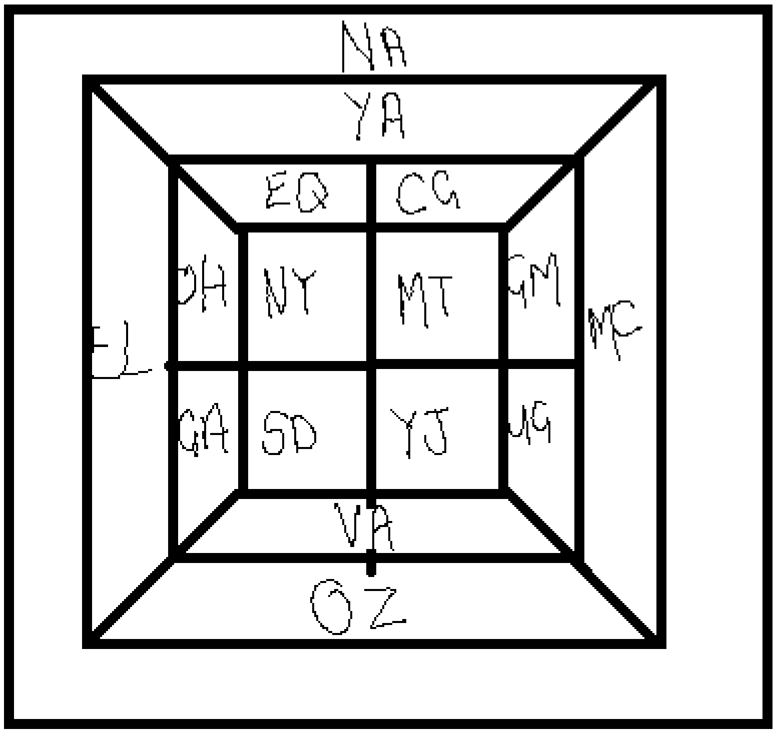
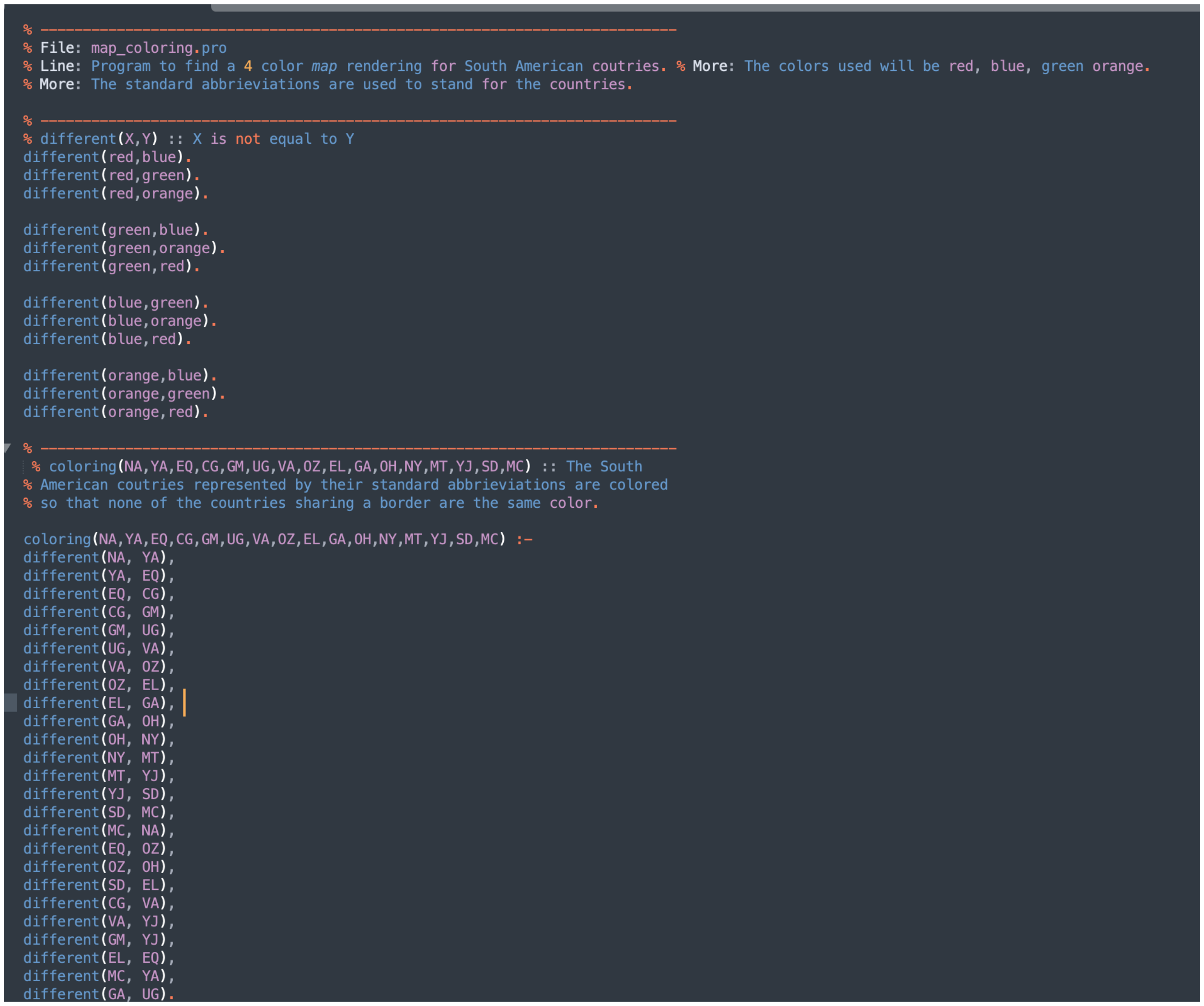
Prolog Programming Assignment #1: Various Computations

Learning Abstract:

This is Programming practices that emphasize information portrayal, search, and rundown handling in Prolog.It also features a pattern matching system much of the time utilized in Prolog to dismantle/build records, known as head/tail documentation which is presented in the list processing demos.

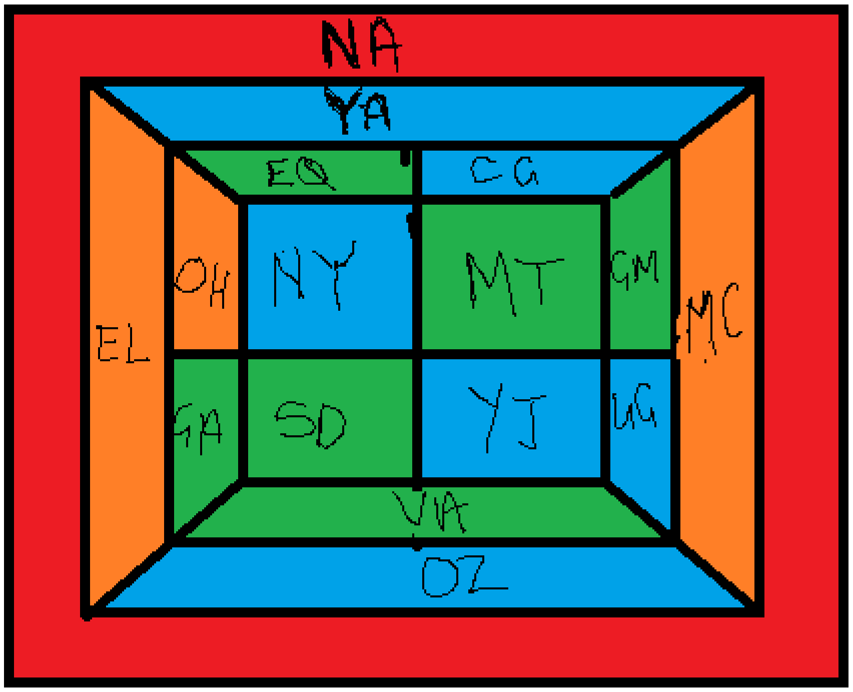
Task 1: Map Coloring



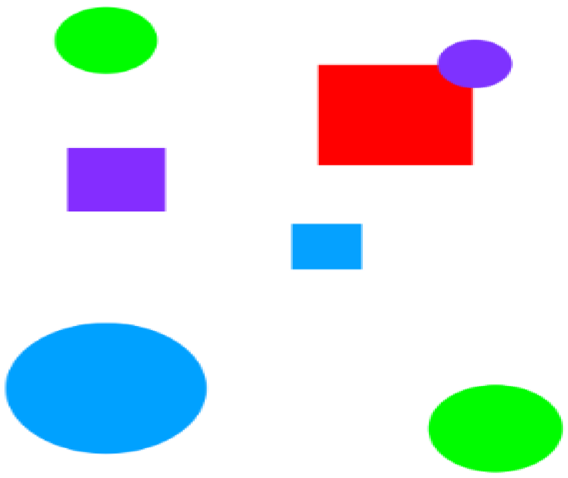


```
% ----------------------------------------------------------------------------
% File: map_coloring.pro
% Line: Program to find a 4 color map rendering for South American couties. % More: The colors used will be red, blue, green orange.
% More: The standard abbrieviations are used to stand for the countries.

% ----------------------------------------------------------------------------
% different(X,Y) :: X is not equal to Y
different(red,blue).
different(red,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(NA,YA,EQ,CG,GM,UG,VA,OZ,EL,GA,OH,NY,MT,YJ,SD,MC) :: The South
% American couties represented by their standard abbrieviations are colored
% so that none of the countries sharing a border are the same color.

coloring(NA,YA,EQ,CG,GM,UG,VA,OZ,EL,GA,OH,NY,MT,YJ,SD,MC) :-
different(NA, YA),
different(YA, EQ),
different(EQ, CG),
different(CG, GM),
different(GM, UG),
different(UG, VA),
different(VA, OZ),
different(OZ, EL),
different(EL, GA),
different(GA, OH),
different(OH, NY),
different(NY, MT),
different(MT, YJ),
different(YJ, SD),
different(SD, MC),
different(MC, NA),
different(EQ, OZ),
different(OZ, OH),
different(SD, EL),
different(CG, VA),
different(VA, YJ),
different(GM, YJ),
different(EL, EQ),
different(MC, YA),
different(GA, UG).
```

```
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).

?- ['/Users/nadrataabdul-salam/Desktop/Prolog/test.pL']
|     .
true.

?- coloring(NA,YA,EQ,CG,GM,UG,VA,OZ,EL,GA,OH,NY,MT,YJ,SD,MC)
|     .
NA = red,
YA = CG, CG = UG, UG = OZ, OZ = NY, NY = YJ, YJ = blue,
EQ = GM, GM = VA, VA = GA, GA = MT, MT = SD, SD = green,
EL = OH, OH = MC, MC = orange
```

