# Prolog Programming Assignment #1: Various Computations

## What's It All About?

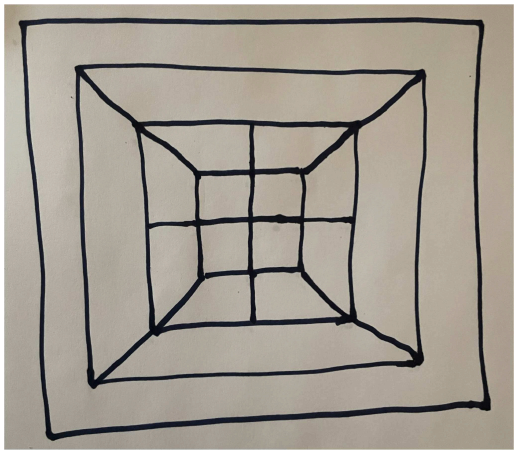
Programming exercises that focus on knowledge representation, search, and list processing in Prolog.

## Tasks

1. Working within a nice text editor and with a good Prolog interpreter, do the following:
   (a) Task 1: Write a map coloring program, by analogy with the map coloring program provided in class, which appears in Lesson 3.
   (b) Task 2: Mimic the shapes world programming and interaction presented in class, all of which is recorded in Lesson 4.
   (c) Task 3: Interact with and extend a given KB regarding Pokemon trading cards.
   (d) Task 4: With reference to Lesson 5, perform some lisp processing interactions, and write and test some programs list processing programs.
2. Craft a nicely structured document that contains representations of each of the four tasks that you were just asked to do. Moreover, be sure to title the document, and place a "learning abstract" just after the title, before presenting your work on each of the four tasks.
3. Post your document to you web work site.

## Task 1: Map Coloring

Working by analogy with the map coloring program provided in class, which you can find in Lesson 3, write a map coloring program to solve the problem of coloring the following map in four colors.



Then, find a way to color the map according to the solution produced by your program, for inclusion in your presentation document.

## Presentational Notes for Task 1

Place the following items within the "Task 1: Map Coloring" section of your presentation document:

1. An image of the given map, with the regions labelled.
2. Your source program.
3. The demo of your program.
4. An image of the map colored according to the output of your program.

## Task 2: The Floating Shapes World

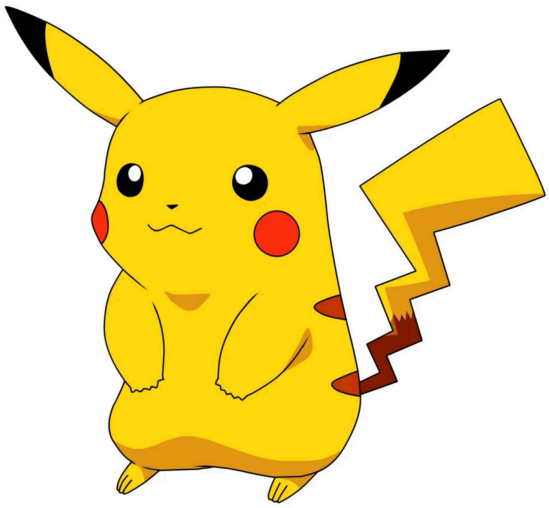
Please mindfully type in the Prolog code for the floating shapes world KB presented in class, which appears in Lesson 4. Then, carefully interact with the shapes world KB to mimic the demo that is provided in the lesson.

## Presentational Notes for Task 2

Place the following items within the "Task 2: The Floating Shapes World" section of your presentation document:

1. The image presented in the lesson.
2. The Prolog KB
3. The demo that you generate (corresponding to that presented the lesson).

## Task 3: Pokemon KB Interaction and Programming



## Preliminary Note

For this task, you will need to incorporate, into your computational world, the knowledge base on pokemon trading cards that I am providing as a sibling document to the one that you are now reading.

You should probably just copy and paste the pokemon code, look it over, and then make sure that it loads into Prolog. It works for me, so if it doesn't work for you, that is probably because of an "error in transmission" that you will need to sort out.

