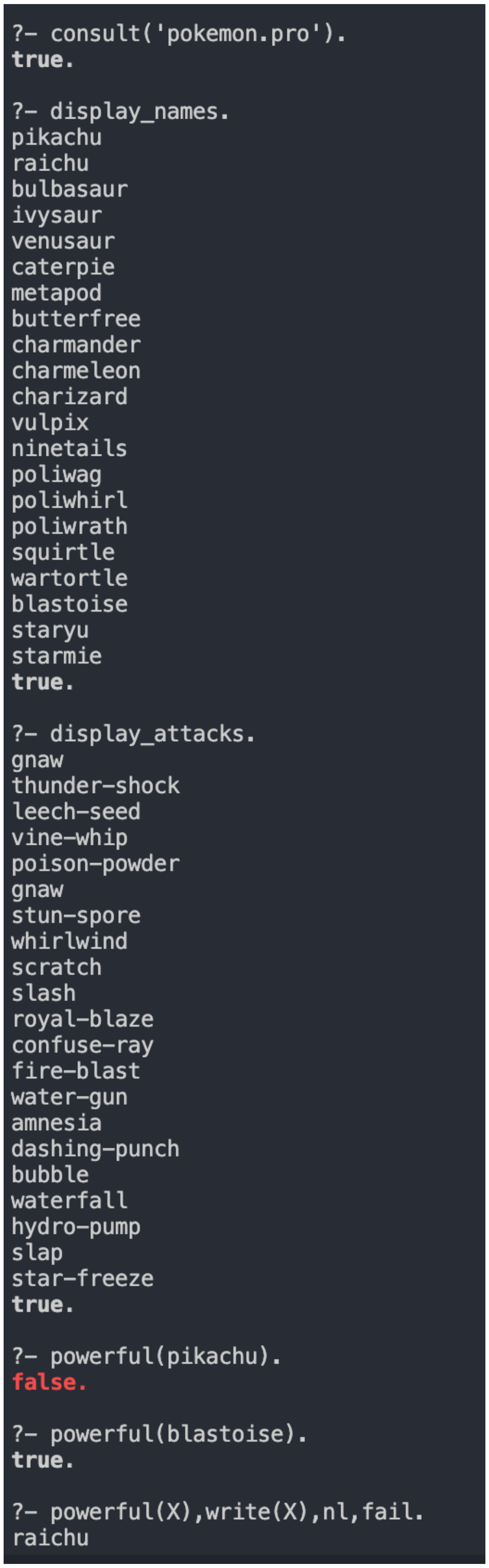
```
?- pokemon(name(Name),_,hp(HP),_), HP>85, write(Name), nl, fail.
raichu
venusaur
butterfree
charizard
ninetails
poliwrath
blastoise
false.

?- pokemon(name(Name),_,_,attack(_,ATK)), ATK>60, write(Name), nl, fail.
raichu
venusaur
butterfree
charizard
ninetails
false.

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



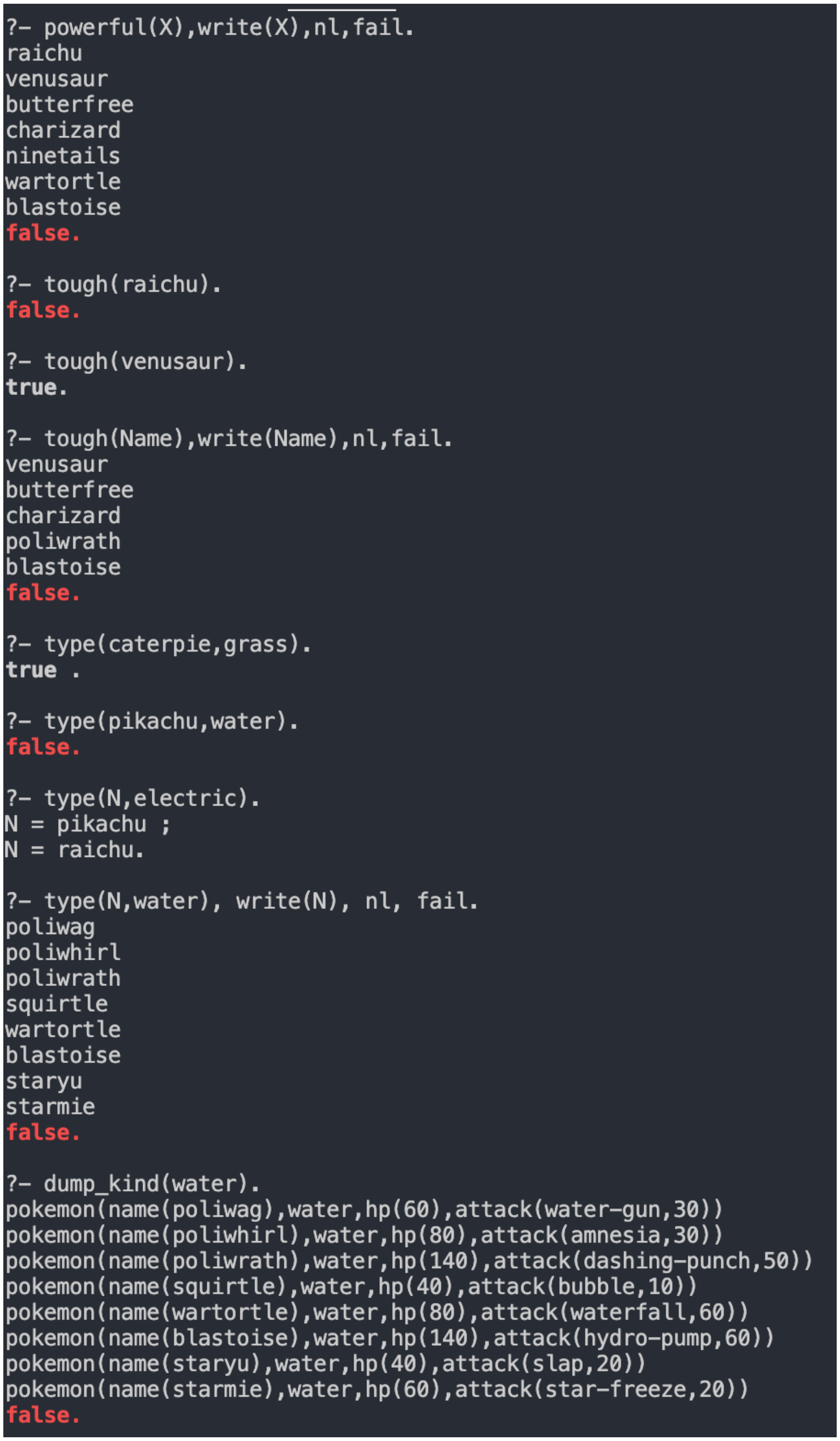
```
% -------------------------------- PROGRAMS ------------------------------------