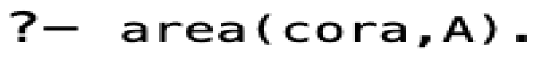


Task 2: The Floating Shapes World

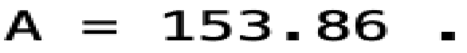


```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.0)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

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

?- ['/Users/nadrataabdul-salam/Desktop/Prolog/test.pl2']
|    .
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.
```

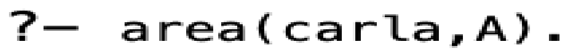
```
?- listing(shapes).
shapes :-
    circles,
    squares.

true.

?- shapes.

carla
cora
connie
claire
sera
sara
sarah
true.

?- blue(Shape).
Shape = sara .

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

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



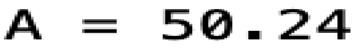
```
?- area(cora,A).
```



```
A = 153.86 .
```

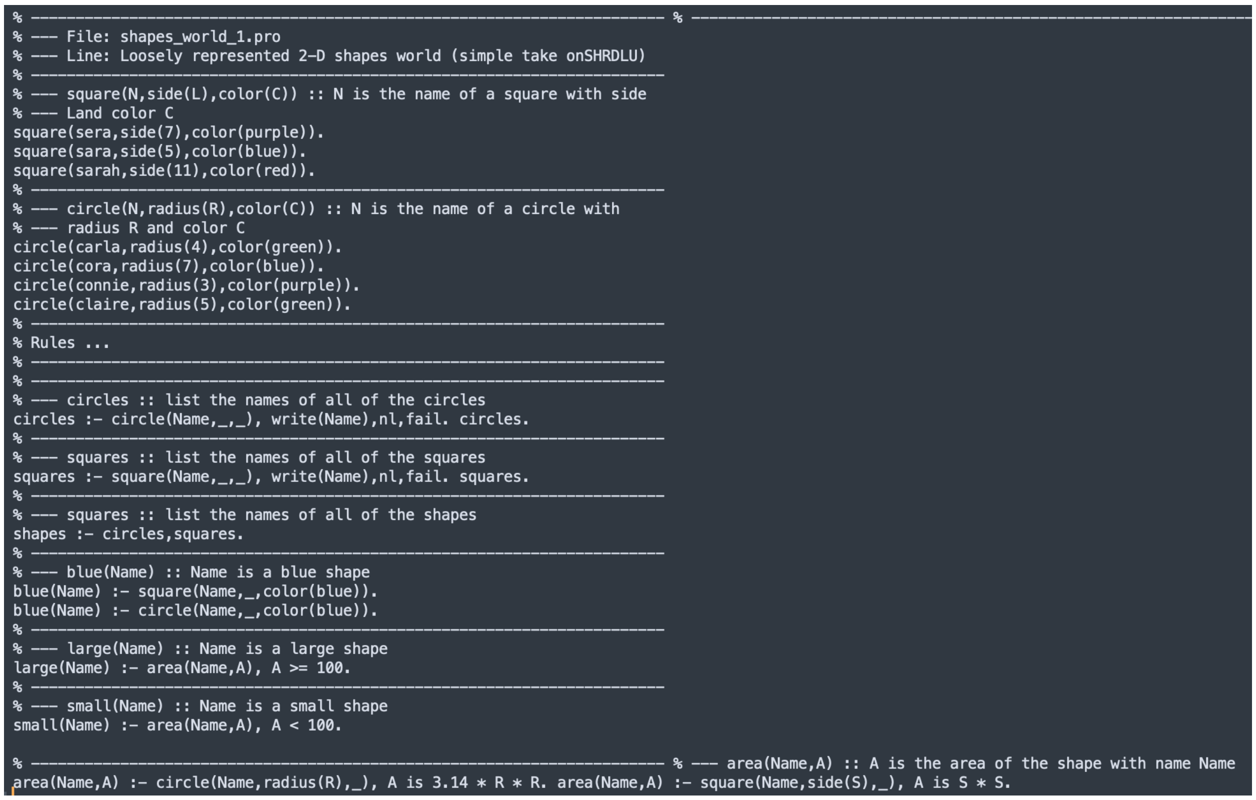


```
?- area(carla,A).
```



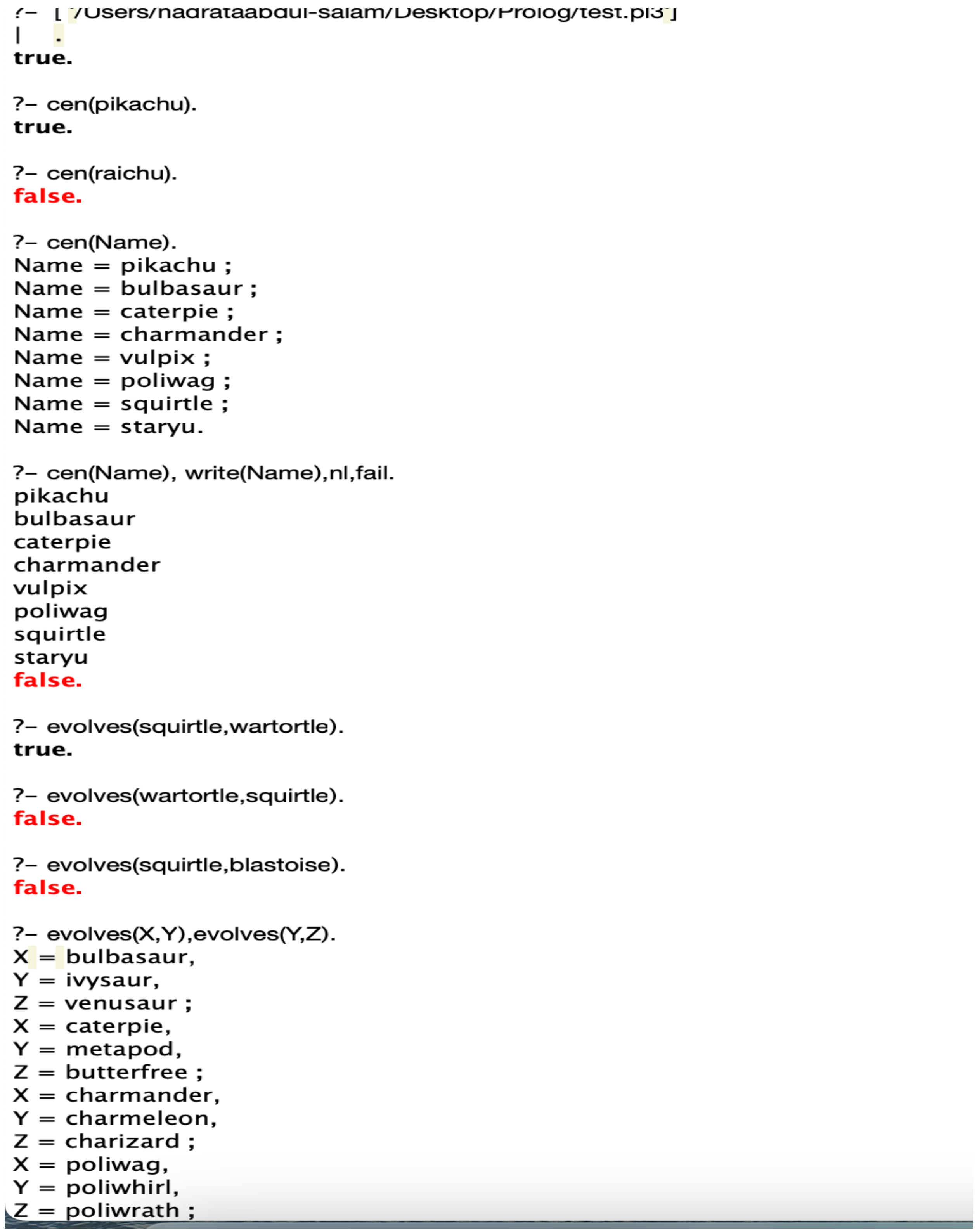
```
A = 50.24
```

Task2 code

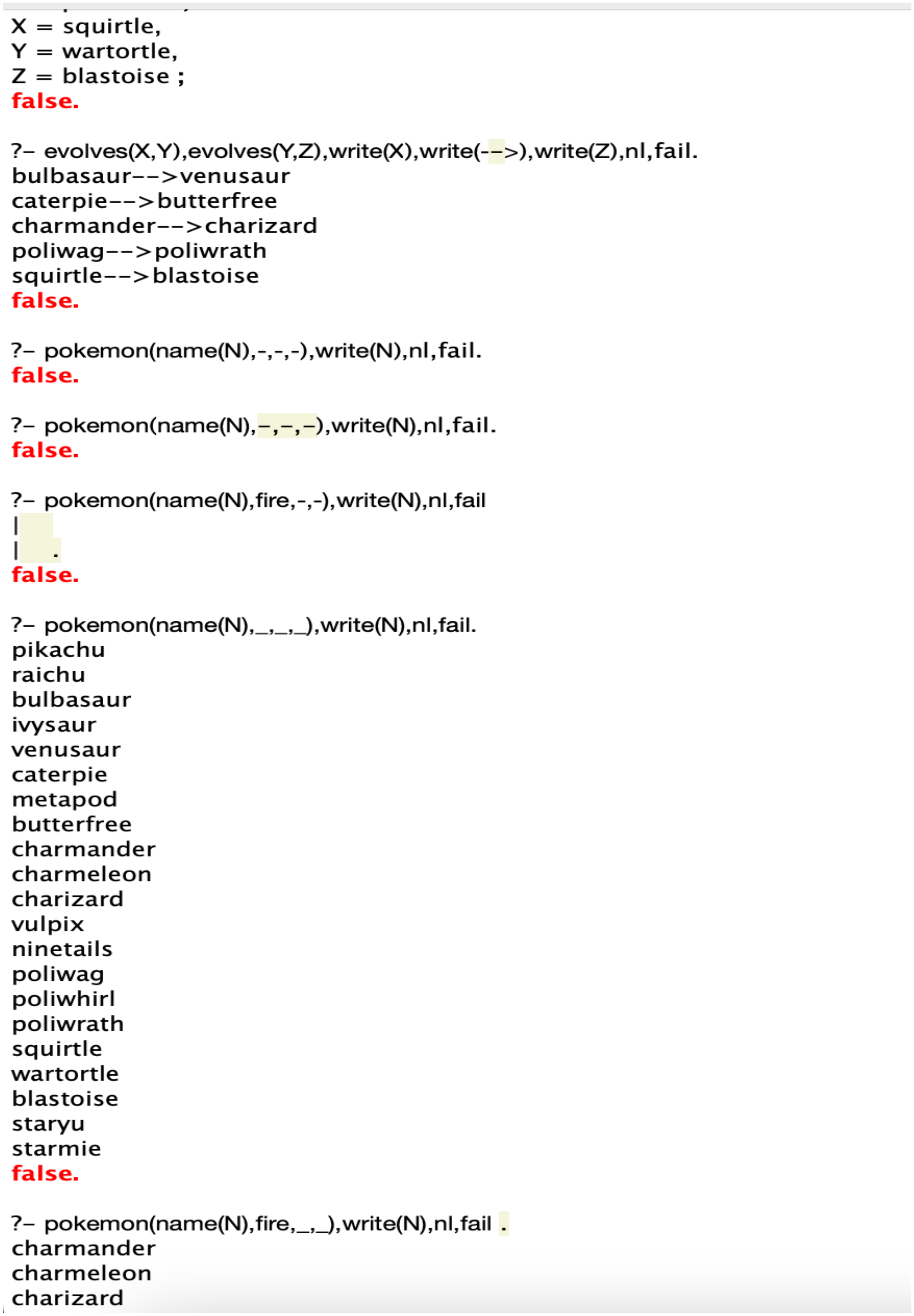


