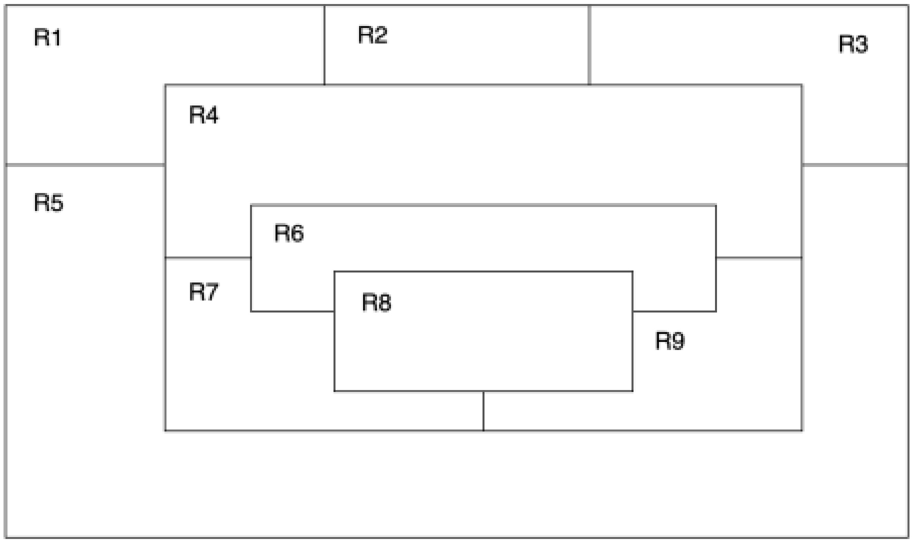
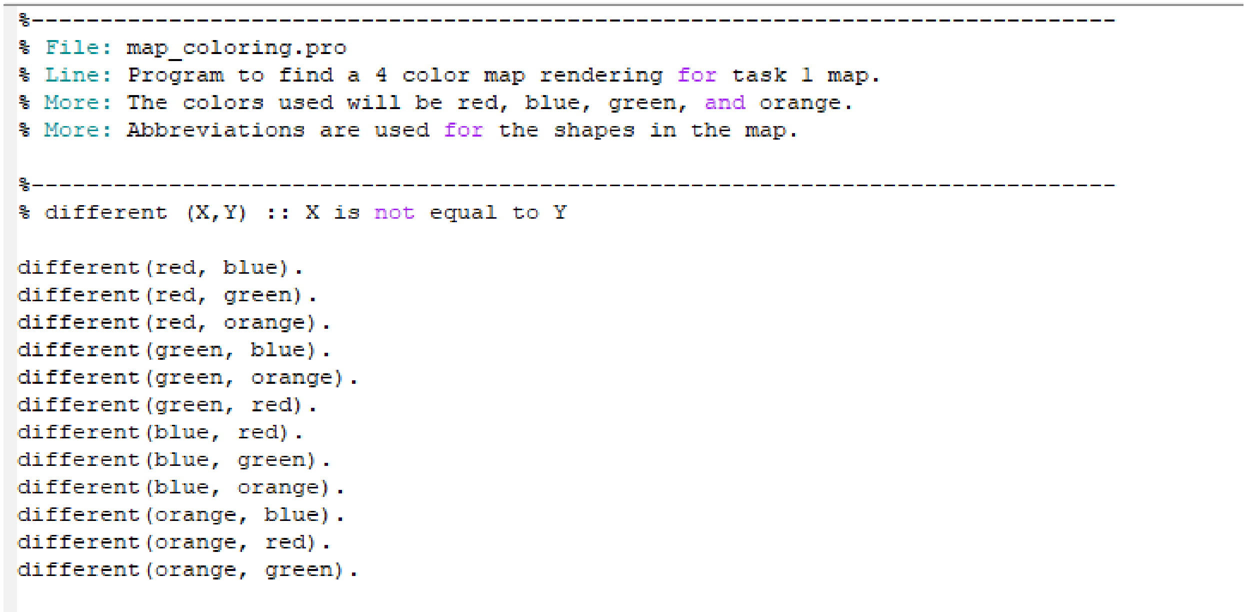
First Prolog Programming Assignment

Abstract

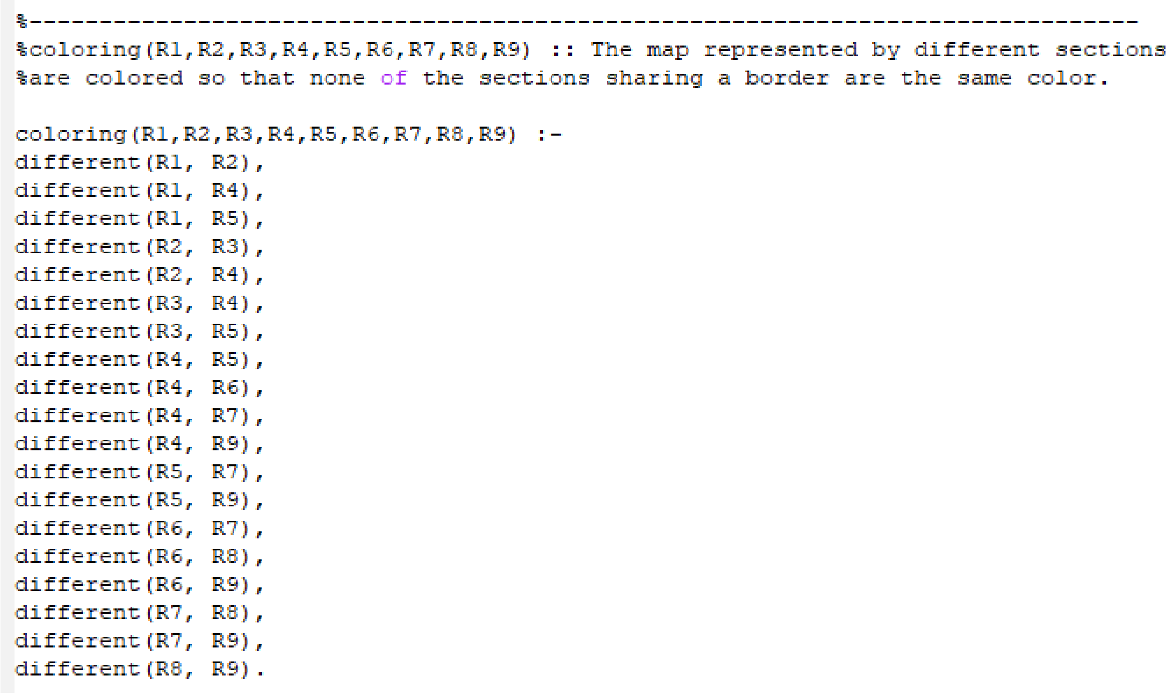
For this assignment I demonstrated the capabilities of Prolog, a language based in predicate logic. Utilizing a Knowledge Base, a collection of facts and rules, I was able to run queries, which return either true or false, based upon the Knowledge Base provided for the program.