## Part 1: Queries

Please engage in a Prolog interaction to duplicate what I did in rendering the accompanying demo – before I replaced my queries with "query stubs". In other words, you are to recreate the demo by providing the 20 queries that I "redacted". The following "20 questions" indicate what you need to focus on for each query:

1. Query 1: Is picachu a "creatio ex nihilo" (created out of nothing) pokemon?
2. Query 2: Is raichu a "creatio ex nihilo" pokemon?
3. Query 3: By means of hand intervention, list all of the "creatio ex nihilo" pokemon.
4. Query 4: By means of the standard idiom of repetition, list all of the "creatio ex nihilo" pokemon.
5. Query 5: Does squirtle evolve into wartortle?
6. Query 6: Does wartortle evolve into squirtle?
7. Query 7: Does squirtle evolve into blastoise?
8. Query 8: By means of hand intervention, list all triples of pokemon such that the first evolves into the second and the second evolves into the third.

9. Query 9: By means of the standard idiom of repetition, list all pairs of pokemon such that the first evolves through an intermediary to the second - placing an arrow between each pair.
10. Query 10: By means of the standard idiom of repetition, list the names of all of the pokemon.
11. Query 11: By means of the standard idiom of repetition, list the names of all of the fire pokemon.
12. Query 12: By means of the standard idiom of repetition, provide a summary of each pokemon and its kind, representing each pairing of name and kind in the manner suggested by the redacted demo.
13. Query 13: What is the name of the pokemon with the waterfall attack?
14. Query 14: What is the name of the pokemon with the poison-powder attack?
15. Query 15: By means of the standard idiom of repetition, list the names of the attacks of all of the water pokemon.
16. Query 16: How much damage (hp count) can poliwhirl absorb?
17. Query 17: How much damage (hp count) can butterfree absorb?
18. Query 18: By means of the standard idiom of repetition, list the names of all of the pokemon that can absorb more than 85 units of damage.
19. Query 19: By means of the standard idiom of repetition, list the names of all of the pokemon that can dish out more than 60 units of damage with one instance of their attack.
20. Query 20: By means of the standard idiom of repetition, list the names and the hit point value for each of the "creation ex nihilo" pokemon, with the results formatted as the redacted demo suggests.

---

```
bash-3.2$ swipl
<<redacted>>

?- consult('pokemon_plus.pro').
% pokemon.pro compiled 0.00 sec, 54 clauses
true.

?- <<Query 1>>
true.

?- <<Query 2>>
false.

?- <<Query 3>>
Name = pikachu ;
Name = bulbasaur ;
Name = caterpie ;
Name = charmander ;
Name = vulpix ;
Name = poliwag ;
Name = squirtle ;
Name = staryu.

?- <<Query 4>>
pikachu
bulbasaur
caterpie
charmander
vulpix
```

```
poliwag
squirtle
staryu
false.

?- <<Query 5>>
true.

?- <<Query 6>>
false.

?- <<Query 7>>
false.

?- <<Query 8>>
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.

?- <<Query 9>>
bulbasaur --> venusaur
caterpie --> butterfree
charmander --> charizard
poliwag --> poliwrath
squirtle --> blastoise
false.

?- <<Query 10>>
pikachu
raichu
bulbasaur
ivysaur
venusaur
caterpie
metapod
butterfree
charmander
charmeleon
charizard
vulpix
ninetails
```

```
poliwag
poliwhirl
polywrath
squirtle
wartortle
blastoise
staryu
starmie
false.

?- <<Query 11>>
charmander
charmeleon
charizard
vulpix
ninetails
false.

?- <<Query 12>>
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(polywrath),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.

?- <<Query 13>>
N = wartortle

?- <<Query 14>>
N = venusaur

?- <<Query 15>>
water-gun
amnesia
dashing-punch
bubble
waterfall
```

```
hydro-pump
slap
star-freeze
false.

?- <<Query 16>>
HP = 80

?- <<Query 17>>
HP = 130

?- <<Query 18>>
raichu
venusaur
butterfree
charizard
ninetails
polywrath
blastoise
false.

?- <<Query 19>>
thunder-shock
poison-powder
whirlwind
royal-blaze
fire-blast
false.

?- <<Query 20>>
pikachu:  60
bulbasaur:  40
caterpie:  50
charmander:  50
vulpix:  60
poliwag:  60
squirtle:  40
staryu:  40
false.
```

## Part 2: Programs

Please extend the pokemon knowledge base in the `pokemon.pro` file by adding rules so that you can duplicate the accompanying demo. That is perform the following programming tasks in the context of the Pokemon KB:

1. Define the parameterless predicate called `display_names` to list the names of all of the pokemon represented in the KB.
2. Define the parameterless predicate called `display_attacks` to list the name of each pokemon's attack.
3. Define a predicate called `powerful` taking one parameter, the name of a pokemon, which succeeds only if the attack associated with the named pokemon yields with more than 55 units of damage.