```
% -------------------------------------------------------------------------- % -----------------------------------------------------------------
% --- File: shapes_world_1.pro
% --- Line: Loosely represented 2-D shapes world (simple take onSHRDLU)
% --------------------------------------------------------------------------
% --- square(N,side(L),color(C)) :: N is the name of a square with side
% --- Land 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
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.
```

Task 3: Pokemon KB Interaction and Programming

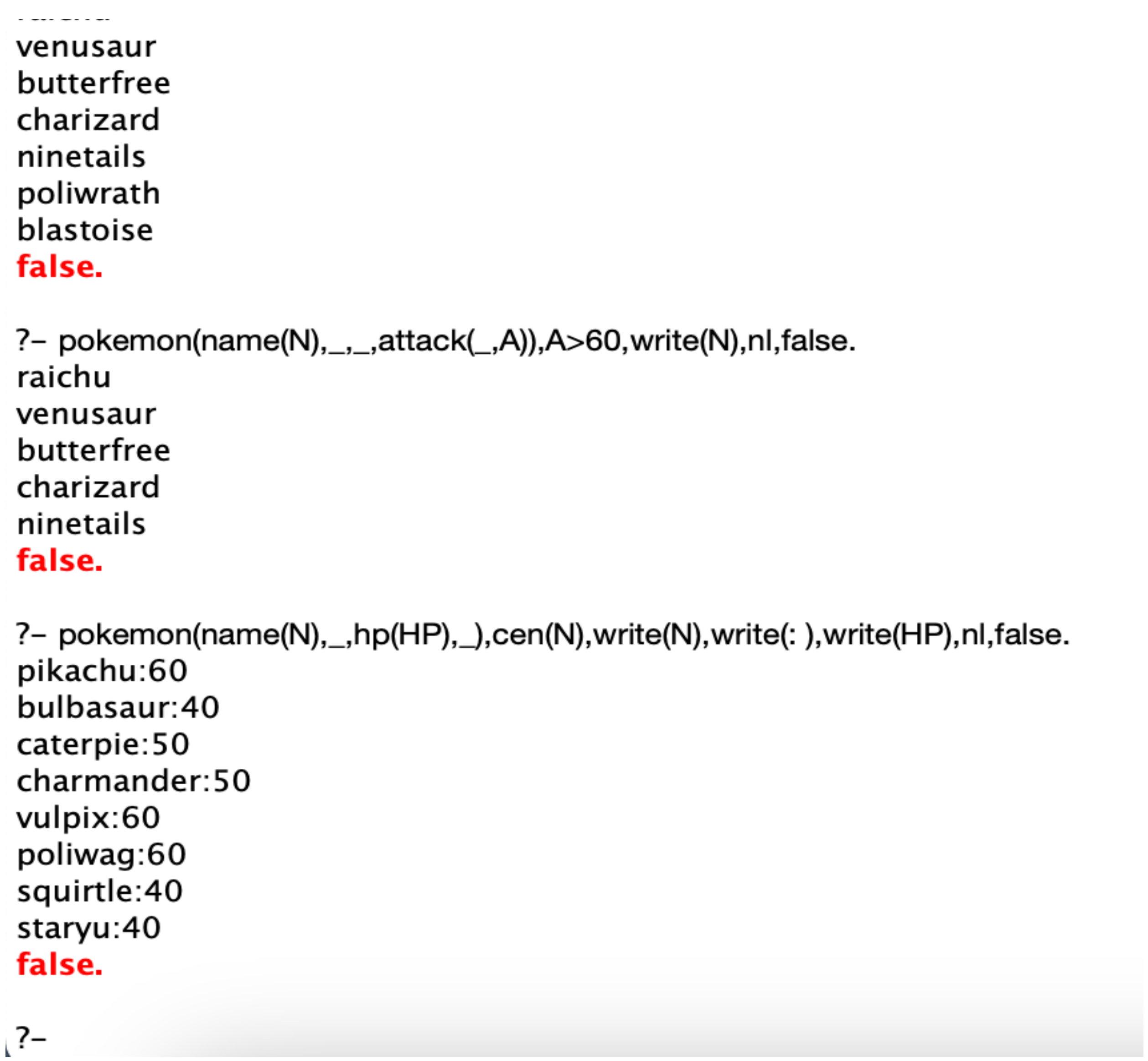


```
?- ['/Users/nadrataabdul-salam/Desktop/Prolog/test.pl3']
|    .
true.

?- 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(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 ;
```



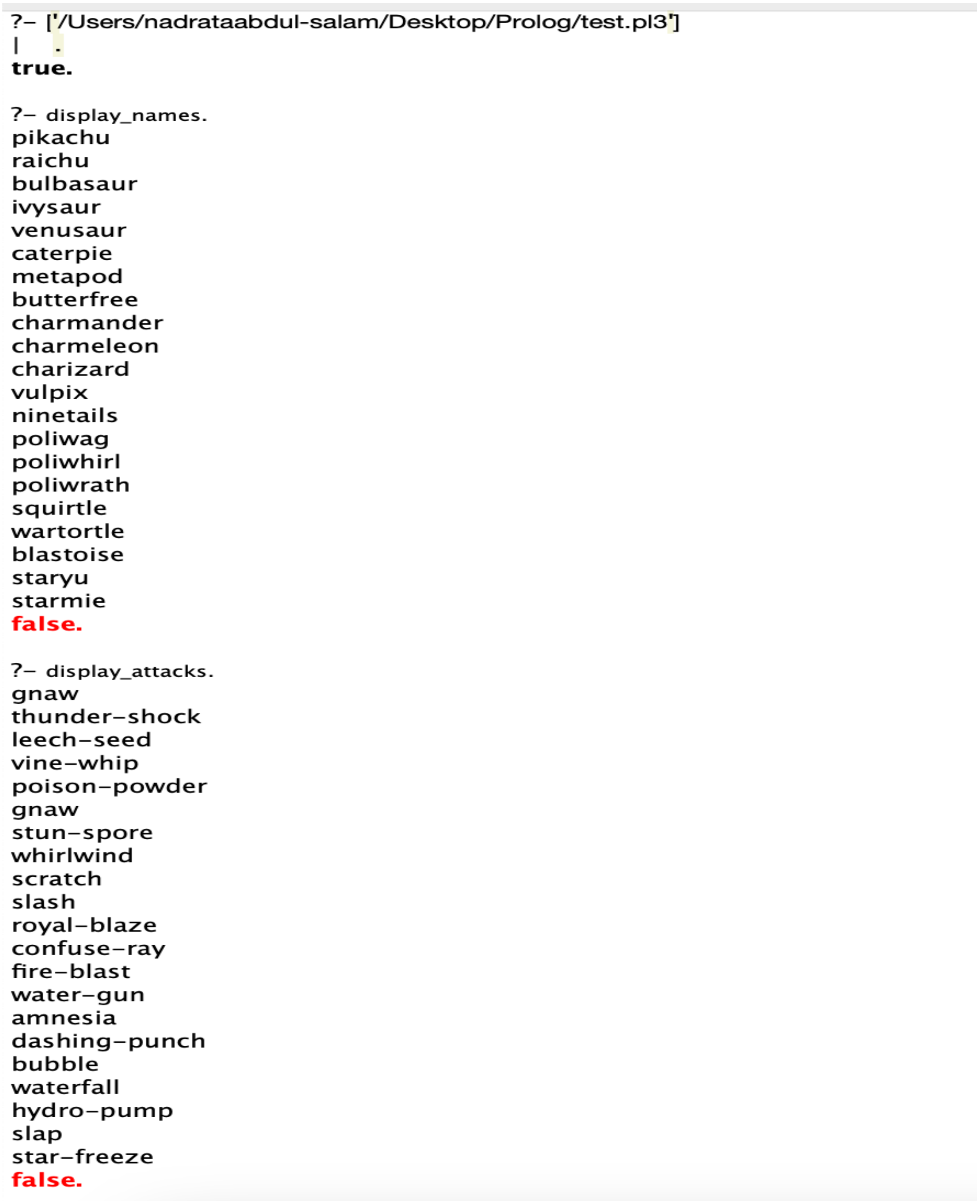
```
X = squirtle,
Y = wartortle,
Z = blastoise ;
false.