Task 1: Map Coloring

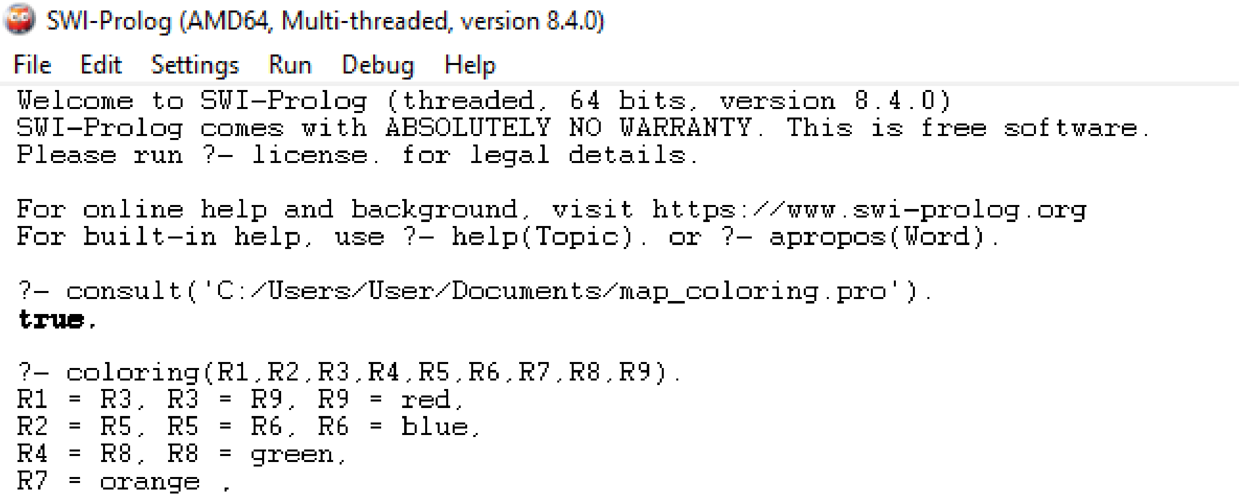




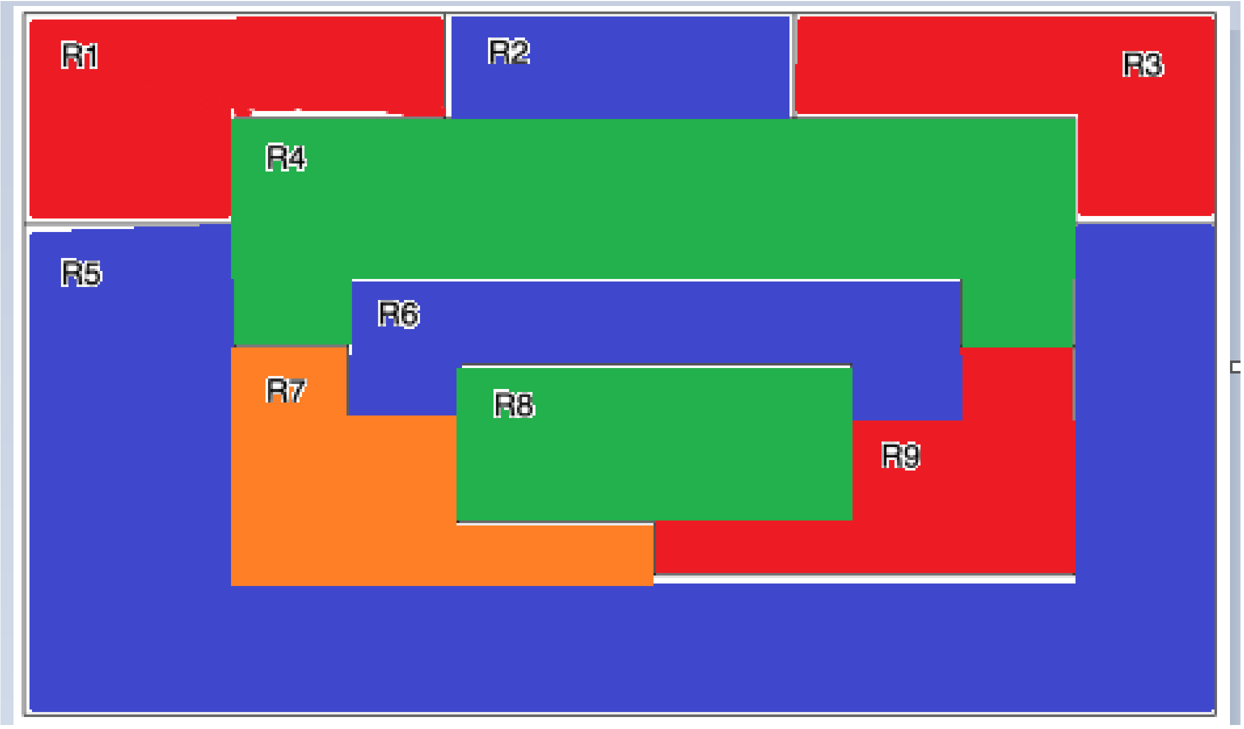
```
%--------------------------------------------------------------------------------
% File: map_coloring.pro
% Line: Program to find a 4 color map rendering for task 1 map.
% More: The colors used will be red, blue, green, and orange.
% More: Abbreviations are used for the shapes in the map.

%--------------------------------------------------------------------------------
% 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, red).
different(blue, green).
different(blue, orange).
different(orange, blue).
different(orange, red).
different(orange, green).
```



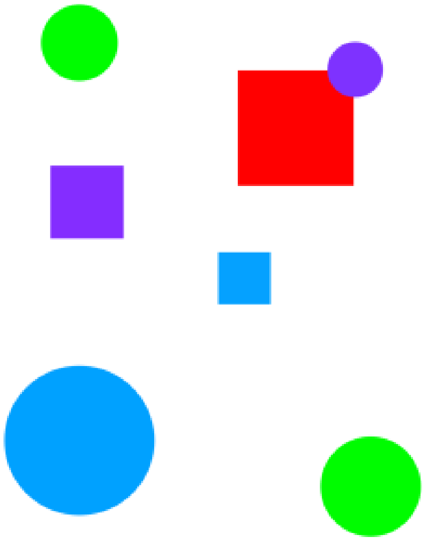
```
%---------------------------------------------------------------------------------
%coloring(R1,R2,R3,R4,R5,R6,R7,R8,R9) :: The map represented by different sections
%are colored so that none of the sections sharing a border are the same color.

coloring(R1,R2,R3,R4,R5,R6,R7,R8,R9) :-
different(R1, R2),
different(R1, R4),
different(R1, R5),
different(R2, R3),
different(R2, R4),
different(R3, R4),
different(R3, R5),
different(R4, R5),
different(R4, R6),
different(R4, R7),
different(R4, R9),
different(R5, R7),
different(R5, R9),
different(R6, R7),
different(R6, R8),
different(R6, R9),
different(R7, R8),
different(R7, R9),
different(R8, R9).
```

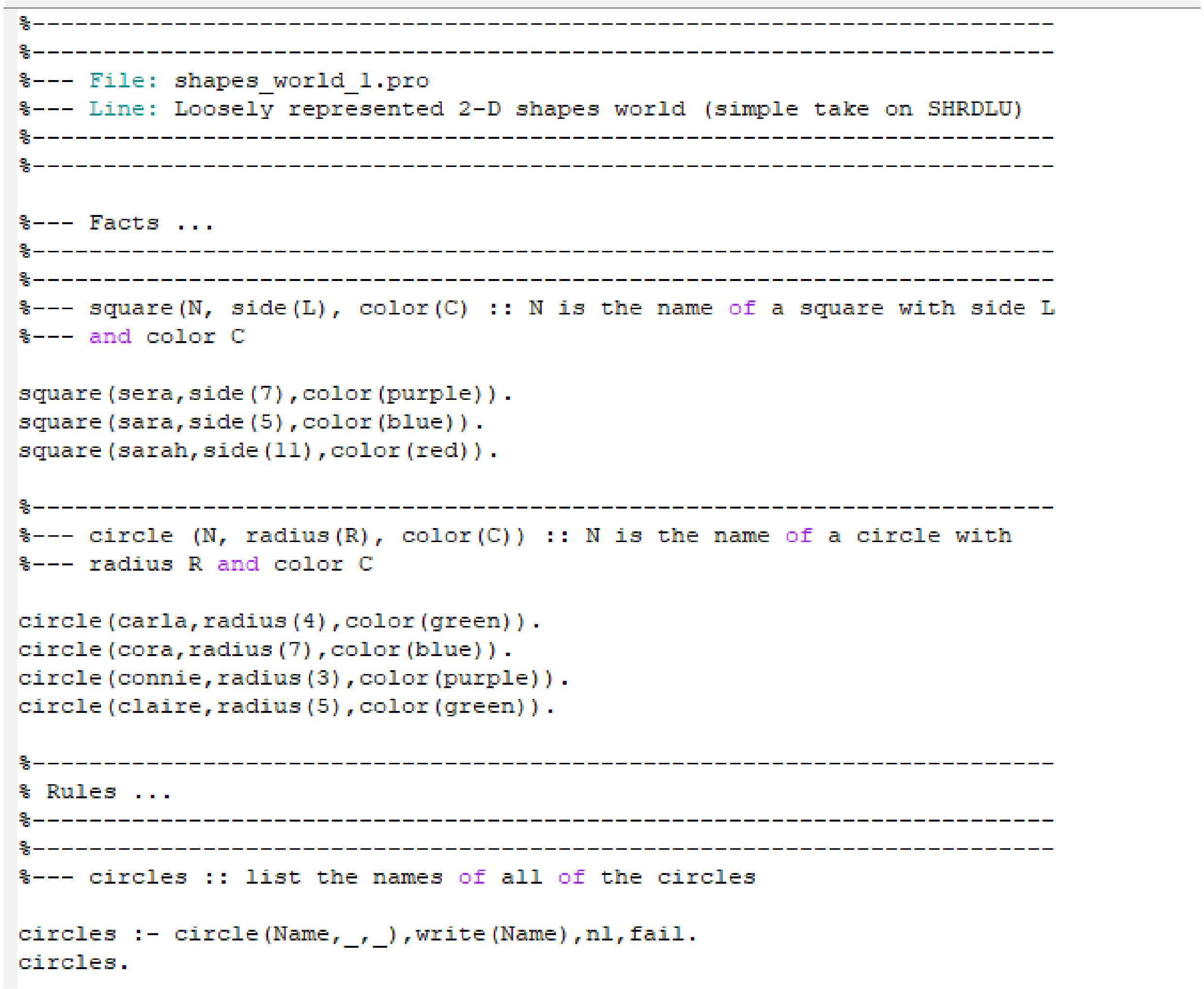


```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.0)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

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

?- consult('C:/Users/User/Documents/map_coloring.pro').
true.

?- coloring(R1,R2,R3,R4,R5,R6,R7,R8,R9).
R1 = R3, R3 = R9, R9 = red,
R2 = R5, R5 = R6, R6 = blue,
R4 = R8, R8 = green,
R7 = orange .
```