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

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

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

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

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

dump_kind(T) :- pokemon(N,T,Hp,Atk), write(pokemon(N,T,Hp,Atk)), nl, fail.

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

family(P) :- pokemon(name(P),_,_,_), evolves(P,P1), evolves(P1,P2), write(P), write(" "),
                        write(P1), write(" "), write(P2).
family(P) :- pokemon(name(P),_,_,_), evolves(P,P1), write(P), write(" "), write(P1).


families :- pokemon(name(P),_,_,_), family(P), nl, fail.
families.
```



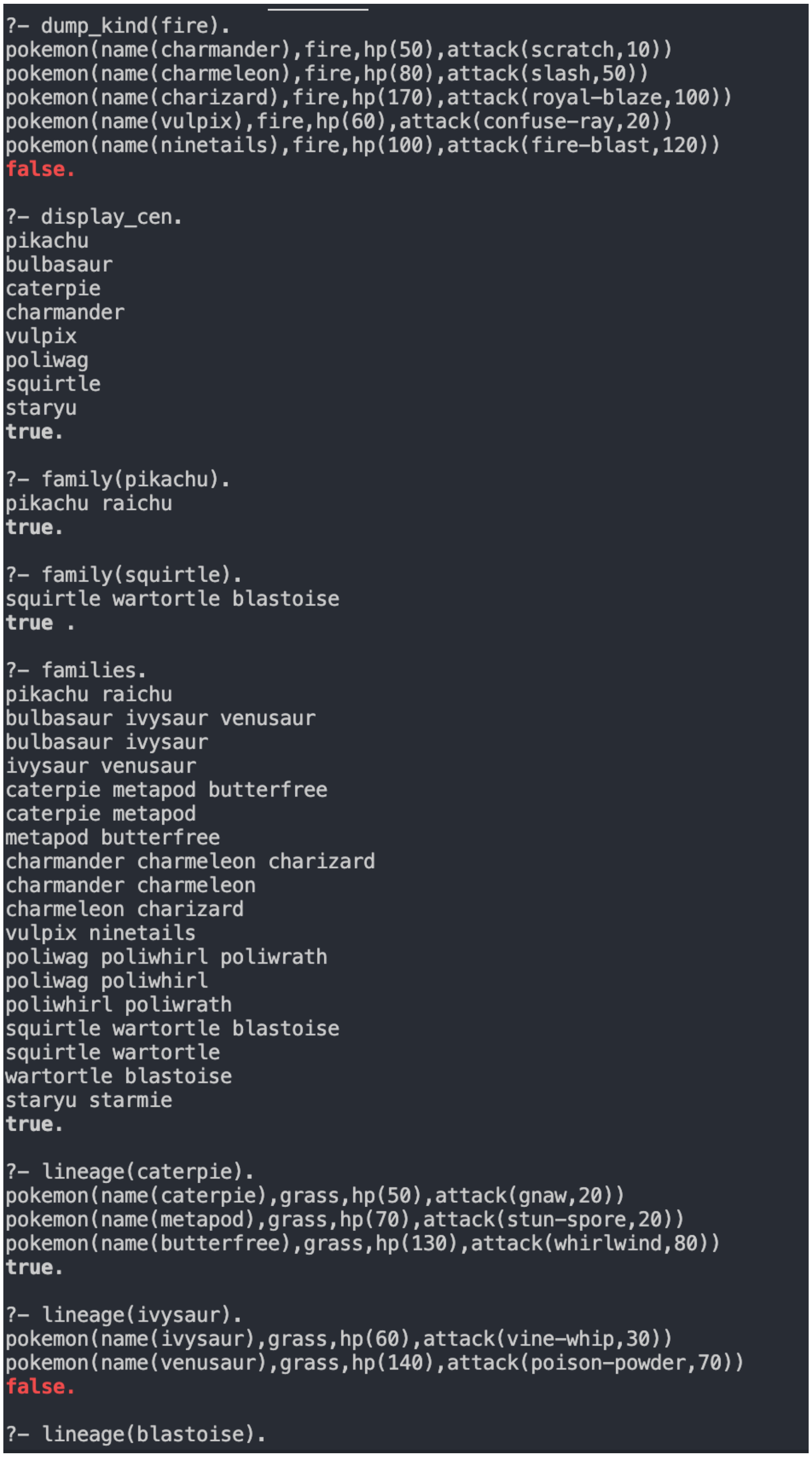
```
lineage(N1) :-
    pokemon(name(N1),T1,H1,A1),
    write(pokemon(name(N1),T1,H1,A1)),
    nl,
    evolves(N1, N2),
    pokemon(name(N2),T2,H2,A2),
    write(pokemon(name(N2),T2,H2,A2)),
    nl,
    evolves(N2, N3),
    pokemon(name(N3),T3,H3,A3),
    write(pokemon(name(N3),T3,H3,A3)).
```



```
?- 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.

?- powerful(pikachu).
false.

?- powerful(blastoise).
true.

