Prolog Assignment #1: Various Computations

Learning Abstract

For this assignment, we had taken a break from using Racket to learn the concept of Prolog. Prolog is a programming language that helps conceptualize logic by making rules and facts turn into a world of knowledge. It is also a great database of theoretical thought. Each test looks at a different aspect of the uses of Prolog.

Task 1 - Map Coloring

Code

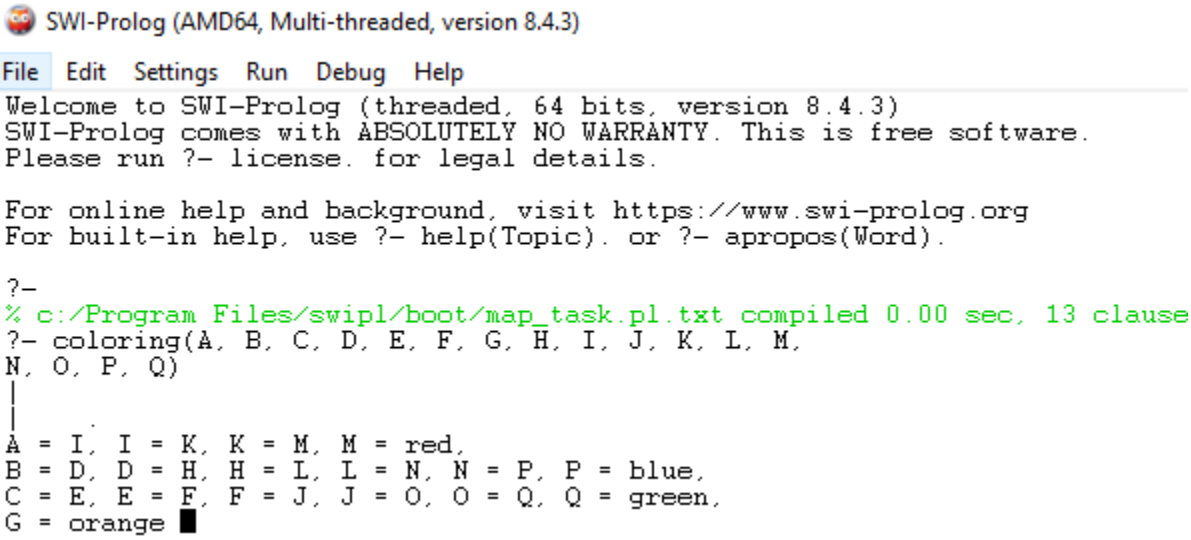
map_task.pro - Notepad

File Edit Format View Help

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

% ----------------------------------------------------
% Sections of the map colored so no sections sharing a border are the same color
% ----------------------------------------------------

coloring(A, B, C, D, E, F, G, H, I, J, K, L, M,
N, O, P, Q):-
different(A, B),
different(A, C),
different(A, D),
different(A, E),
different(B, C),
different(B, E),
different(C, D),
different(D, E),
different(B, F),
different(B, I),
different(C, H),
different(C, I),
```

```
different(D, J),
different(D, K),
different(E, L),
different(E, M),
different(F, G),
different(F, M),
different(G, H),
different(H, I),
different(I, J),
different(J, K),
different(L, M),
different(F, N),
different(G, O),
different(H, O),
different(I, P),
different(J, P),
different(K, Q),
different(L, Q),
different(M, N),
different(N, O),
different(N, Q),
different(O, P),
different(P, Q).
```

Demo



Task 2 - The Floating Shapes World

Code

floating_shape_world.pro - Notepad

File Edit Format View Help

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

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

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

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

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

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

shapes :-
circles,
squares.

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

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

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

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

Demo

SWI-Prolog (AMD64, Multi-threaded, version 8.4.3)

File Edit Settings Run Debug Help

```
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).

?-
% c:/Users/dmmit/floating_shape_world.pro.txt compiled 0.00 sec, 18 clauses
?- listing(squares).
squares :-
    square(Name, _, _),
    write(Name),
    nl,
    fail.
squares.

true.

?- square.
Correct to: "squares"? yes
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 ;

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

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

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

?- area(core, A).
false.

