Set of exercises 1

Getting started

First you have to install SWI-Prolog or Gnu Prolog on your computer.
Then, when you have written a Prolog program called for example `prog.pl`, while running the Prolog interpreter, you can load it running the goal `?- [prog].`
(Note that if it's a problem for you to use the extension `.pl` to your Prolog program because of the confusion that it may generate with Perl programs, you can use any extension you want, for example `.pro` but then you have to load it running the goal `?- ['prog.pro'].`)

Exercise 1

Consider the following facts and rules:

```
father(mike,mary).
father(mike,jane).
father(franz,camilla).
father(franz,agatha).
father(john,andy).

mother(abby,mary).
mother(abby,jane).
mother(mary,camilla).
mother(mary,agatha).
mother(jane,andy).

grand_father(X,Y) :- father(X,Z),father(Z,Y).
grand_father(X,Y) :- father(X,Z),mother(Z,Y).
```

Put those facts into a file called `family1.pl`
Launch your Prolog interpreter
When you see the command prompt, run the goal `?- trace.` to activate the debugger (you can run `?- notrace.` to desactivate the debugger)
Observe step by step, understanding what happens, the resolution of the following goals:

```
?- father(franz,camilla).
?- father(franz,axel).
?- father(franz,A).
?- grand_father(mike,agatha).
?- grand_father(mike,andy).
?- grand_father(mike,A).
?- grand_father(A,camilla).
?- grand_father(A,B).
```

Exercise 2

Convert to Horn clauses (a Prolog program) the following english sentences:

1. John, Jack and Mary are students
2. John and Jack are males
3. Mary is a female
4. Two individuals X and Y may fall in love if they are students and one is a female, the other is a male (I know this is not the only situation, but it's just to make the exercise simple)

love(X,Y) :- student(X), student(Y), male(X), female(Y); female(X), male(Y).

Once you have designed your Prolog program, load it into the knowledge base of the Prolog interpreter (using consult/1 or [filename]) and write the queries corresponding to the following english queries:

1. Is John a student?
2. Is John a female?
3. Can John love Mary?
4. Can John love Jack?
5. Who can love who?

Exercise 3

Here are some informations about a simple world:

1. If someone is young and plays the guitar then that person is happy.
2. If someone is old and plays the violin then that person is happy.
3. If someone plays the drums then he's happy.
4. Mary is young, owns a guitar, has learned the guitar and loves Paul.
5. John is old and he knows how to dance.
6. If a person owns an instrument and has learned to play it then that person plays that instrument.
7. If a person is a genius and a musical instrument is a chord instrument, then that person plays that instrument.

8. If a person has built a musical instrument then that person plays that instrument.
9. Lindsey has built a violin.
10. The guitar and violin are chord instruments.
11. George is a genius.
12. For any person P1 and P2,
    - If P1 is happy and loves P2, then P1 is a happy lover of P2.
    - If P1 knows how to dance and P2 is happy then P1 is a happy dancer with P2.
    - If P1 is a happy dancer with P2 then P1 dances with P2.
    - If P1 is a happy lover of P2 then P1 dances with P2.
    - If P1 has built and instrument, then P1 dances with this instrument.

Write a Prolog program and run a judicious query to help you answer the question: "Who dances with whom/what?"

Exercise 4

Convert to Horn clauses (a Prolog program) the following english sentences:

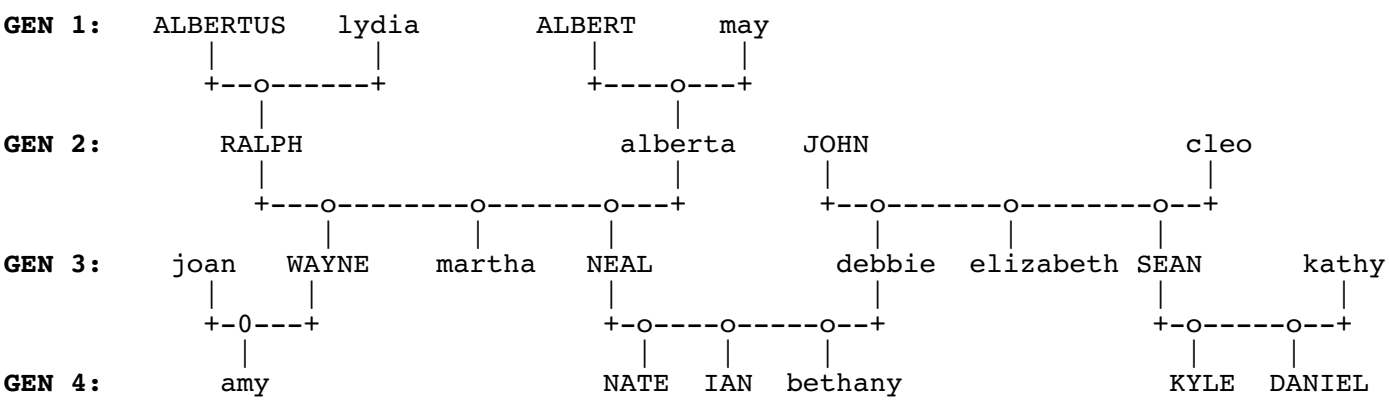
1. Dijon is a town located in Burgundy
2. Nice is a town located in Provence
3. Paris is a town located in Ile de France
4. London is a town located in Greater London
5. Newcastle is a town located in Avon
6. If a town is located in Burgundy, then it is located in France
7. If a town is located in Provence, then it is located in France
8. If a town is located in Ile de France, then it is located in France
9. If a town is located in Greater London, then it is located in Great Britain
10. If a town is located in Avon, then it is located in Great Britain
11. If a town is located in France, then it is located in Europe
12. If a town is located in Great Britain, then it is located in Europe