?- powerful(X),write(X),nl,fail.
raichu
```

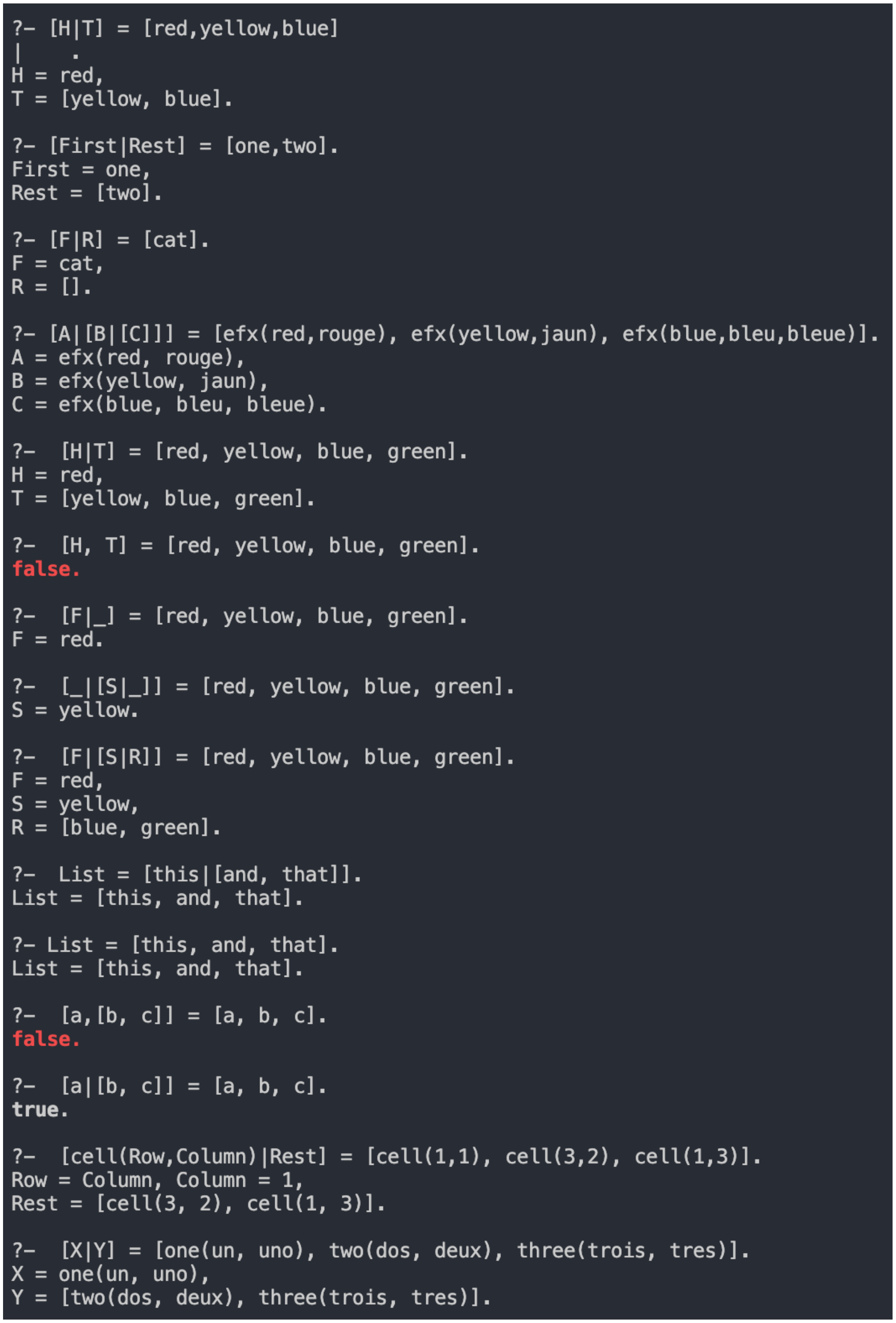


```
?- powerful(X),write(X),nl,fail.
raichu
venusaur
butterfree