?- 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.
false.

?- pokemon(name(N),-,-,-),write(N),nl,fail.
false.

?- pokemon(name(N),fire,-,-),write(N),nl,fail
|
|    .
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,Element,_,_),write(nks(N,kind(Element))),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(Ok,_)),write(Ok),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(N),_,hp(HP),_),HP>85,write(N),nl,false.
raichu
```



```
venusaur
butterfree
charizard
ninetails
poliwrath
blastoise
false.

?- pokemon(name(N),_,_,attack(_,A)),A>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.

?-
```

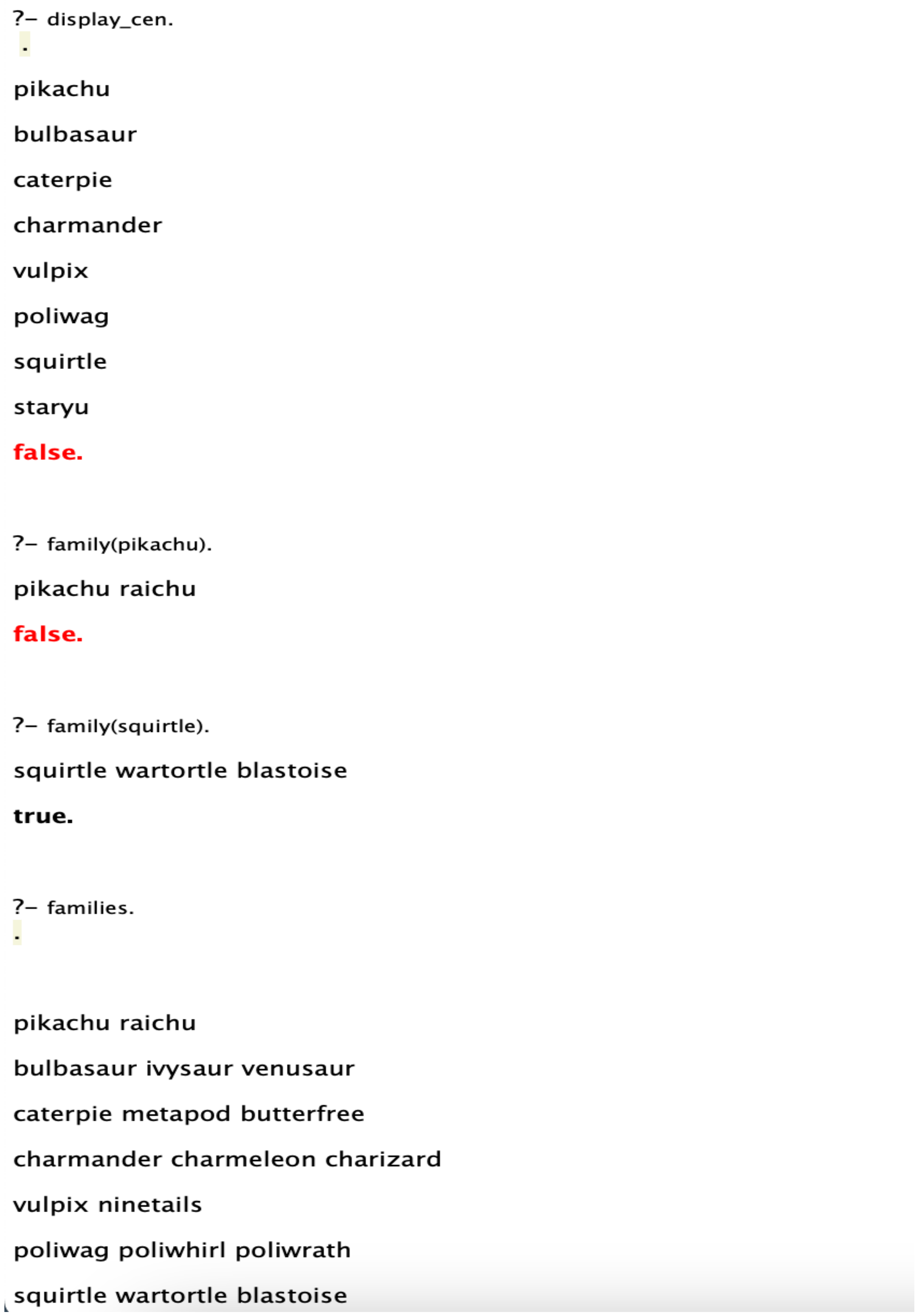
Part 1: Queries



```
?- ['/Users/nadrataabdul-salam/Desktop/Prolog/test.pl3']
|    .
true.

?- 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(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
ninetails
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.
```

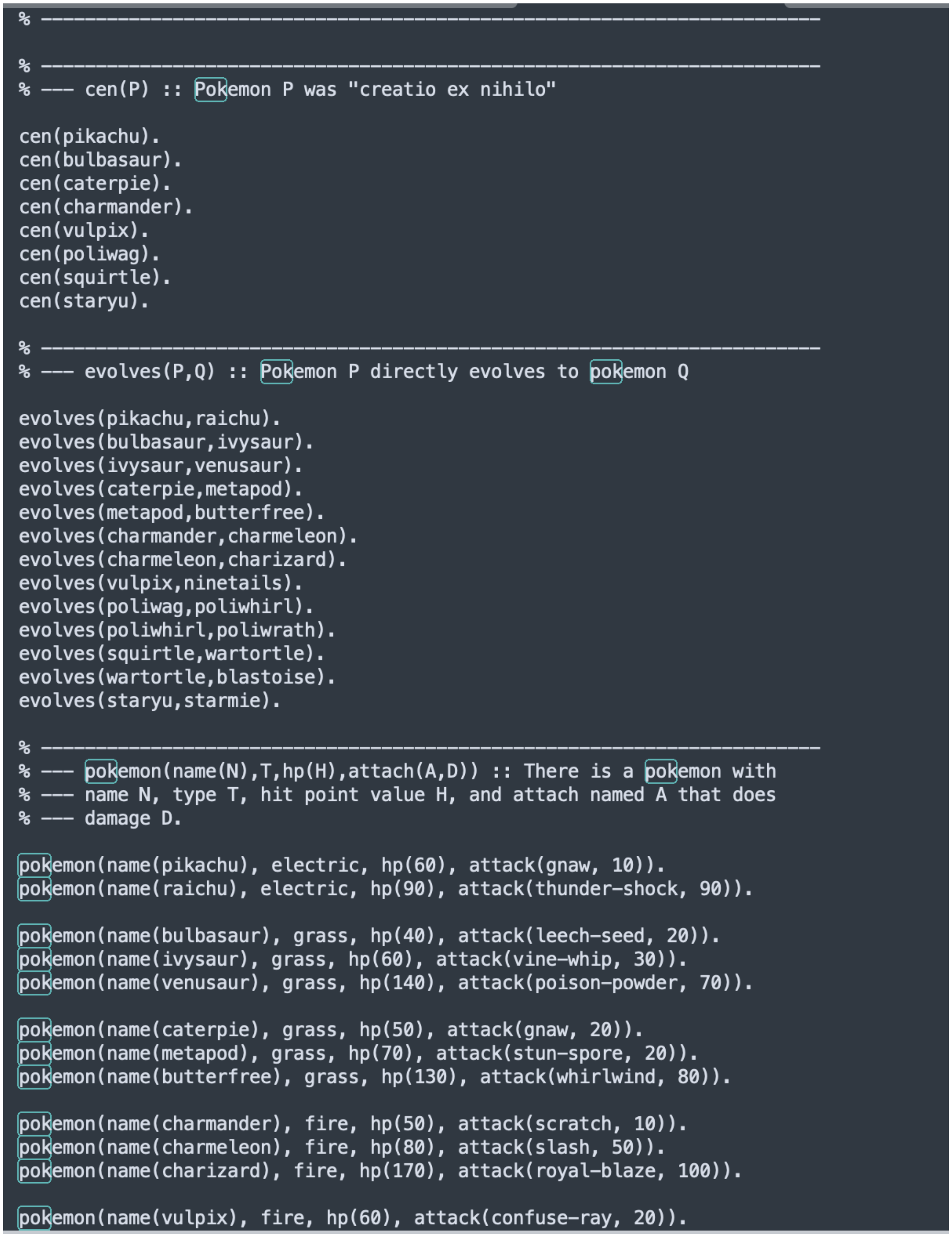
```
?- dump_kind(water).
pokemon(name(poliwag), water, hp(60), attack(water-gun, 30)).
pokemon(name(poliwhirl), water, hp(80), attack(amnesia, 30)).
pokemon(name(poliwrath), water, hp(140), attack(dashing-punch, 50)).
pokemon(name(squirtle), water, hp(40), attack(bubble, 10)).
pokemon(name(wartortle), 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)).

true.