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

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

Task 3 - Pokemon KB Interaction and Programming

Code

```
cen(pikachu).
cen(bulbasaur).
cen(caterpie).
cen(charmander).
cen(vulpix).
cen(poliwag).
cen(squirtle).
cen(staryu).

evolve(pikachu,raichu).
evolve(bulbasaur,ivysaur).
evolve(ivysaur,venusaur).
evolve(caterpie,metapod).
evolve(metapod,butterfree).
evolve(charmander,charmeleon)
evolve(charmeleon,charizard).
evolve(vulpix,ninetails).
evolve(poliwag,poliwhirl).
evolve(poliwhirl,poliwrath).
evolve(squirtle,wartortle).
evolve(wartortle,blastoise).
evolve(staryu,starmie).
```

```
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)).
```

```
info(T) :-
pokemon(N, T, HP, A),
write(pokemon(N, T, HP, A)), nl,
fail.

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

family(N) :-
evolve(N, X),
evolve(X, Y),
write(N), write(' '), write(X), write(' '), write(Y).
family(N) :-

evolve(N, X),
write(N), write(' '), write(X).
```

```
families :-
cen(N),
family(N), nl,
fail.
families.
```

```
lineage(N) :-
evolve(N, X),
evolve(X, Y),
pokemon(name(N), T1, HP1, A1),
pokemon(name(X), T2, HP2, A2),
pokemon(name(Y), T3, HP3, A3),
write(pokemon(name(N), T1, HP1, A1)), nl,
write(pokemon(name(X), T2, HP2, A2)), nl,
write(pokemon(name(Y), T3, HP3, A3)).
lineage(N) :-
evolve(N, X),
pokemon(name(N), T1, HP1, A1),
pokemon(name(X), T2, HP2, A2),
write(pokemon(name(N), T1, HP1, A1)), nl,
write(pokemon(name(X), T2, HP2, A2)).
lineage(N) :-
pokemon(name(N), T, HP, A),
write(pokemon(name(N), T, HP, A)).
```

Demo

```
?-
% c:/Users/dmmit/pokemon_info.pro.txt compiled 0.00 sec, 59 clauses
?- 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.
```

```
?- strong(pikachu).
false.

?- strong(blastoise).
true .

?- strong(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 ,

?- type(N, electric).
N = pikachu ;
N = raichu.
```

```
?- type(N, water), write(N), nl, fail.
poliwag
poliwhirl
poliwrath
squirtle
wartortle
blastoise
staryu
starmie
false.

?- info(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.

?- info(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.
```

```
?- families.
pikachu raichu
bulbasaur ivysaur venusaur
bulbasaur ivysaur
caterpie metapod butterfree
caterpie metapod
charmander charmeleon charizard
charmander charmeleon
vulpix ninetails
poliwag poliwhirl poliwrath
poliwag poliwhirl
squirtle wartortle blastoise
squirtle wartortle
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(metapod).
pokemon(name(metapod),grass,hp(70),attack(stun-spore,20))
pokemon(name(butterfree),grass,hp(130),attack(whirlwind,80))
true .

?- lineage(butterfree).
pokemon(name(butterfree),grass,hp(130),attack(whirlwind,80))
true.

?-
```

Task 4 - 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).

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(Num, Factorial) :-
iota(Num, Iota),
product(Iota, Product),
Factorial is Product.
make_list(0, _, _).
make_list(Occurences, Item, List) :-
K is Occurences - 1,
make_list(K, Item, ListK),
add_last(Item, ListK, List).
% This is pretty much the same as last...?
but_first([_|Cdr], Cdr).
but_last(L, List) :-
reverse(L, FirstPass),
but_first(FirstPass, Cdr),
```

```
reverse(Cdr, List).
is_palindrome([]).
is_palindrome(List) :-
length(List, L),
L = 1.
is_palindrome(List) :-
first(List, H),
last(List, L),
L = H,
but_last(List, Truncated),
but_first(Truncated, Final),
is_palindrome(Final).
noun_phrase(NP) :-
pick([potato, tomato, habanero, taquito, emoji, shirt, controller,
fridge], Noun),
pick([spicy, large, evil, deceptive, annoying, boring], Adjective),
NP = [the, Adjective, Noun].
sentence(S) :-
noun_phrase(NP1),
noun_phrase(NP2),
pick([ate, destroyed, wrote, buried, vetoed, threw, stamped], Verb),
append(NP1, [Verb], S0),
append(S0, NP2, S).
```

Demo

```
?- noun_phrase(NP).
NP = [the, annoying, shirt] ,

?- noun_phrase(NP).
NP = [the, deceptive, tomato] ,

?- noun_phrase(NP).
NP = [the, large, fridge] ,

?- noun_phrase(NP).
NP = [the, spicy, fridge] ,

?- noun_phrase(NP).
NP = [the, large, potato] ,

?- sentence(S).
S = [the, evil, tomato, buried, the, annoying, habanero] ,

?- sentence(S).
S = [the, deceptive, habanero, wrote, the, annoying, shirt] ,

?- sentence(S).
S = [the, spicy, habanero, stamped, the, boring, taquito] ,

?- sentence(S).
S = [the, large, shirt, threw, the, deceptive, habanero] ,

?- sentence(S).
S = [the, large, shirt, wrote, the, deceptive, tomato] ,

?- sentence(S).
S = [the, boring, tomato, vetoed, the, evil, taquito] ,

?- sentence(S).
S = [the, boring, taquito, ate, the, spicy, taquito] ,

?- sentence(S).
S = [the, large, controller, threw, the, boring, taquito] ,

?- sentence(S).
S = [the, annoying, tomato, destroyed, the, deceptive, fridge] ,

?- sentence(S).
S = [the, annoying, emoji, threw, the, large, emoji] ,

?- sentence(S).
S = [the, large, potato, wrote, the, annoying, emoji] ,

?- 
```