charizard
ninetails
wartortle
blastoise
false.

?- 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.
```

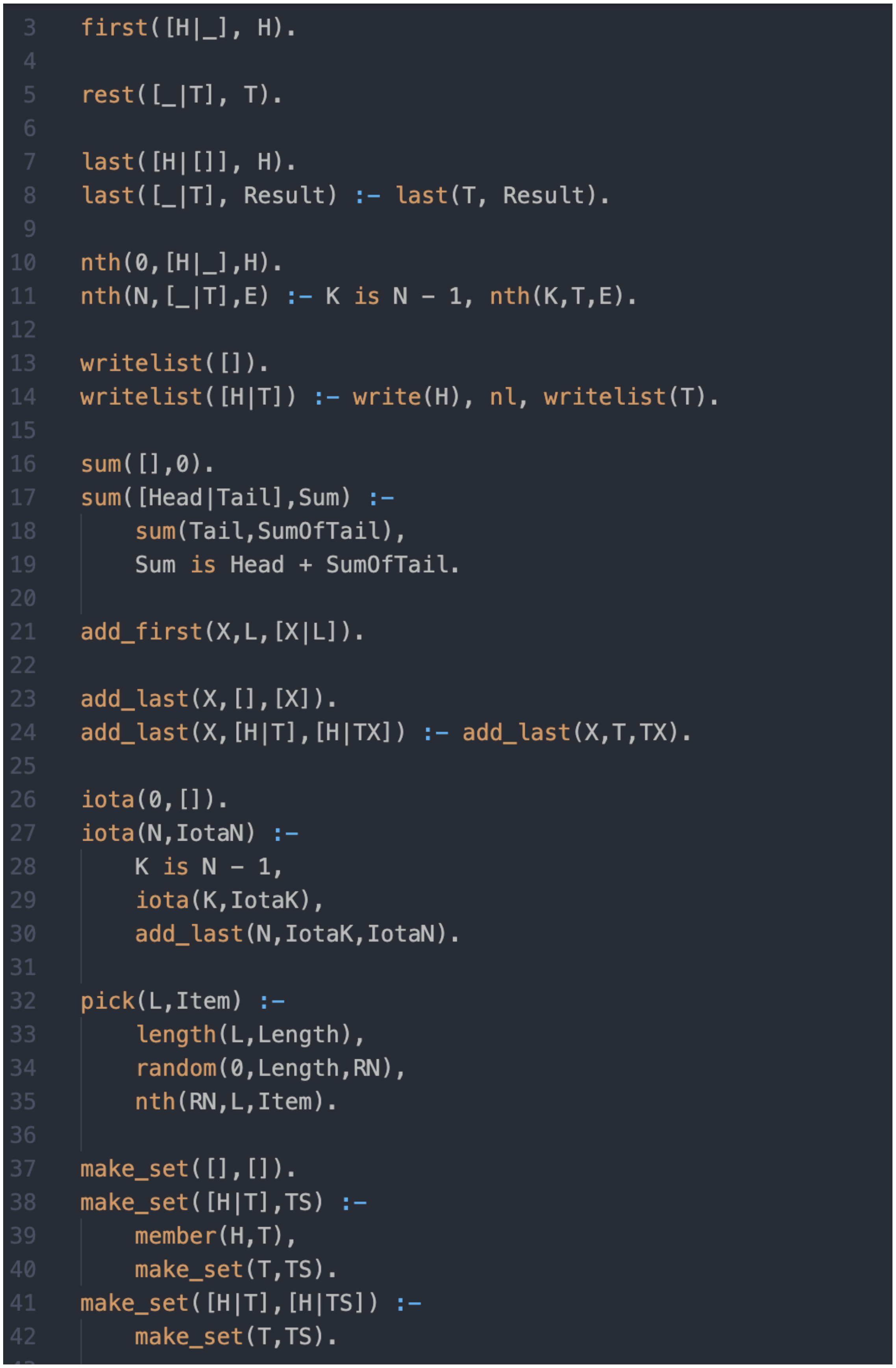