?- 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)).

true.
```



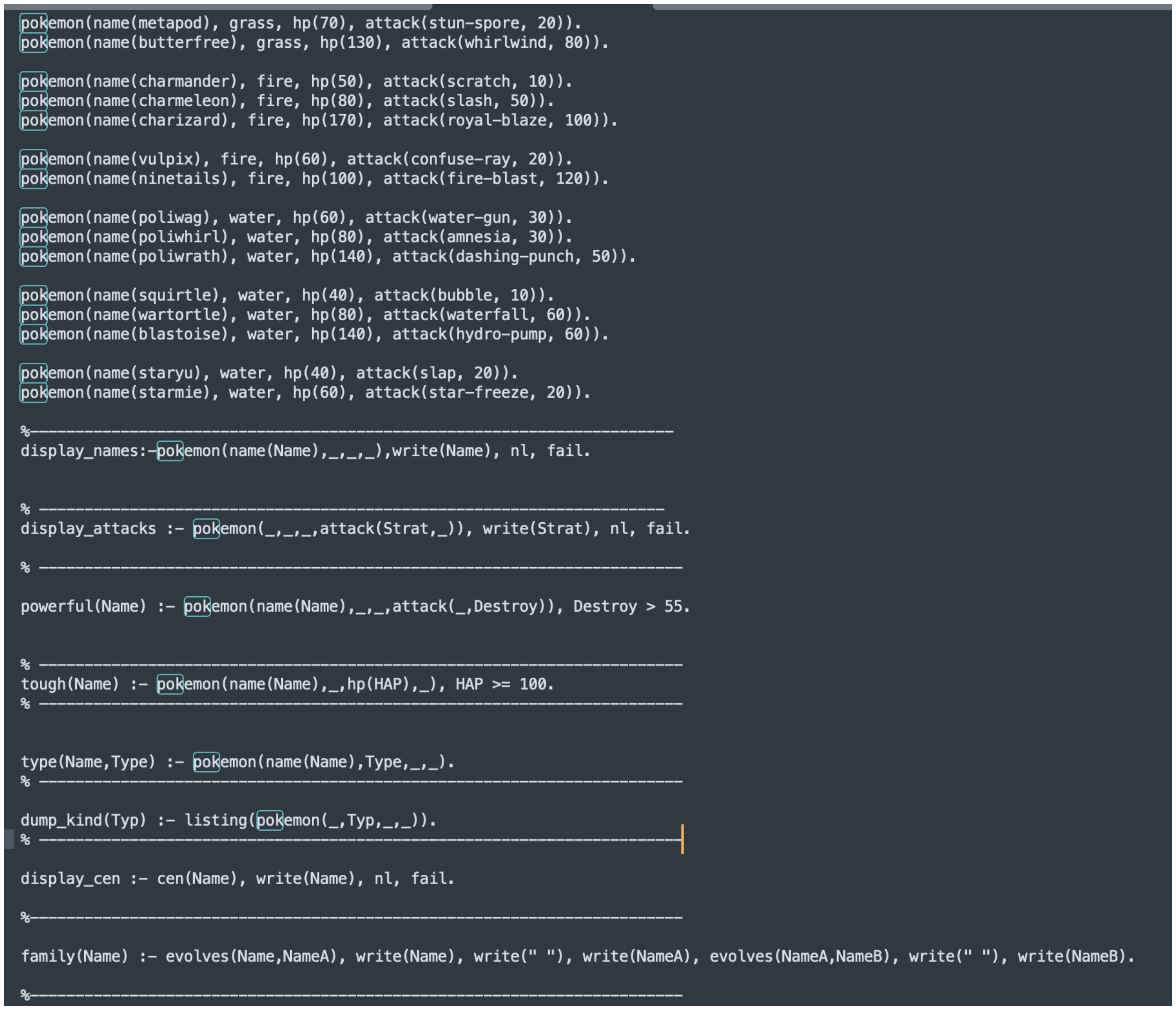
```
?- display_cen.
.
pikachu
bulbasaur
caterpie
charmander
vulpix
poliwag
squirtle
staryu
false.


?- family(pikachu).
pikachu raichu
false.


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


?- families.
.


pikachu raichu
bulbasaur 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.

?-
```

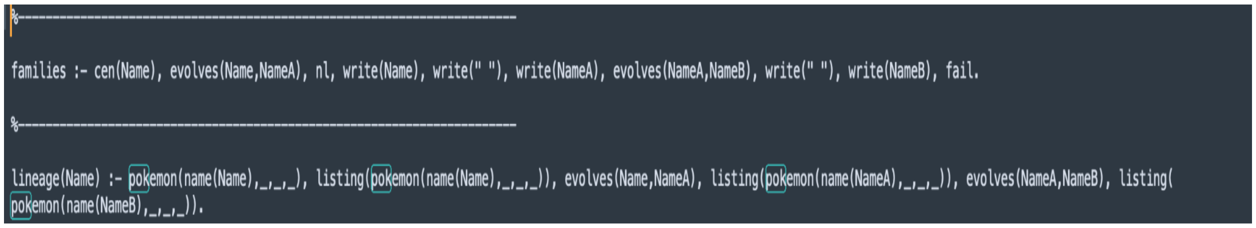
Part 2: Programs



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

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

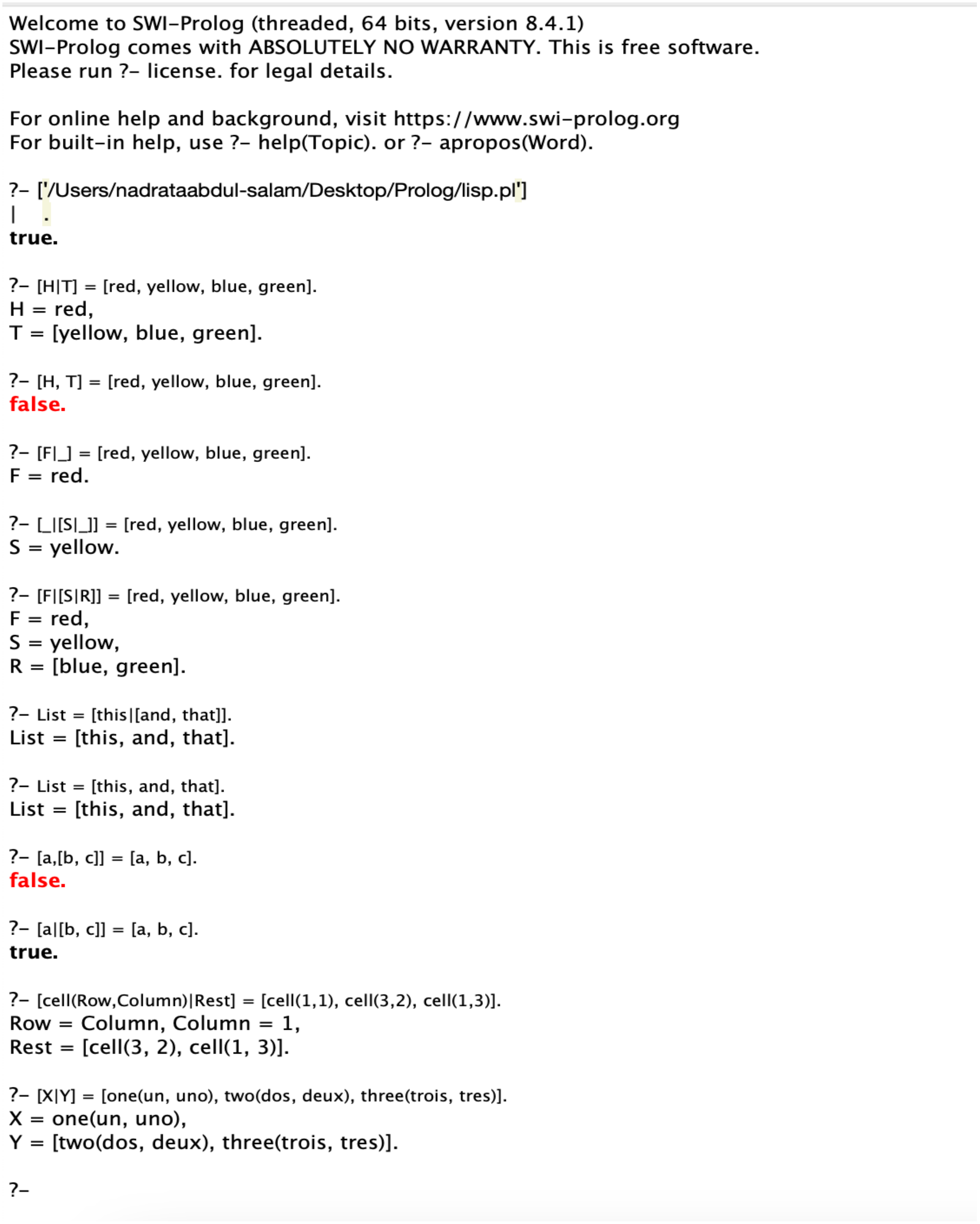


```
pokemon(name(metapod), grass, hp(70), attack(stun-spore, 20)).
pokemon(name(butterfree), grass, hp(130), attack(whirlwind, 80)).

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(Strat,_)), write(Strat), nl, fail.