4. Define a predicate called `tough` taking one parameter, the name of a pokemon, which succeeds only if the the named pokemon can absorb more than 100 units of damage (that is, has an hp count that is more than 100).
5. Define a predicate called `type` taking two parameters, the name of a pokemon, and the type of a pokemon, which succeeds only if the the named pokemon is of the specified type.
6. Define a predicate called `dump_kind` taking one parameter, the kind (type) of a pokemon, which displays all of the information for all of the pokemon of the specified type, doing so in a manner that is consistent with the representation of the pokemon in the KB.
7. Define a parameterless predicate called `display_cen` to display the names of all of the "creatio ex nihilo" pokemon.
8. Define a predicate called `family` taking one parameter, presumed to be a "creatio ex nihilo" pokemon, which displays the "evolutionary family" of the specified pokemon, all on a given line, as illustrated in the demo.
9. Define a parameterless predicate called `families` to display all of the evolutionary pokemon families, representing the families in the manner illustrated in the demo.
10. Define a predicate called `lineage` taking one parameter, the name of a pokemon, which displays all of the information for the pokemon and for each subsesquent pokemon in the evolutionary lineage of the pokemon. (Please see the demo for whatever clarification you might need.)

**Note: You should probably proceed by adding one predicate at a time, being sure to demo that predicate before moving on to do the next one.** After all of your predicates are written and working, then you should generate the complete demo (the one that I am requiring you to recreate) to place in your presentation document.

---

```
bash-3.2$ swipl
<<redacted>>

?- consult('pokemon_plus.pro').
% pokemon_plus.pro compiled 0.00 sec, 69 clauses
true.

?- display_names.
pikachu
raichu
bulbasaur
ivysaur
venusaur
caterpie
metapod
butterfree
charmander
charmeleon
charizard
vulpix
ninetails
poliwag
poliwhirl
polywrath
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
venusaur
butterfree
charizard
ninetails
wartortle
blastoise
false.

?- tough(raichu).
false.

?- tough(venusaur).
true

?- tough(Name), write(Name), nl, fail.
venusaur
butterfree
charizard
ninetails
```

```
polywrath
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
polywrath
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(polywrath),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
caterpie metapod butterfree
charmander charmeleon charizard
vulpix ninetails
poliwag poliwhirl poliwrath
squirtle 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(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.

?-
```

## Presentational Notes for Task 3

Place the following items within the "Task 3: Pokemon KB Interactions and Programming" section of your presentation document:

1. The demo for Part 1 of this task.
2. The Extended knowledge base (the `pokemon.pro` file) for Part 2 of this task.
3. Your very own Part 2 demo for this task, which mimics mine, thus assuring that your predicate definions are soundly written.

## Task 4: Lisp Processing in Prolog

1. Do what is asked in the "Head/Tail Referencing Exercises" section that is presented in Lesson 5 on list processing in Prolog.
2. Establish a file called `list_processors.pro` in which to place some list processing functions. Then, add to it definitions of the functions appearing in the "Example List Processors" section of Lesson 5. Then, mimic the demo associated with the example list processor functions presented in Lesson 5, being sure to save the demo for inclusion in your presentation document.
3. Please do what is asked in the "List Processing Exercises" section of Lesson 5, which involves defining some functions and performing a demo.

## Presentational Notes for Task 4

Place the following items within the "Task 4: List Processing in Prolog" section of your presentation document:

1. Your demo corresponding to the "Head/Tail Referencing Exercises".
2. The Prolog file containing all of the list processing function definitions that you were asked to write, those associated with the "Example List Processors" section from Lesson 5, and those associated with the "List Processing Exercises" from Lesson 5.
3. The demo associated with the "Example List Processors" that is provided in Lesson 5.
4. The demo that you are asked to create in the "List Processing Exercises" section of Lesson 5.

## Additional Notes on your Solution Document

Craft a nicely structured solution document that contains:

1. A nice title, indicating that this is your second Racket assignment.
2. A nice learning abstract, which foreshadows the tasks that you are asked to complete in this assignment.
3. A section for each of the 4 tasks, complete with the required code and demos.

Then post your document, in **pdf** format, to you web work site.

## Due Date

Please complete your work on this assignment, and post your work to your web work site no later than Friday, April 13, 2022.