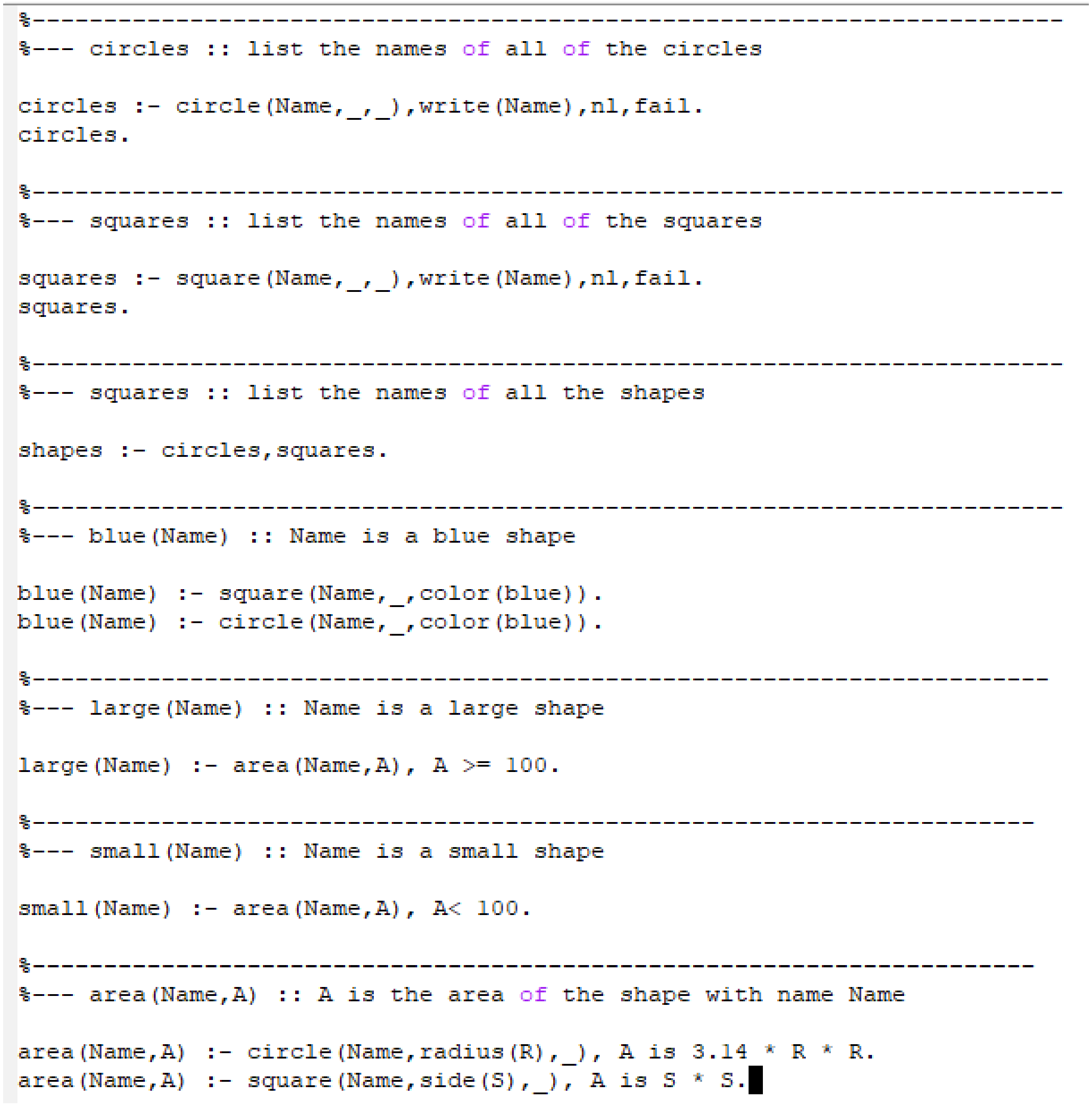


Task 2: The Floating Shapes World

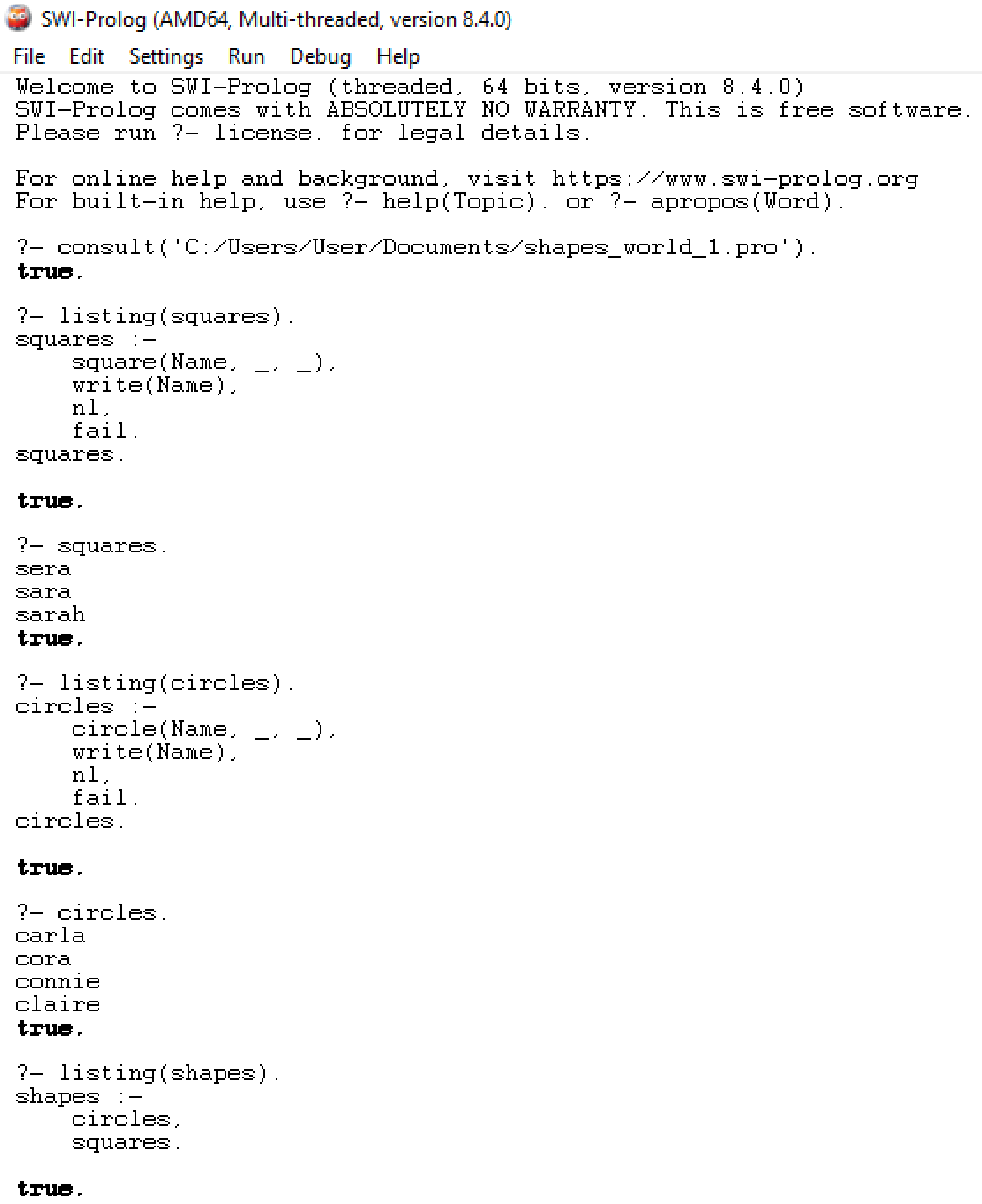
Image




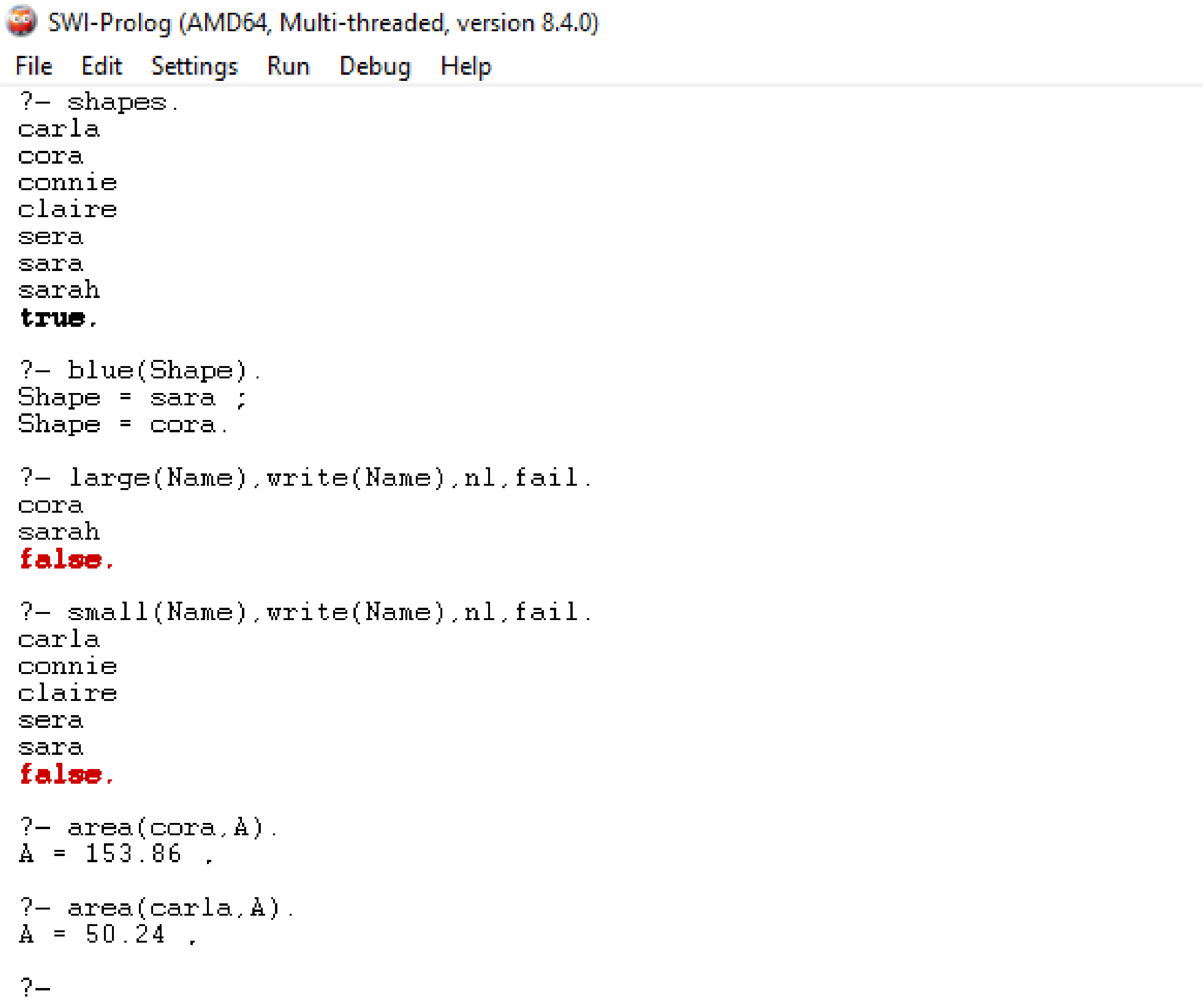
```
%------------------------------------------------------------------------
%------------------------------------------------------------------------
%--- File: shapes_world_1.pro
%--- Line: Loosely represented 2-D shapes world (simple take on SHRDLU)
%------------------------------------------------------------------------
%------------------------------------------------------------------------

%--- Facts ...
%------------------------------------------------------------------------
%------------------------------------------------------------------------
%--- square(N, side(L), color(C) :: N is the name of a square with side L
%--- and color C

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

%------------------------------------------------------------------------