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

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


% ----------------------------------------------------------------------
tough(Name) :- pokemon(name(Name),_,hp(HAP),_), HAP >= 100.
% ----------------------------------------------------------------------


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

dump_kind(Typ) :- listing(pokemon(_,Typ,_,_)).
% ----------------------------------------------------------------------

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

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

family(Name) :- evolves(Name,NameA), write(Name), write(" "), write(NameA), evolves(NameA,NameB), write(" "), write(NameB).

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



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


families :- cen(Name), evolves(Name,NameA), nl, write(Name), write(" "), write(NameA), evolves(NameA,NameB), write(" "), write(NameB), fail.


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


lineage(Name) :- pokemon(name(Name),_,_,_), listing(pokemon(name(Name),_,_,_)), evolves(Name,NameA), listing(pokemon(name(NameA),_,_,_)), evolves(NameA,NameB), listing(
pokemon(name(NameB),_,_,_)).
```

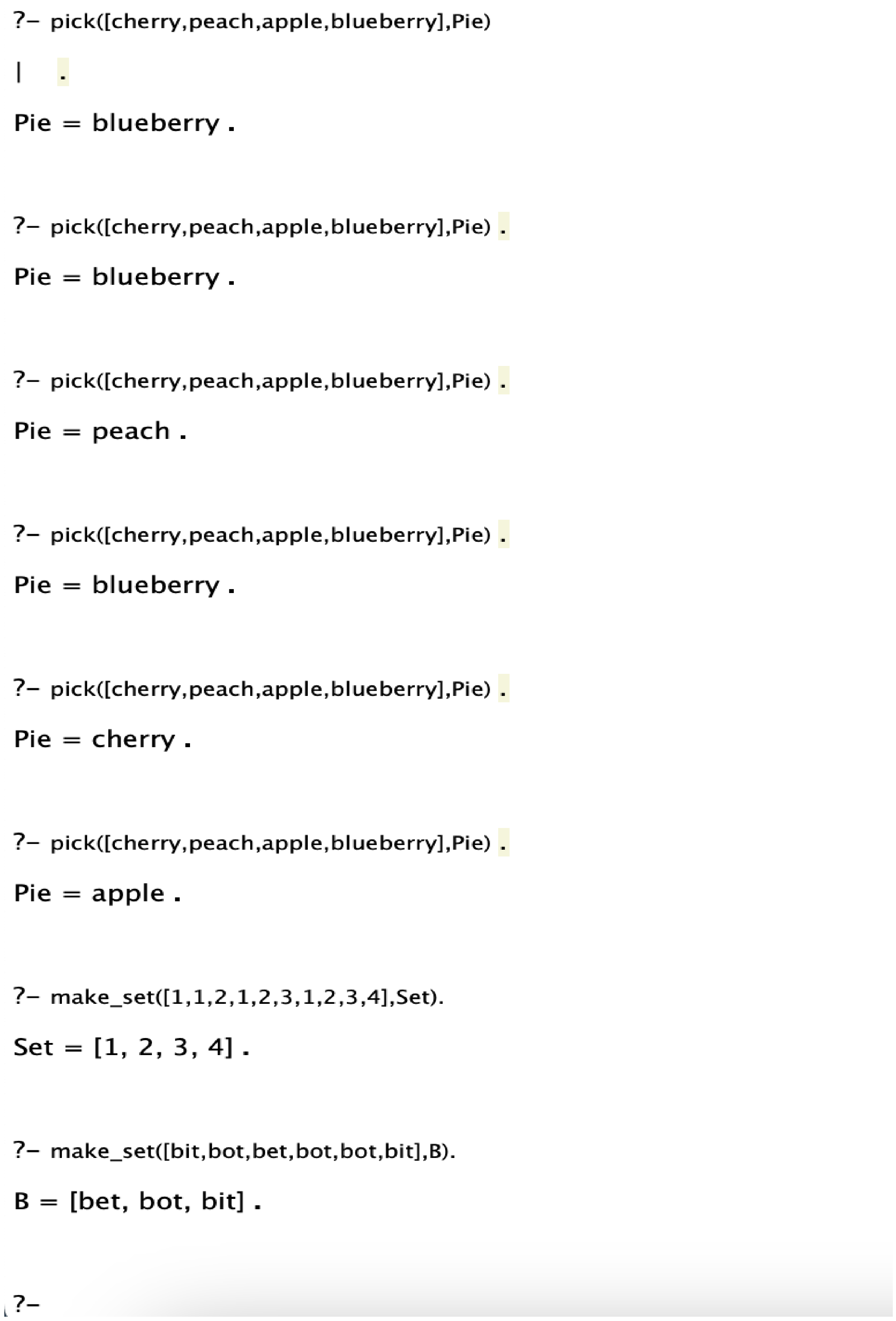
Task 4: Lisp Processing in Prolog



```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.1)
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).

?- ['/Users/nadrataabdul-salam/Desktop/Prolog/lisp.pl']
|    .
true.

?- [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)].

?-
```

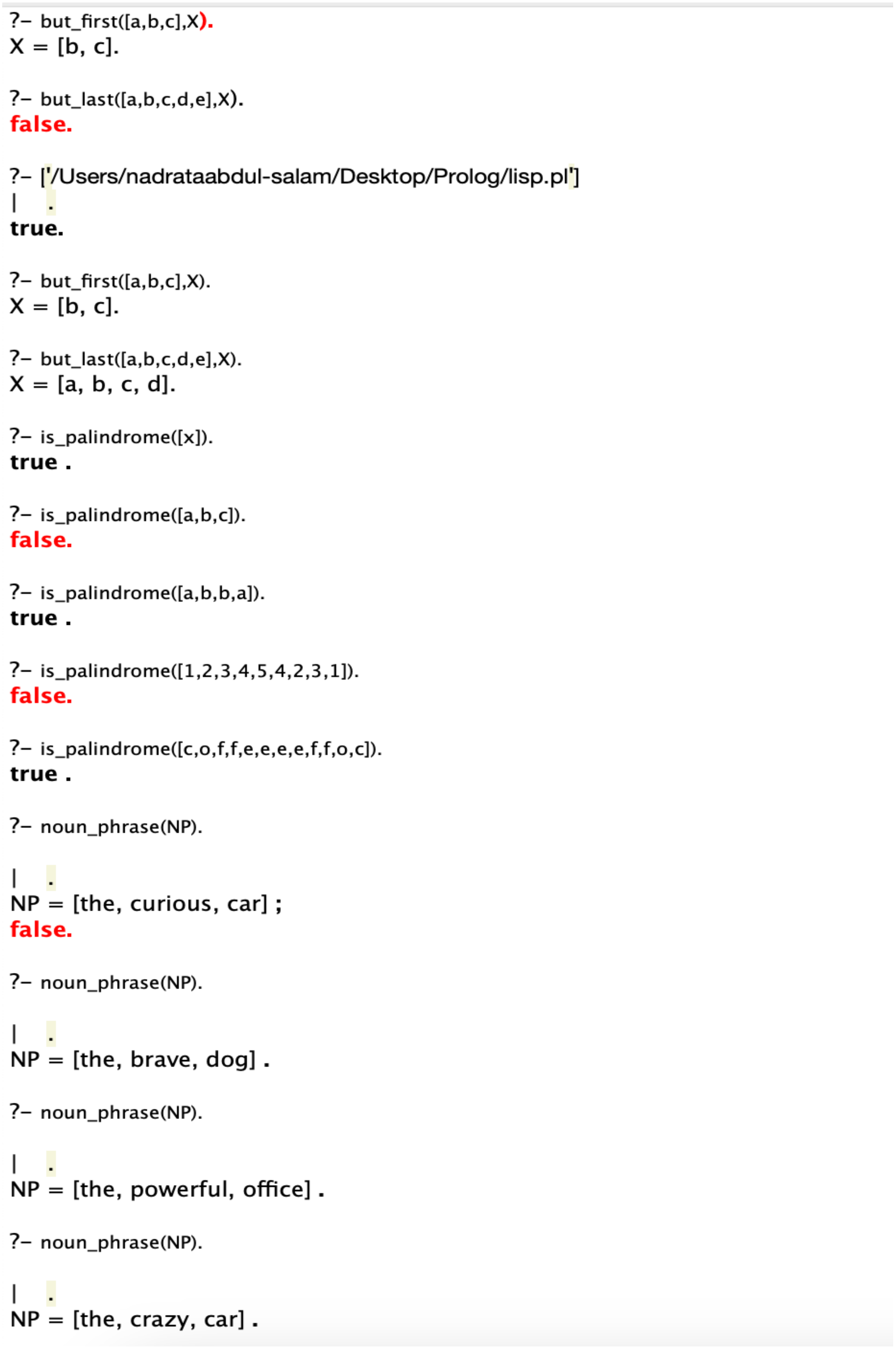
```
?- first([apple],First).
First = apple.