Once you have designed your Prolog program, load it in the knowledge base of Prolog interpreter (using consult/1 or [filename]) and write the queries corresponding to the following english queries:

1. Is Dijon located in France?
2. Is Nice located in Great Britain?
3. What are the towns located in France?
4. What are the towns located in Europe?
5. In what land, country and continent is located Dijon?

Exercise 5

Here is a family tree



This information is modeled in the Prolog file family2.pl that defines the predicates `male/1`, `female/1`, `parent/2`, `married/2`
Define the following predicates, and solve the appropriate goals to show they work:

```
spouse/2
husband/2
wife
father/2
mother/2
sibling/2
brother/2
sister/2
grandparent/2
grandfather/2
grandmother/2
grandchild/2
grandson/2
granddaughter/2
ancestor/2
child/2
son/2
daughter/2
descendent/2
```

```
auntoruncle/2
uncle/2
aunt/2
cousin/2
nieceornephew/2
nephew/2
niece/2
greatgrandparent/2
greatgrandfather/2
greatgrandmother/2
greatgrandchild/2
greatgrandson/2
greatgranddaughter/2
parentinlaw/2
fatherinlaw/2
motherinlaw/2
siblinginlaw/2
brotherinlaw/2
sisterinlaw/2
childinlaw/2
soninlaw/2
daughterinlaw/2
```

Exercise 6

1. The predefined predicate `append/3` is defined by: `append(L1,L2,L3)` is true if the list `L3` is the concatenation of the lists `L1` and `L2`. To be convinced that Prolog is not an imperative programming language, nor a functional programming language but a logic programming language, that is a language that defines relationships between "objects", run the following goals.

   ```
   ?- append([1,2,3],[4,5,6],[1,2,3,4,5,6]).
   ?- append([1,2,3],[4,5,6],L3).
   ?- append([1,2,3],L2,[1,2,3,4,5,6]).
   ?- append(L1,[4,5,6],[1,2,3,4,5,6]).
   ?- append(L1,L2,[1,2,3,4,5,6]).
   ?- append(L1,L2,L3).
   ```

2. Try to define a predicate `concat/3` defined by: `concat(L1,L2,L3)` is true if the list `L3` is the concatenation of the lists `L1` and `L2`. Of course you are not allowed to write `concat(L1,L2,L3) :- append(L1,L2,L3)`. In fact your definition should only contain one fact and one recursive rule.
3. When your predicate `concat/3` is correctly defined, take time to look at the resolution steps using the debugger on the following goals:

   ```
   ?- concat([1,2,3],[4,5,6],[1,2,3,4,5,6]).
   ?- concat([1,2,3],[4,5,6],L3).
   ?- concat([1,2,3],L2,[1,2,3,4,5,6]).
   ?- concat(L1,[4,5,6],[1,2,3,4,5,6]).
   ?- concat(L1,L2,[1,2,3,4,5,6]).
   ?- concat(L1,L2,L3).
   ```

Exercise 7

1. The built-in predicate `is/2` is used when we want to evaluate arithmetic expressions. You must not confuse it with the built-in predicate `=/2` which can be used to unify two terms. It is very important to understand the difference between the built-in predicate `=/2` and `is/2` so run the following goals and be sure to understand the difference:

   ```
   ?- X = 1+2.
   ?- =(X,1+2).
   ?- X is 1+2.
   ?- is(X,1+2).
   ```

   Note that we generally prefer the infix notation (X is 1+2 and X=1+2)
2. Write a predicate `mylength/2` such as `mylength(L,N)` is true if `N` is the length of the list `L`