%--- circle (N, radius(R), color(C)) :: N is the name of a circle with
%--- radius R and color C

circle(carla,radius(4),color(green)).
circle(cora,radius(7),color(blue)).
circle(connie,radius(3),color(purple)).
circle(claire,radius(5),color(green)).

%------------------------------------------------------------------------
% Rules ...
%------------------------------------------------------------------------
%------------------------------------------------------------------------
%--- circles :: list the names of all of the circles

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



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

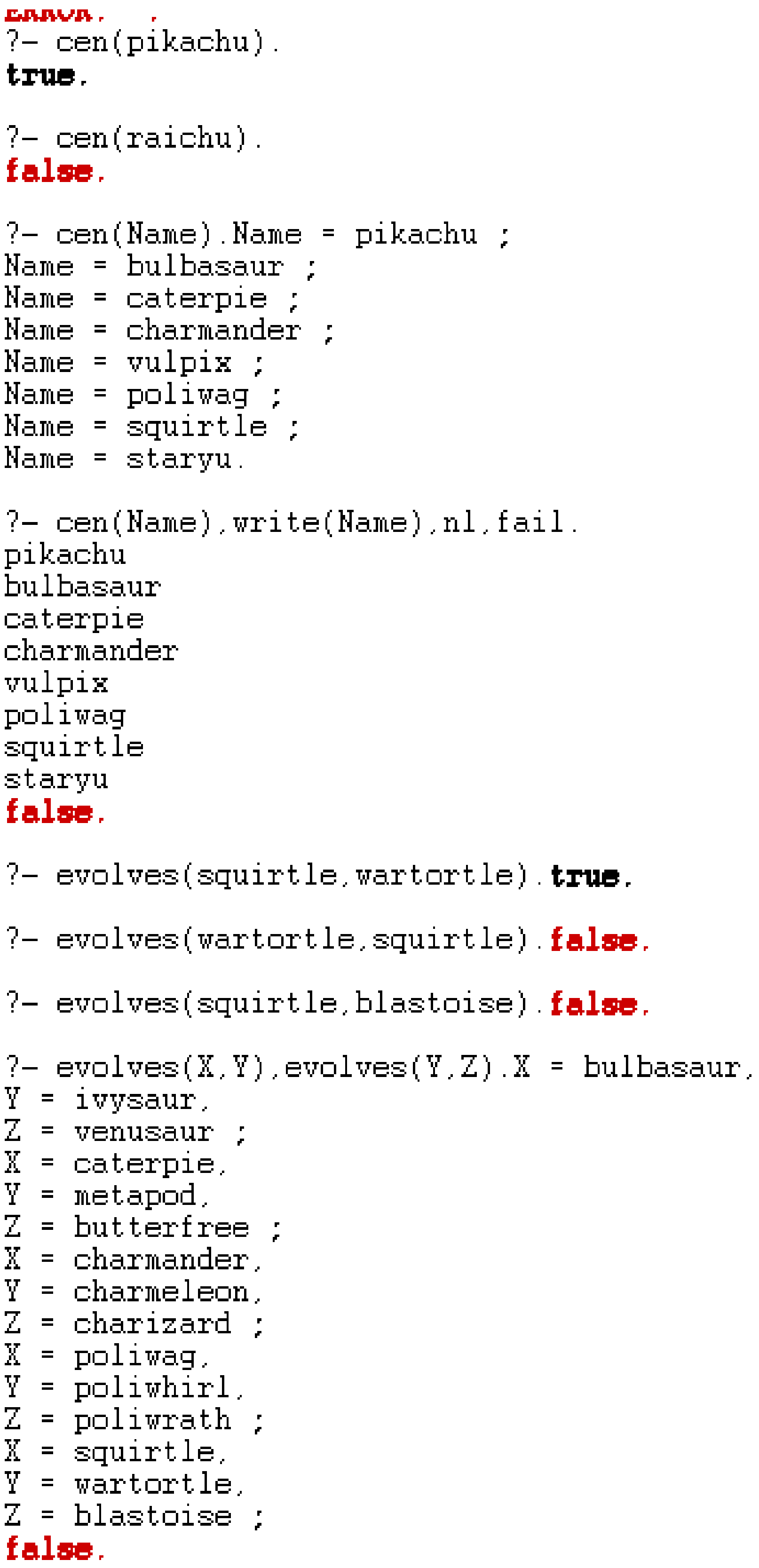

SWI-Prolog (AMD64, Multi-threaded, version 8.4.0)
File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.0)
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).

?- consult('C:/Users/User/Documents/shapes_world_1.pro').
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.
```