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

?- 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(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 .
```



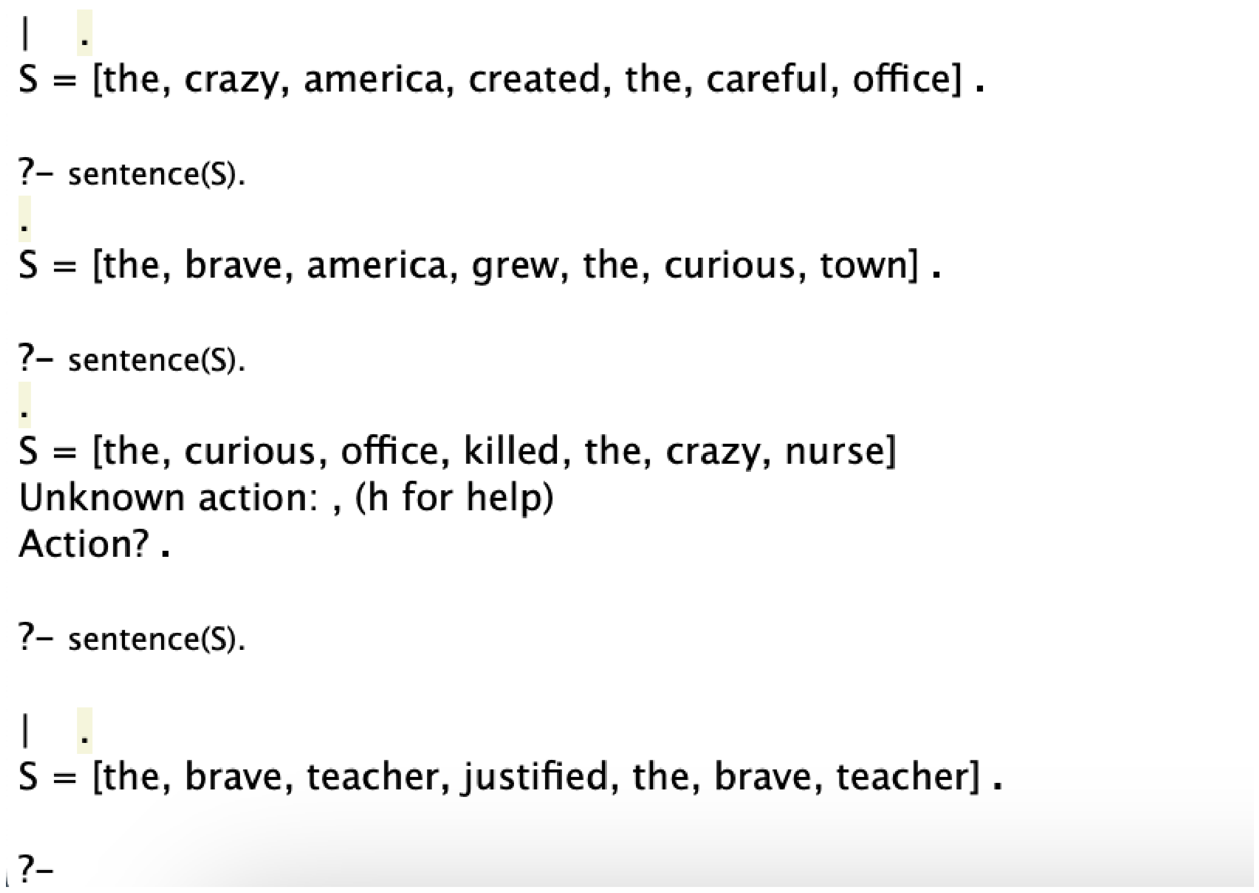
```
?- pick([cherry,peach,apple,blueberry],Pie)
|    .
Pie = blueberry .

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

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

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

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

?- 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] .

?-
```

```
?- ['/Users/nadrataabdul-salam/Desktop/Prolog/lisp.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] .
```

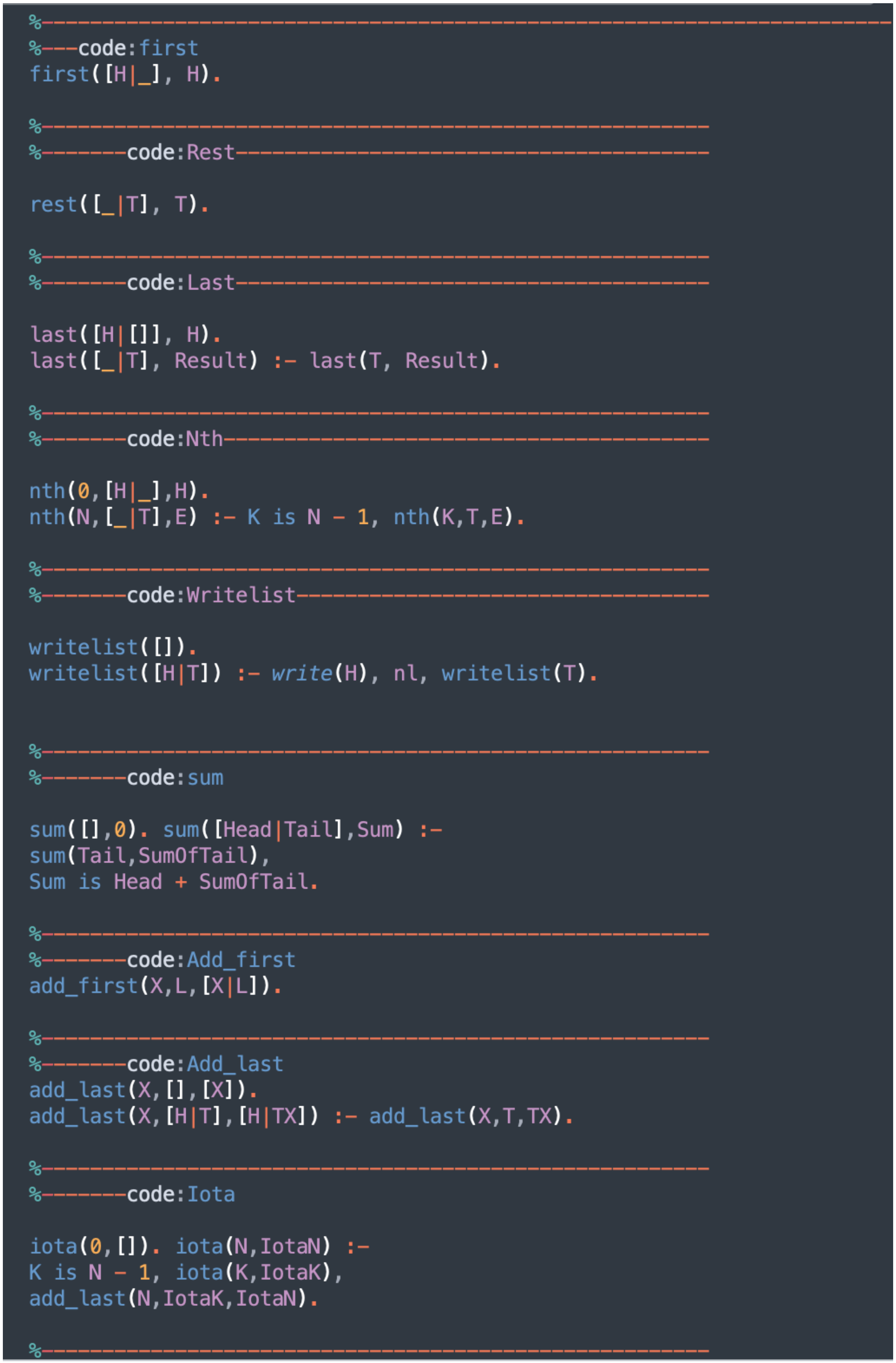


```
?- but_first([a,b,c],X).
X = [b, c].

?- but_last([a,b,c,d,e],X).
false.

?- ['/Users/nadrataabdul-salam/Desktop/Prolog/lisp.pl']
|    .
true.

?- 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, curious, car] ;
false.

?- noun_phrase(NP).

|    .
NP = [the, brave, dog] .

?- noun_phrase(NP).

|    .
NP = [the, powerful, office] .

?- noun_phrase(NP).

|    .
NP = [the, crazy, car] .
```

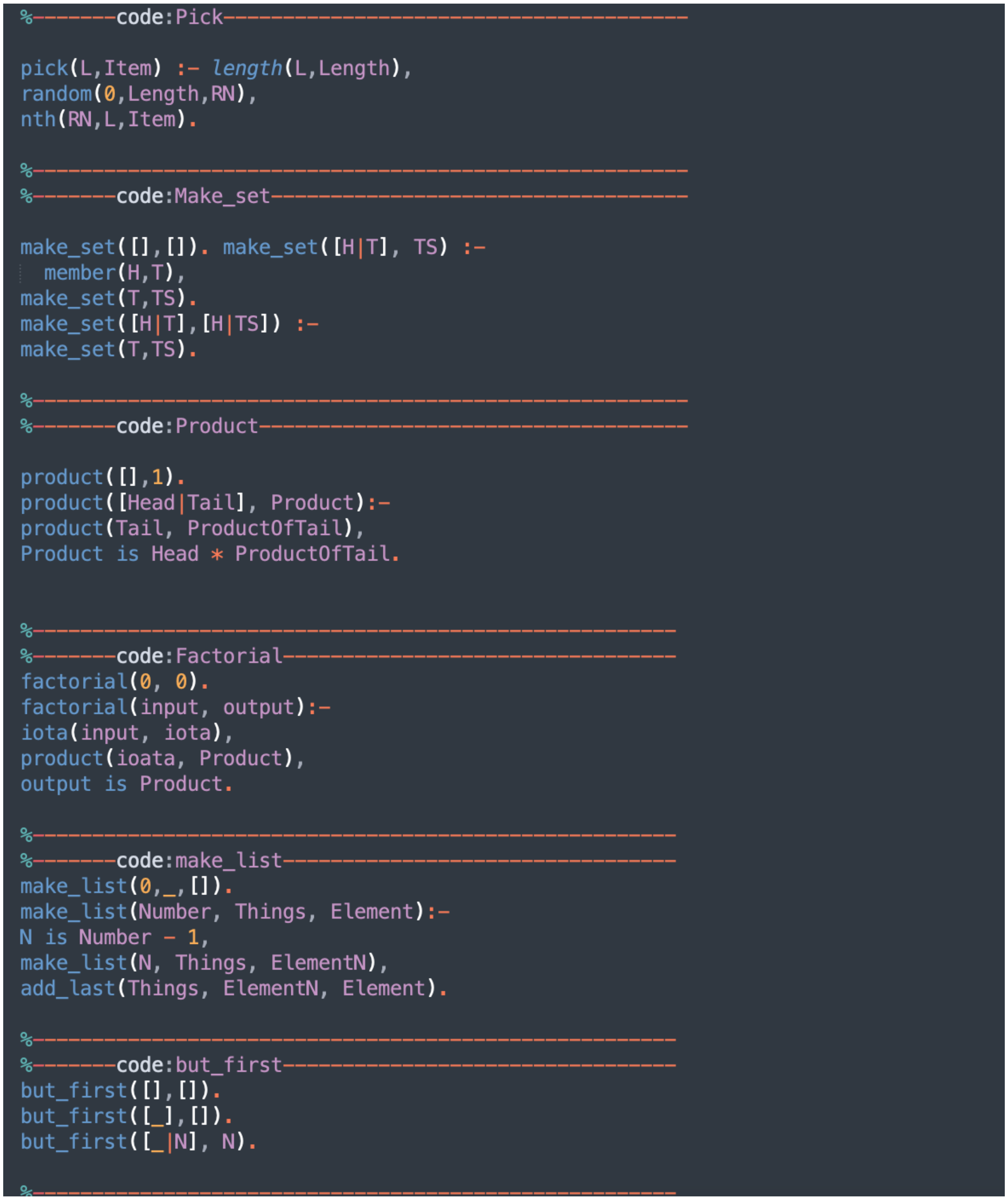
```
?- noun_phrase(NP).

|    .
NP = [the, brave, nurse] .