```
?- dump_kind(fire).
pokemon(name(charmander),fire,hp(50),attack(scratch,10))
pokemon(name(charmeleon),fire,hp(80),attack(slash,50))
pokemon(name(charizard),fire,hp(170),attack(royal-blaze,100))
pokemon(name(vulpix),fire,hp(60),attack(confuse-ray,20))
pokemon(name(ninetails),fire,hp(100),attack(fire-blast,120))
false.

?- display_cen.
pikachu
bulbasaur
caterpie
charmander
vulpix
poliwag
squirtle
staryu
true.

?- family(pikachu).
pikachu raichu
true.

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

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

?- 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(ivysaur).
pokemon(name(ivysaur),grass,hp(60),attack(vine-whip,30))
pokemon(name(venusaur),grass,hp(140),attack(poison-powder,70))
false.

?- lineage(blastoise).
```

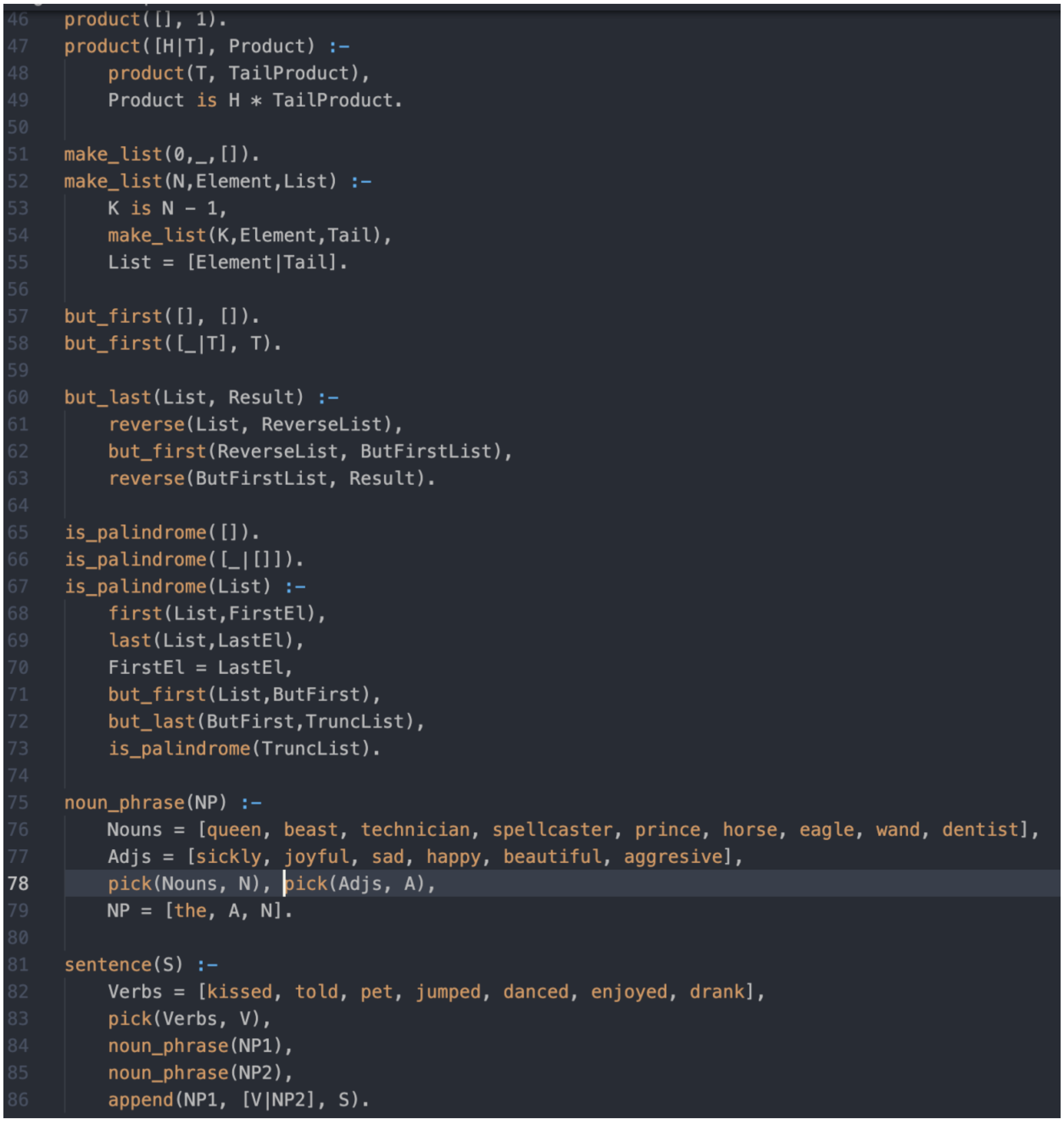
Task 4: List Processing in Prolog



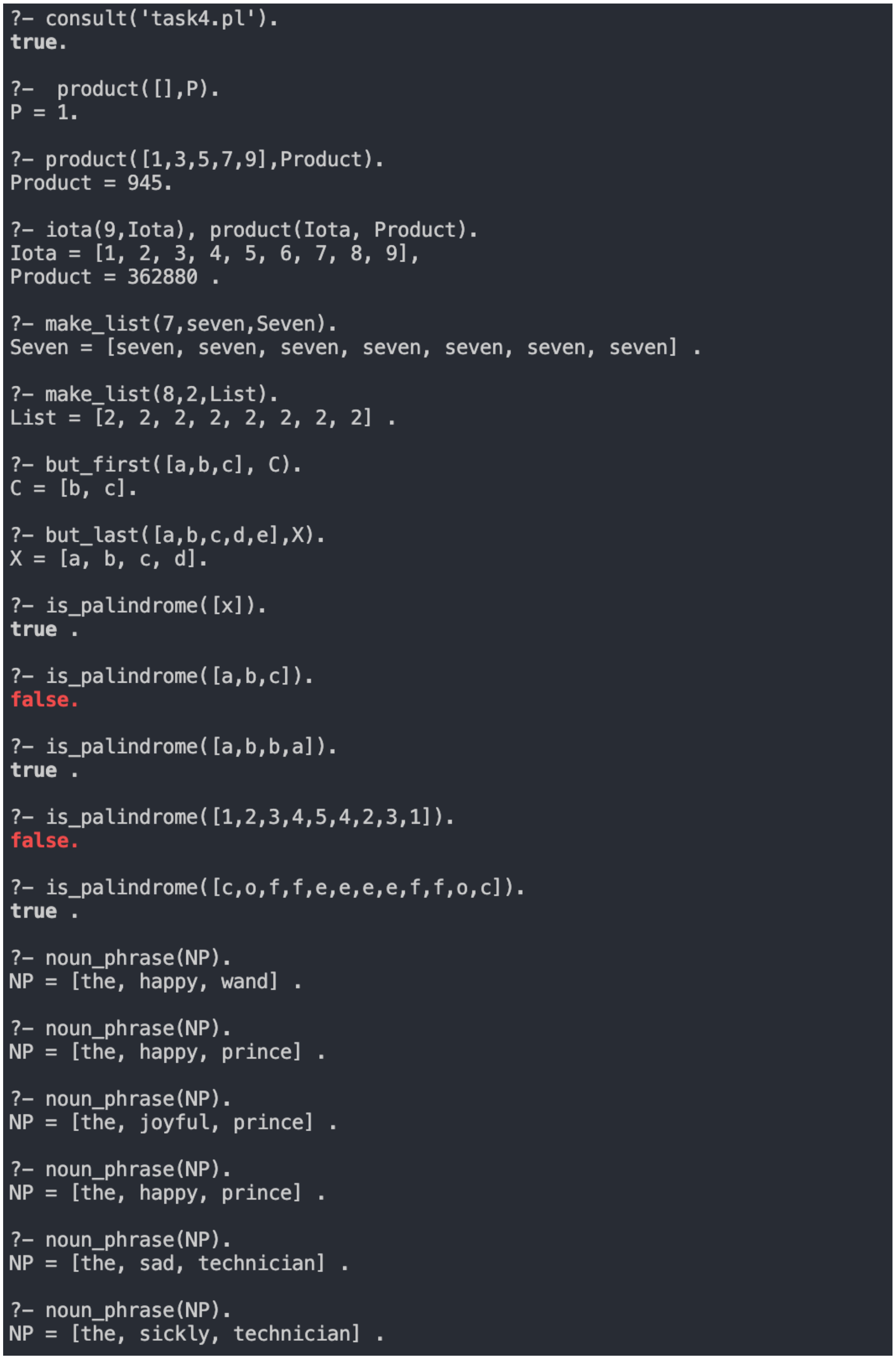
```
?- [H|T] = [red,yellow,blue]
|    .
H = red,
T = [yellow, blue].