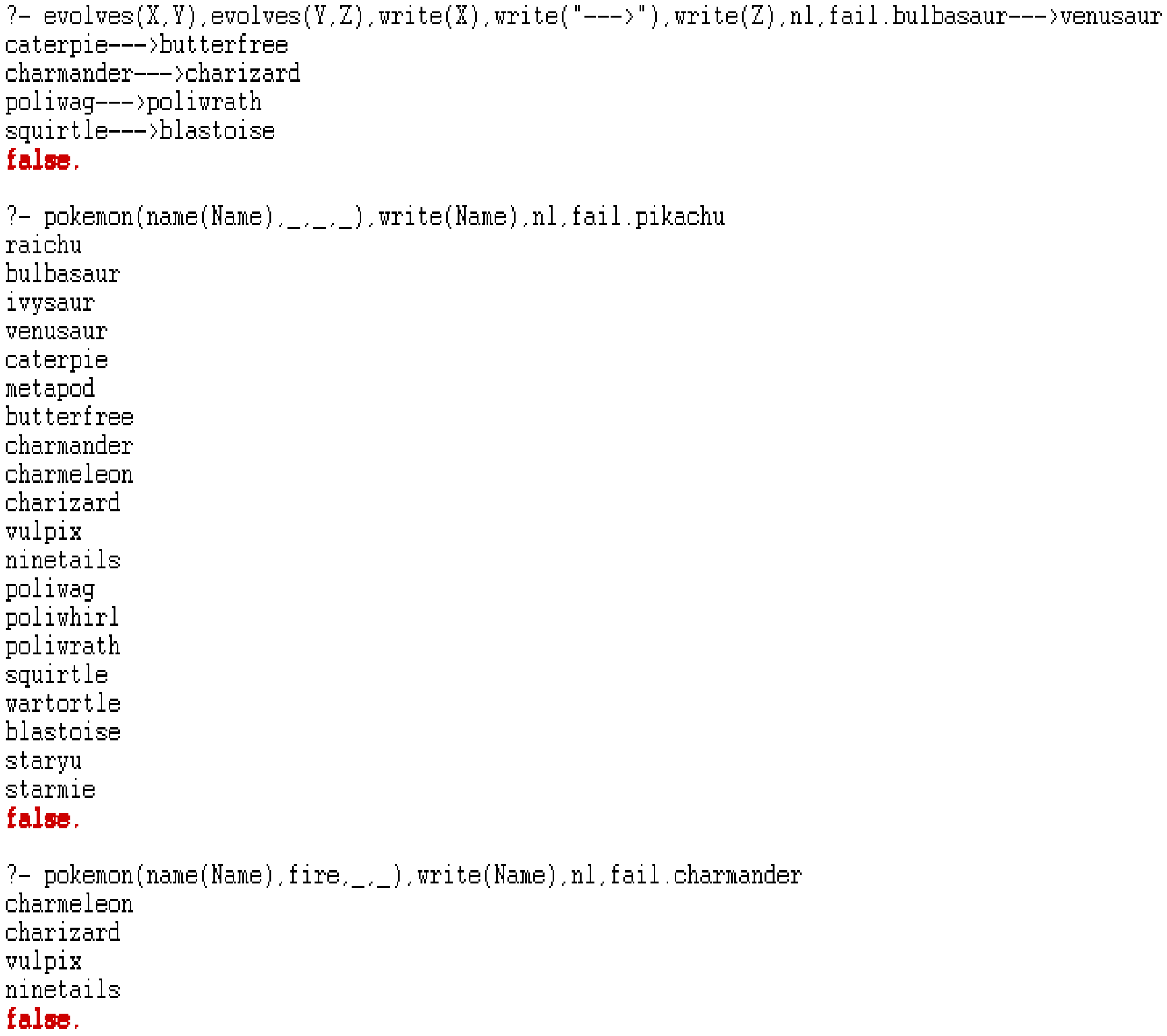


```
?- shapes.
carla
cora
connie
claire
sera
sara
sarah
true.

?- 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 .

?-
```

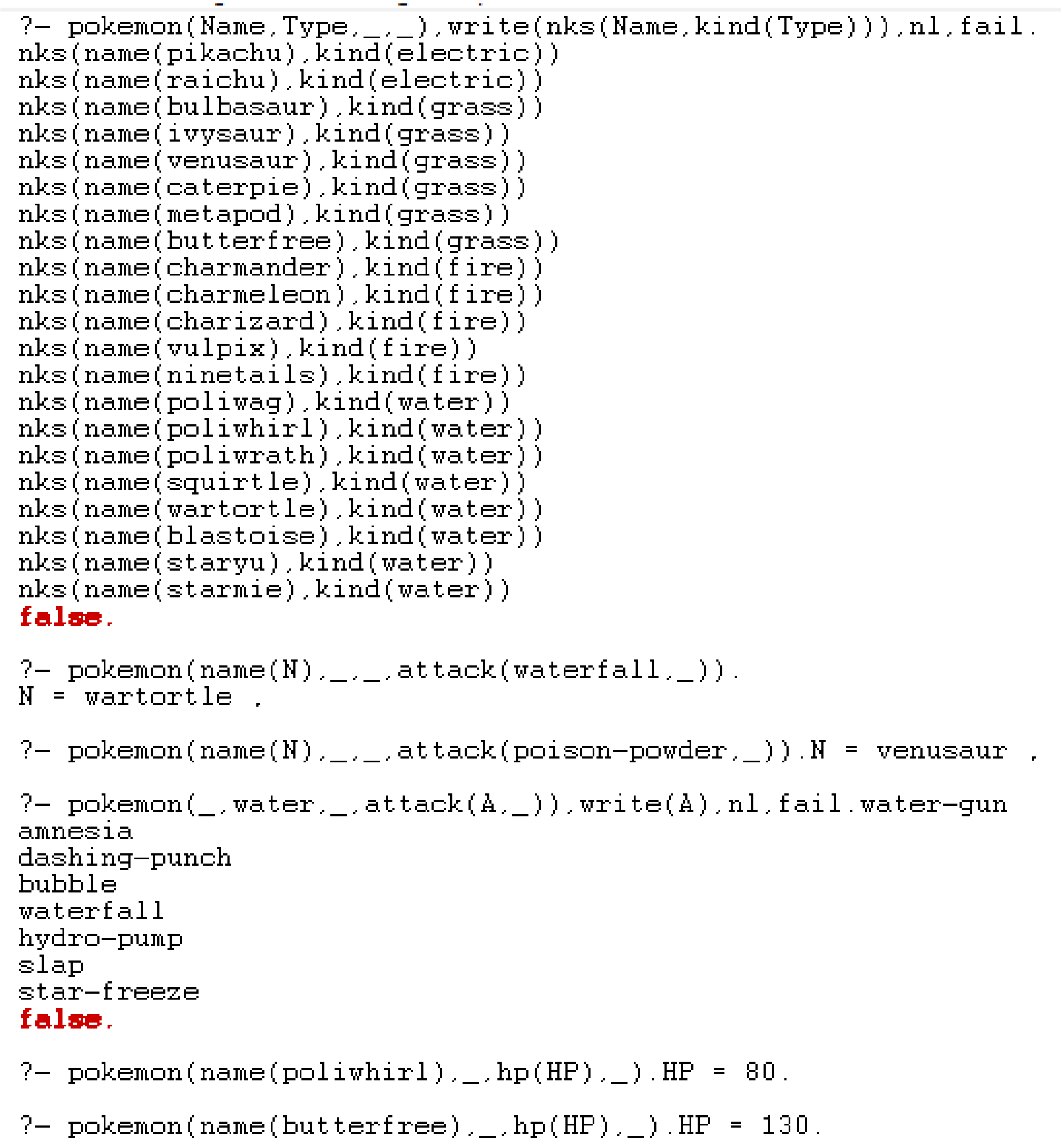
Task 3: Pokemon KB Interaction and Programming