?- sentence(S).

|    .
S = [the, brave, teacher, advised, the, powerful, teacher] .

?- sentence(S).

|    .
S = [the, clever, america, killed, the, powerful, america] .

?- sentence(S).
.
S = [the, crazy, dog, killed, the, clever, teacher] .

?- sentence(S).

|    .
S = [the, powerful, town, killed, the, powerful, dog] .

?- sentence(S).

|    .
S = [the, powerful, teacher, justified, the, careful, office] .

?- sentence(S).

|    .
S = [the, curious, dog, created, the, clever, dog] .

?- sentence(S).

|    .
S = [the, crazy, dog, advised, the, powerful, car] .

?- sentence(S).

|    .
S = [the, brave, america, lead, the, brave, nurse] .

?- sentence(S).

|    .
S = [the, careful, nurse, justified, the, curious, car] .

?- sentence(S).
```



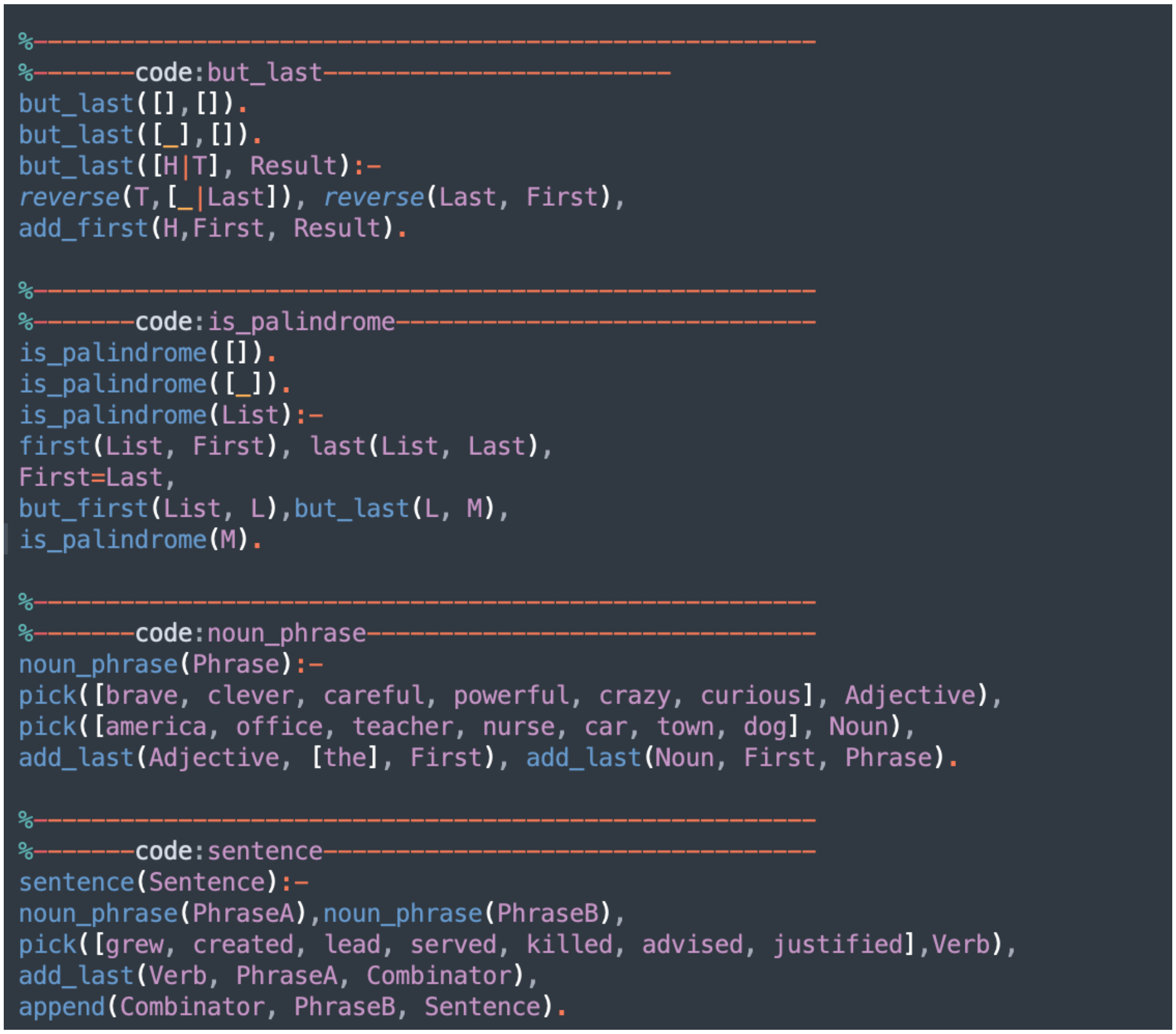
```
|    .
S = [the, crazy, america, created, the, careful, office] .

?- sentence(S).
.
S = [the, brave, america, grew, the, curious, town] .

?- sentence(S).
.
S = [the, curious, office, killed, the, crazy, nurse]
Unknown action: , (h for help)
Action? .

?- sentence(S).

|    .
S = [the, brave, teacher, justified, the, brave, teacher] .

?-
```



```prolog
%------------------------------------------------------------------------------------
%---code:first
first([H|_], H).

%-------------------------------------------------------------
%-------code:Rest----------------------------------------

rest([_|T], T).

%-------------------------------------------------------------
%-------code:Last----------------------------------------

last([H|[]], H).
last([_|T], Result) :- last(T, Result).

%-------------------------------------------------------------
%-------code:Nth-----------------------------------------

nth(0,[H|_],H).
nth(N,[_|T],E) :- K is N - 1, nth(K,T,E).

%-------------------------------------------------------------
%-------code:Writelist-----------------------------------

writelist([]).
writelist([H|T]) :- write(H), nl, writelist(T).


%-------------------------------------------------------------
%-------code:sum

sum([],0). sum([Head|Tail],Sum) :-
sum(Tail,SumOfTail),
Sum is Head + SumOfTail.

%-------------------------------------------------------------
%-------code:Add_first
add_first(X,L,[X|L]).

%-------------------------------------------------------------
%-------code:Add_last
add_last(X,[],[X]).
add_last(X,[H|T],[H|TX]) :- add_last(X,T,TX).

%-------------------------------------------------------------
%-------code:Iota

iota(0,[]). iota(N,IotaN) :-
K is N - 1, iota(K,IotaK),
add_last(N,IotaK,IotaN).

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



```
%-------code:Pick---------------------------------------

pick(L,Item) :- length(L,Length),
random(0,Length,RN),
nth(RN,L,Item).

%--------------------------------------------------------
%-------code:Make_set-----------------------------------

make_set([],[]). make_set([H|T], TS) :-
  member(H,T),
make_set(T,TS).
make_set([H|T],[H|TS]) :-
make_set(T,TS).

%--------------------------------------------------------
%-------code:Product------------------------------------

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


%-------------------------------------------------------
%-------code:Factorial---------------------------------
factorial(0, 0).
factorial(input, output):-
iota(input, iota),
product(ioata, Product),
output is Product.

%-------------------------------------------------------
%-------code:make_list---------------------------------
make_list(0,_,[]).
make_list(Number, Things, Element):-
N is Number - 1,
make_list(N, Things, ElementN),
add_last(Things, ElementN, Element).

%-------------------------------------------------------
%-------code:but_first---------------------------------
but_first([],[]).
but_first([_],[]).
but_first([_|N], N).

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



```
%------------------------------------------------------
%-------code:but_last------------------------
but_last([],[]).
but_last([_],[]).
but_last([H|T], Result):-
reverse(T,[_|Last]), reverse(Last, First),
add_first(H,First, Result).

%------------------------------------------------------
%-------code:is_palindrome-----------------------------
is_palindrome([]).
is_palindrome([_]).
is_palindrome(List):-
first(List, First), last(List, Last),
First=Last,
but_first(List, L),but_last(L, M),
is_palindrome(M).

%------------------------------------------------------
%-------code:noun_phrase-------------------------------
noun_phrase(Phrase):-
pick([brave, clever, careful, powerful, crazy, curious], Adjective),
pick([america, office, teacher, nurse, car, town, dog], Noun),
add_last(Adjective, [the], First), add_last(Noun, First, Phrase).

%------------------------------------------------------
%-------code:sentence----------------------------------
sentence(Sentence):-
noun_phrase(PhraseA),noun_phrase(PhraseB),
pick([grew, created, lead, served, killed, advised, justified],Verb),
add_last(Verb, PhraseA, Combinator),
append(Combinator, PhraseB, Sentence).
```