?- [First|Rest] = [one,two].
First = one,
Rest = [two].

?- [F|R] = [cat].
F = cat,
R = [].

?- [A|[B|[C]]] = [efx(red,rouge), efx(yellow,jaun), efx(blue,bleu,bleue)].
A = efx(red, rouge),
B = efx(yellow, jaun),
C = efx(blue, bleu, bleue).

?-  [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)].
```



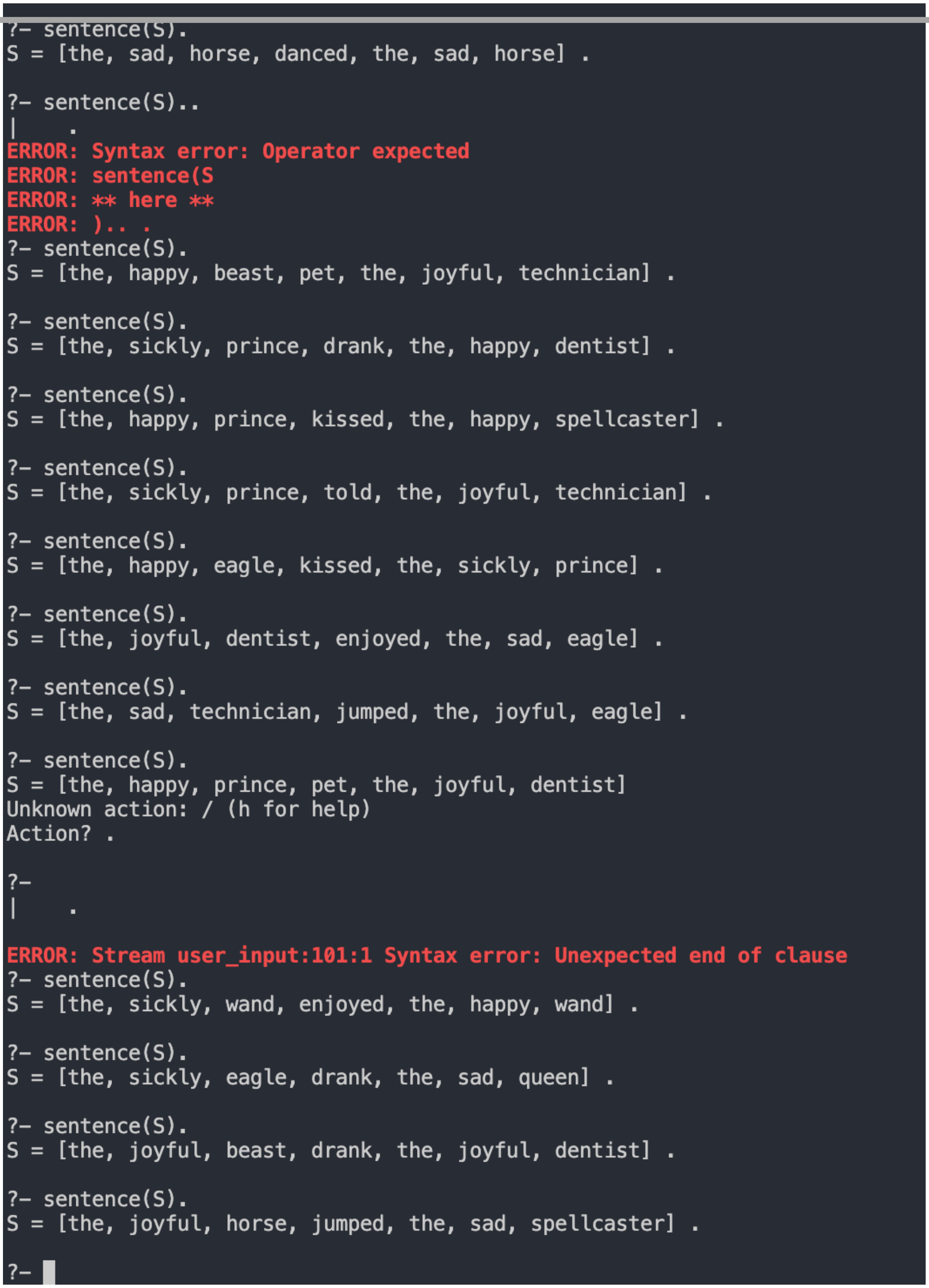
```prolog
first([H|_], H).