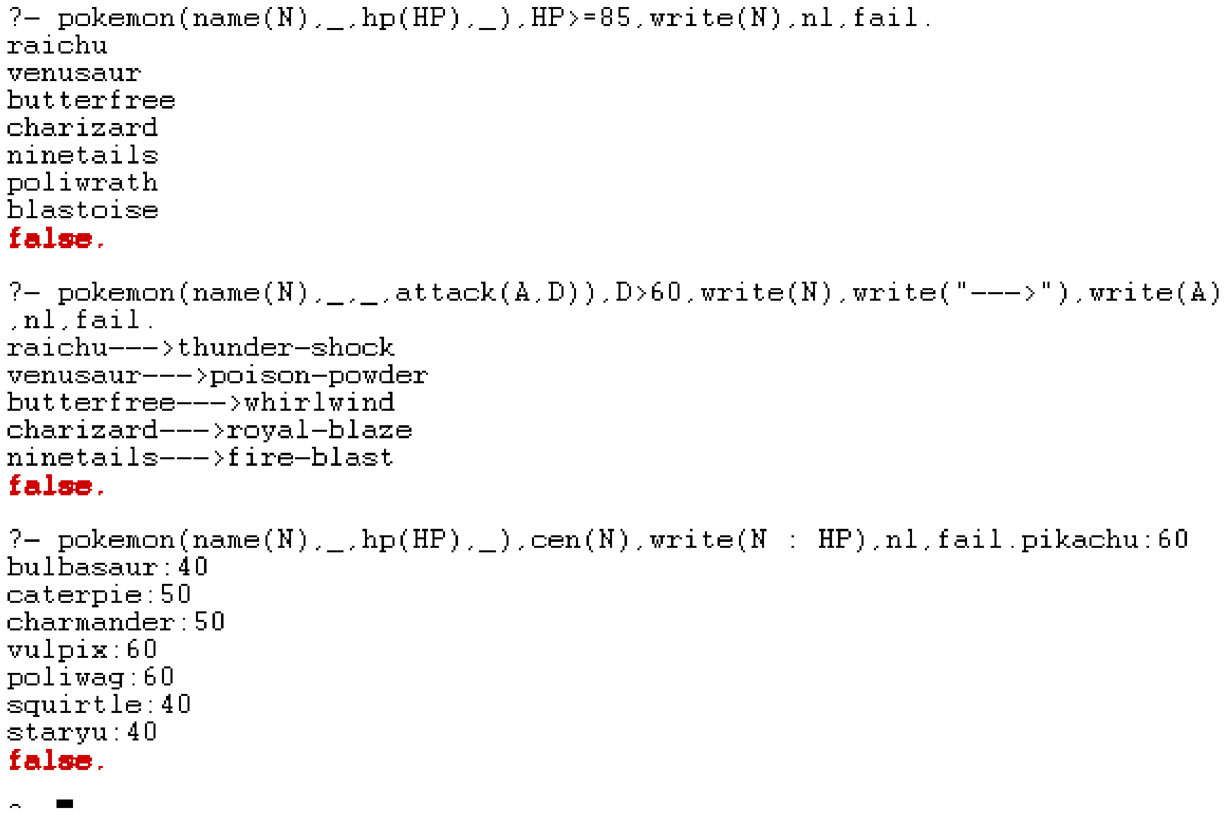
```
false.
?- cen(pikachu).
true.

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

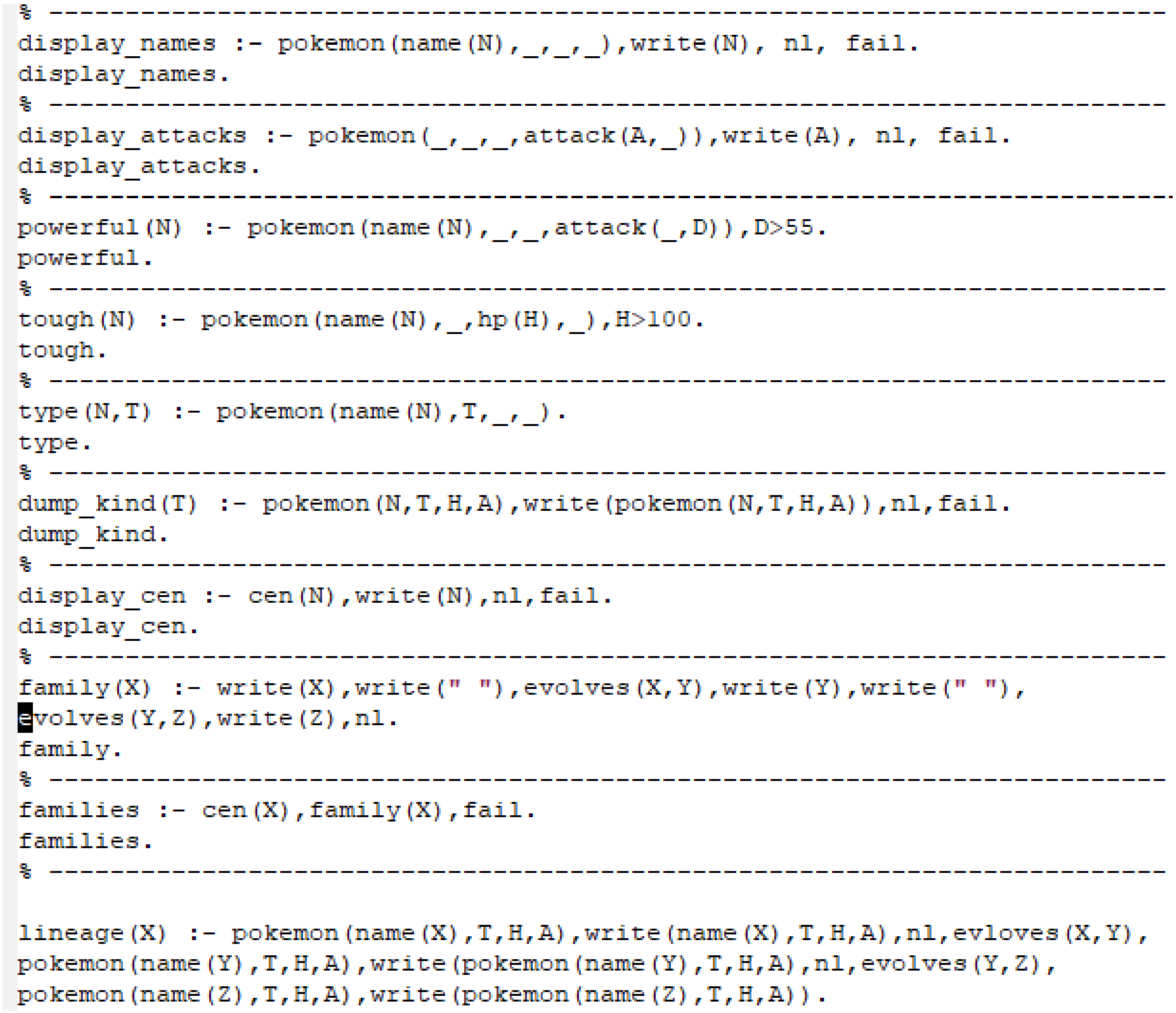


```
?- pokemon(Name,Type,_,_),write(nks(Name,kind(Type))),nl,fail.
nks(name(pikachu),kind(electric))
nks(name(raichu),kind(electric))
nks(name(bulbasaur),kind(grass))
nks(name(ivysaur),kind(grass))
nks(name(venusaur),kind(grass))
nks(name(caterpie),kind(grass))
nks(name(metapod),kind(grass))
nks(name(butterfree),kind(grass))
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(HP),_).HP = 80.

