Exercise 9.1

Which of the following queries succeed, and which fail?

```
?- 12 is 2*6                      success

?- 14 =\= 2*6                     success

?- 14 = 2*7                       fail

?- 14 == 2*7                      fail

?- 14 \== 2*7                     success

?- 14 =:= 2*7                     success

?- [1,2,3|[d,e]] == [1,2,3,d,e]   success

?- 2+3 == 3+2                     fail

?- 2+3 =:= 3+2                    success

?- 7-2 =\= 9-2                    success

?- p == 'p'                       success

?- p =\= 'p'                      fail, p/0 is not a function

?- vincent == VAR                 fail

?- vincent=VAR,VAR==vincent       success
```

Exercise 9.2

How does Prolog respond to the following queries?

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

?- .(a,.(b,.(c,[]))) = [a,b|[c]]             Yes

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

?- .(a,.(b,.(.(c,[]),[]))) = [a,b|[c]]       No
```

Exercise 9.3

Write a two-place `predicate termtype(+Term,?Type)` that takes a term and gives back the type(s) of that term (atom, number, constant, variable etc.). The types should be given back in the order of their generality. The predicate should, e.g., behave in the following way.

```
   ?- termtype(Vincent,variable).

   yes

   ?- termtype(mia,X).

   X = atom ;

   X = constant ;

   X = simple_term ;

   X = term ;

   no

   ?- termtype(dead(zed),X).

   X = complex_term ;

   X = term ;

   no
```

```
termtype(Term,atom) :- atom(Term).

termtype(Term,number) :- number(Term).

termtype(Term,constant) :- atomic(Term).

termtype(Term,variable) :- var(Term).

termtype(Term,simple_term) :- functor(Term,_,0).

termtype(Term,complex_term) :- functor(Term,_,X), X>0.

termtype(_,term).
```

Exercise 9.4

Write a program that defines the predicate `groundterm(+Term)` which tests whether `Term` is a ground term. Ground terms are terms that don't contain variables. Here are examples of how the predicate should behave:

```
   ?- groundterm(X).

   no

   ?- groundterm(french(bic_mac,le_bic_mac)).

   yes
   ?- groundterm(french(whopper,X)).
   no
groundterm(T) :- ground(T).
```

Exercise 9.5

Assume that we have the following operator definitions.

```
   :- op(300, xfx, [are, is_a]).
   :- op(300, fx, likes).
   :- op(200, xfy, and).
   :- op(100, fy, famous).
```

Which of the following is a well formed term? What is the main operator? Give the bracketing.

```
   ?- X is_a witch.
well formed, main operator is is_a
is_a(X, witch)
   ?- harry and ron and hermione are friends.
well formed, main operator is are
are(and(harry,and(ron,hermione),friends)
   ?- harry is_a wizard and likes quidditch.
not well formed, operator priority clash
   ?- dumbledore is_a famous famous wizard.
well formed, main operator is is_a
is_a(dumbledore,famous(famous(wizard)))
```