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([H|T], Product) :-
    product(T, TailProduct),
    Product is H * TailProduct.

make_list(0,_,[]).
make_list(N,Element,List) :-
    K is N - 1,
    make_list(K,Element,Tail),
    List = [Element|Tail].

but_first([], []).
but_first([_|T], T).

but_last(List, Result) :-
    reverse(List, ReverseList),
    but_first(ReverseList, ButFirstList),
    reverse(ButFirstList, Result).

is_palindrome([]).
is_palindrome([_|[]]).
is_palindrome(List) :-
    first(List,FirstEl),
    last(List,LastEl),
    FirstEl = LastEl,
    but_first(List,ButFirst),
    but_last(ButFirst,TruncList),
    is_palindrome(TruncList).

noun_phrase(NP) :-
    Nouns = [queen, beast, technician, spellcaster, prince, horse, eagle, wand, dentist],
    Adjs = [sickly, joyful, sad, happy, beautiful, aggresive],
    pick(Nouns, N), pick(Adjs, A),
    NP = [the, A, N].

sentence(S) :-
    Verbs = [kissed, told, pet, jumped, danced, enjoyed, drank],
    pick(Verbs, V),
    noun_phrase(NP1),
    noun_phrase(NP2),
    append(NP1, [V|NP2], S).
```



```
?- consult('task4.pl').
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], C).
C = [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, happy, wand] .

?- noun_phrase(NP).
NP = [the, happy, prince] .

?- noun_phrase(NP).
NP = [the, joyful, prince] .

?- noun_phrase(NP).
NP = [the, happy, prince] .

?- noun_phrase(NP).
NP = [the, sad, technician] .

?- noun_phrase(NP).
NP = [the, sickly, technician] .
```



```
?- sentence(S).
S = [the, sad, horse, danced, the, sad, horse] .

?- sentence(S)..
|    .
ERROR: Syntax error: Operator expected
ERROR: sentence(S
ERROR: ** here **
ERROR: ).. .
?- sentence(S).
S = [the, happy, beast, pet, the, joyful, technician] .

?- sentence(S).
S = [the, sickly, prince, drank, the, happy, dentist] .

?- sentence(S).
S = [the, happy, prince, kissed, the, happy, spellcaster] .

?- sentence(S).
S = [the, sickly, prince, told, the, joyful, technician] .

?- sentence(S).
S = [the, happy, eagle, kissed, the, sickly, prince] .

?- sentence(S).
S = [the, joyful, dentist, enjoyed, the, sad, eagle] .

?- sentence(S).
S = [the, sad, technician, jumped, the, joyful, eagle] .

?- sentence(S).
S = [the, happy, prince, pet, the, joyful, dentist]
Unknown action: / (h for help)
Action? .

?-
|    .

ERROR: Stream user_input:101:1 Syntax error: Unexpected end of clause
?- sentence(S).
S = [the, sickly, wand, enjoyed, the, happy, wand] .

?- sentence(S).
S = [the, sickly, eagle, drank, the, sad, queen] .

?- sentence(S).
S = [the, joyful, beast, drank, the, joyful, dentist] .

?- sentence(S).
S = [the, joyful, horse, jumped, the, sad, spellcaster] .

?-
```