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

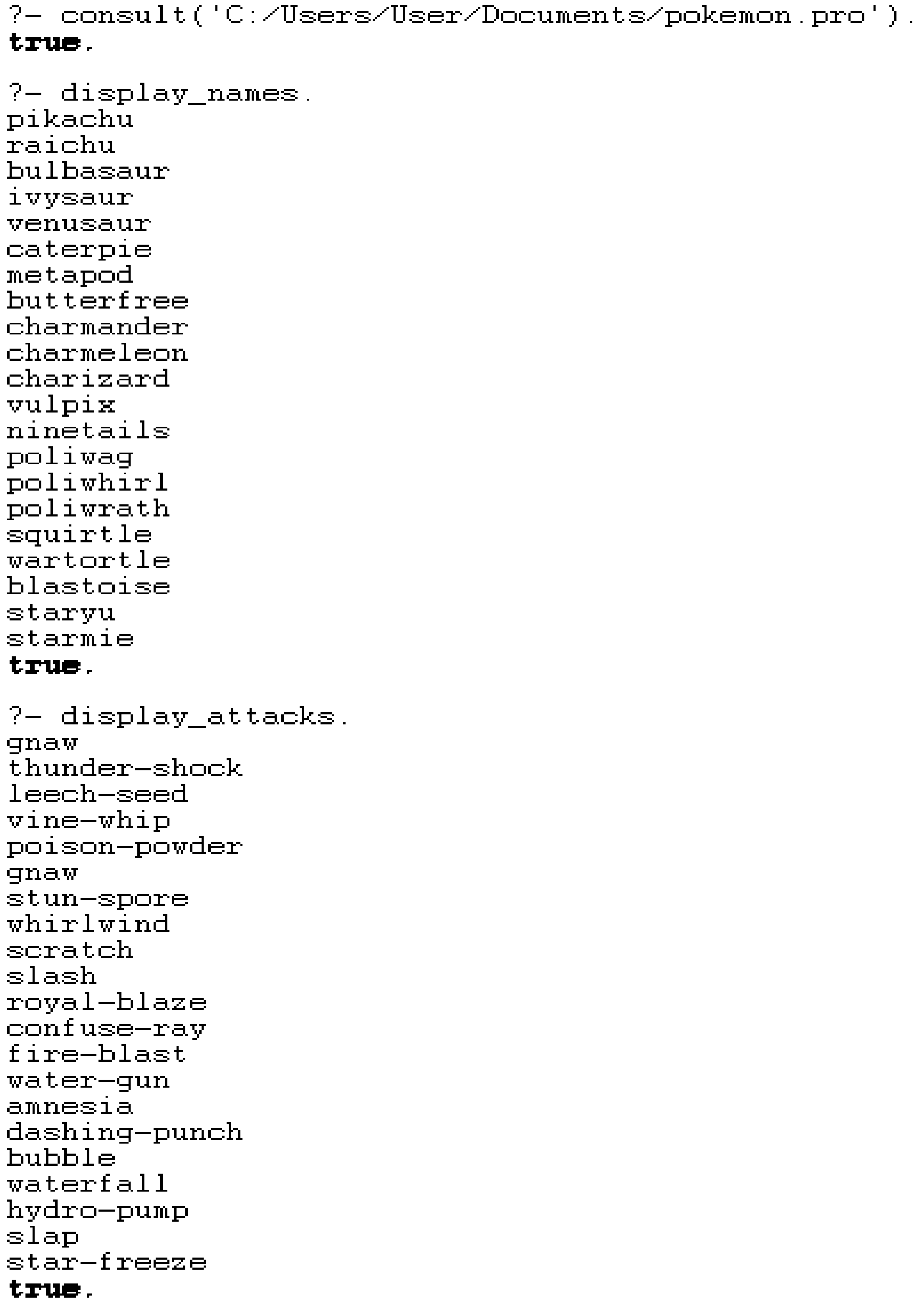


```
?- pokemon(name(N),_,hp(HP),_),HP>=85,write(N),nl,fail.
raichu
venusaur
butterfree
charizard
ninetails
poliwrath
blastoise
false.

?- pokemon(name(N),_,_,attack(A,D)),D>60,write(N),write("--->"),write(A)
,nl,fail.
raichu--->thunder-shock
venusaur--->poison-powder
butterfree--->whirlwind
charizard--->royal-blaze
ninetails--->fire-blast
false.

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

?-
```

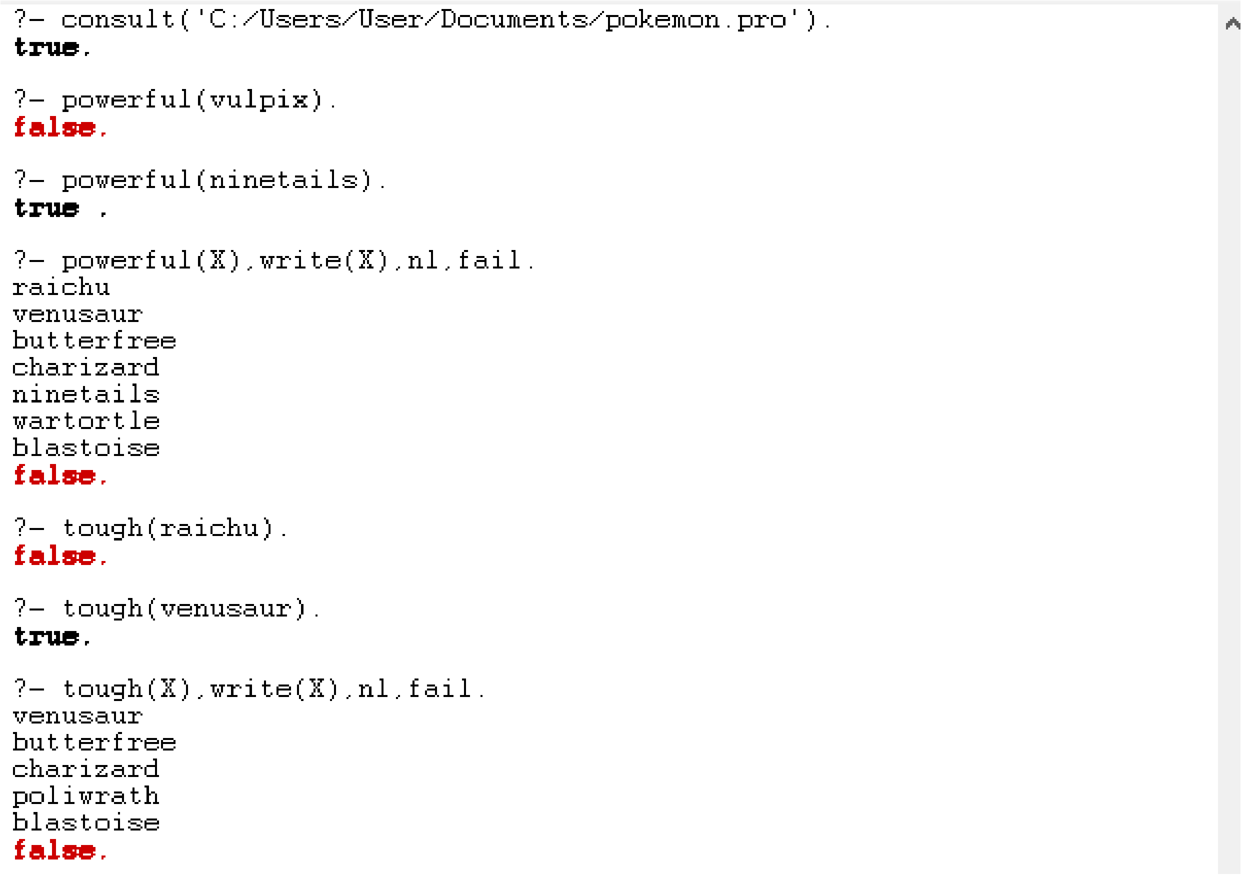
Part 2 – Program



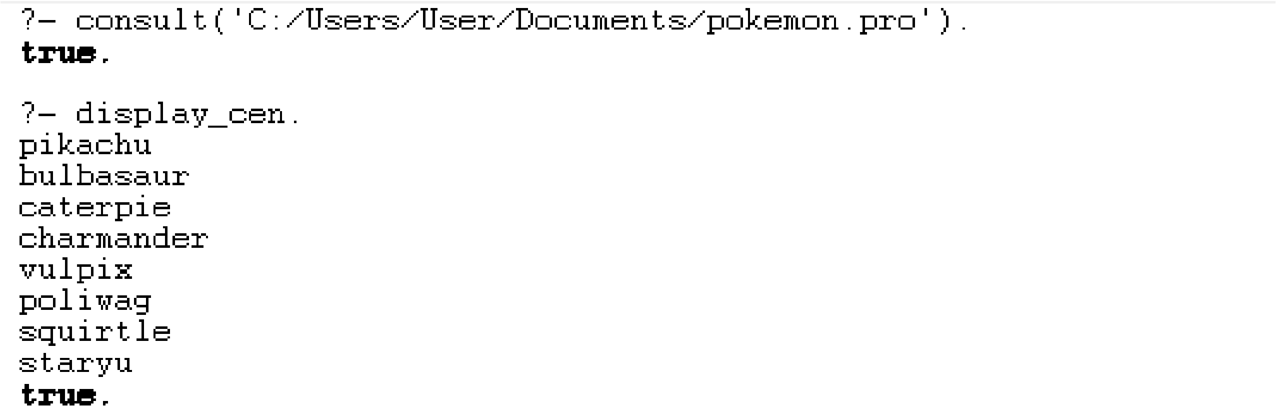
```
% -------------------------------------------------------------------------
display_names :- pokemon(name(N),_,_,_),write(N), nl, fail.
display_names.
% -------------------------------------------------------------------------
display_attacks :- pokemon(_,_,_,attack(A,_)),write(A), nl, fail.
display_attacks.
% -------------------------------------------------------------------------
powerful(N)  :- pokemon(name(N),_,_,attack(_,D)),D>55.
powerful.
% -------------------------------------------------------------------------
tough(N)  :- pokemon(name(N),_,hp(H),_),H>100.
tough.
% -------------------------------------------------------------------------
type(N,T)  :- pokemon(name(N),T,_,_).
type.
% -------------------------------------------------------------------------
dump_kind(T)  :- pokemon(N,T,H,A),write(pokemon(N,T,H,A)),nl,fail.
dump_kind.
% -------------------------------------------------------------------------
display_cen :- cen(N),write(N),nl,fail.
display_cen.
% -------------------------------------------------------------------------
family(X)  :- write(X),write(" "),evolves(X,Y),write(Y),write(" "),
evolves(Y,Z),write(Z),nl.
family.
% -------------------------------------------------------------------------
families :- cen(X),family(X),fail.
families.
% -------------------------------------------------------------------------

lineage(X)  :- pokemon(name(X),T,H,A),write(name(X),T,H,A),nl,evloves(X,Y),
pokemon(name(Y),T,H,A),write(pokemon(name(Y),T,H,A)),nl,evolves(Y,Z),
pokemon(name(Z),T,H,A),write(pokemon(name(Z),T,H,A)).
```



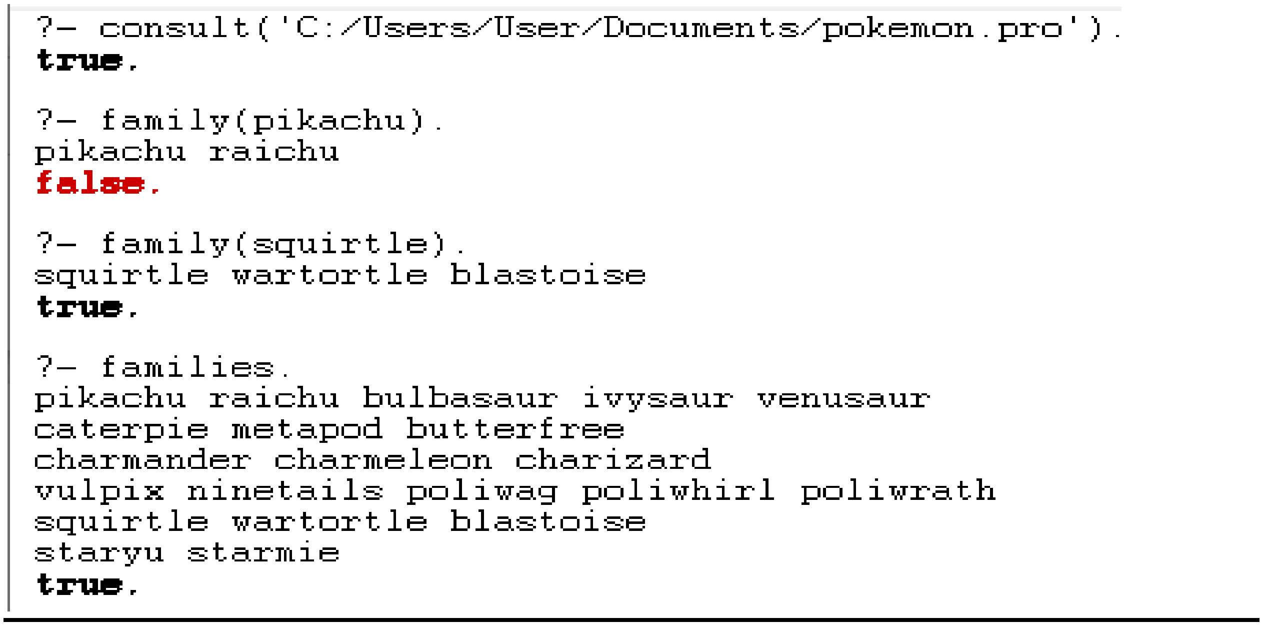
```
?- consult('C:/Users/User/Documents/pokemon.pro').
true.

?- 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.
```



```
?- consult('C:/Users/User/Documents/pokemon.pro').
true.

?- powerful(vulpix).
false.

?- powerful(ninetails).
true .

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

?- tough(raichu).
false.

?- tough(venusaur).
true.

?- tough(X),write(X),nl,fail.
venusaur
butterfree
charizard
poliwrath
blastoise
false.
```

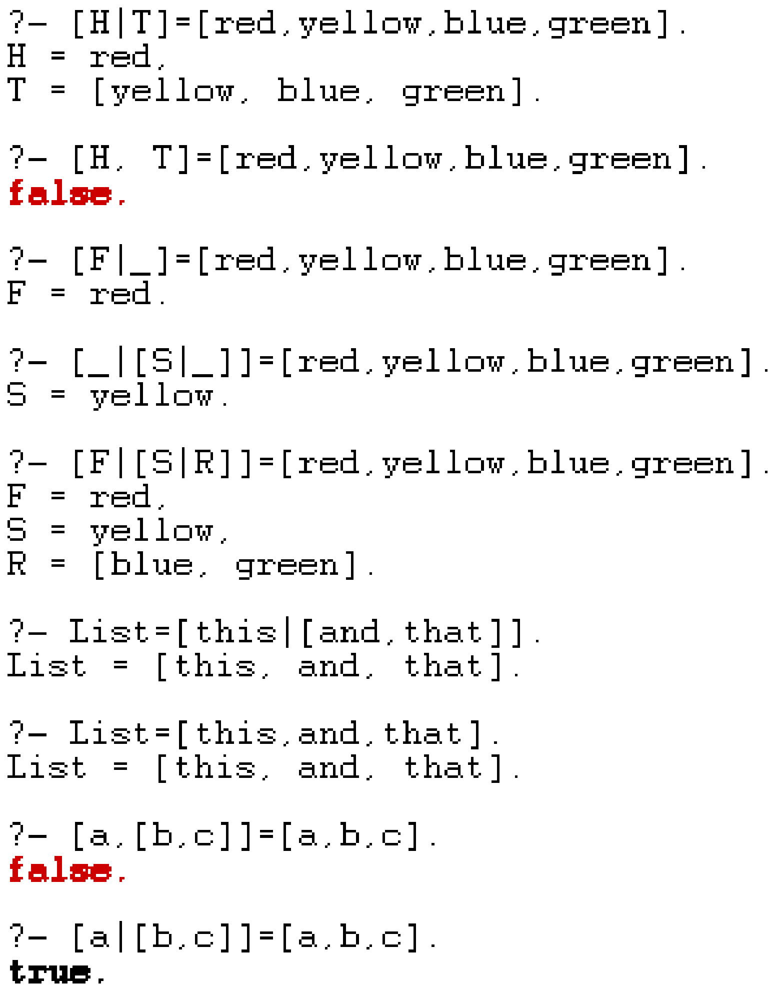


```
?- consult('C:/Users/User/Documents/pokemon.pro').
true.

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

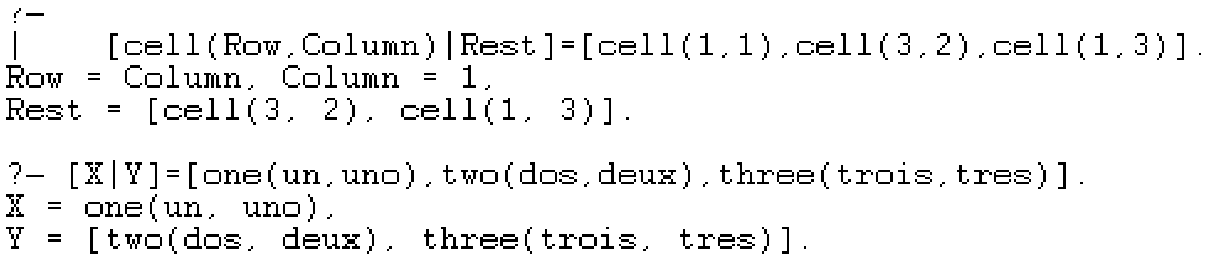


```
?- consult('C:/Users/User/Documents/pokemon.pro').
true.

?- 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
true.
```

Task 4 LISP processing in Prolog



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

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

?- [F|_]=[red,yellow,blue,green].
F = red.

?- [_|[S|_]]=[red,yellow,blue,green].
S = yellow.

?- [F|[S|R]]=[red,yellow,blue,green].
F = red,
S = yellow,
R = [blue, green].

?- List=[this|[and,that]].
List = [this, and, that].

?- List=[this,and,that].
List = [this, and, that].

?- [a,[b,c]]=[a,b,c].
false.

?- [a|[b,c]]=[a,b,c].
true.
```



```
?-
|    [cell(Row,Column)|Rest]=[cell(1,1),cell(3,2),cell(1,3)].
Row = Column, Column = 1,
Rest = [cell(3, 2), cell(1, 3)].

?- [X|Y]=[one(un,uno),two(dos,deux),three(trois,tres)].
X = one(un, uno),
Y = [two(dos, deux), three(trois, tres)].
```