# 02156 Exercises-07


## Jørgen Villadsen


### 2021

**Exercise 1**

Use refutation and the systematic construction of a semantic tableau. State whether this shows that the following formulas are valid or not.

$p(a)$ $\quad \exists x p(x)$ $\quad \forall x p(x)$ $\quad \forall x p(x) \rightarrow \exists x p(x)$ $\quad \exists x p(x) \rightarrow \forall x p(x)$ $\quad \exists x((\neg p(a) \rightarrow (q \wedge \neg q)) \rightarrow p(x))$

**Exercise 2**

Explain the following tiny program: `p :- \+ !.`

**Exercise 3**

Recall the special Univ predicate with the following instantiation patterns:

```
+Term =.. ?List
-Term =.. +List
```

Here `List` is a list which head is the functor of `Term` and which tail is a list of the arguments of the term (it is an error if both `List` and `Term` are variables).

Often the following predicates are more efficient and/or appropriate:

`functor(+Term,?F,?N)` succeeds if and only if `Term` has functor `F` and arity `N` (number of arguments).

`arg(+N,+Term,?Arg)` succeeds if and only if `Arg` is argument number `N` of `Term` (counting starts at 1).

```
?- T = f(a,b), functor(T,F,N), arg(1,T,X), arg(2,T,Y).

T = f(a, b)
F = f
N = 2
X = a
Y = b

Yes
```

The instantiation pattern `functor(-Term,+F,+N)` is also possible and here `Term` is unified with a new term holding only variables, hence this functionality for `functor(Term,F,N)` can be obtained as follows in SWI-Prolog (where `List` is a new variable):

```
...   length(List,N), Term =.. [F|List],   ...
```

Ignore that particular instantiation pattern in the following.

Write programs `fun`, `arg0` and `arg1` corresponding to `functor` and `arg` where counting starts at 0 and 1, respectively (it is really simple).

Of course `functor` and `arg` must not be used, but the auto-loaded predicates `nth0` and `nth1` can be used:

`nth0(?Index,?List,?Elem)` succeeds when the `Index`-th element of `List` unifies with `Elem` (counting starts at 0).

`nth1(?Index,?List,?Elem)` succeeds when the `Index`-th element of `List` unifies with `Elem` (counting starts at 1).

In SWI-Prolog the instantiation pattern is not just `arg(+N,+Term,?Arg)` but even `arg(?N,+Term,?Arg)` — can you handle this as well?

Example:

```
?- arg0(N,p(2,3,5,7),X).

N = 0
X = 2 ;

N = 1
X = 3 ;

N = 2
X = 5 ;

N = 3
X = 7 ;

No
```

**Exercise 4**

Explain the differences between the following queries:

```
?- repeat, true -> true ; true.

?- repeat, (true -> true ; true).

?- repeat, (true -> true ; true), !.
```

**Exercise 5**

Explain the answers to the following queries.

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

?- \+ member(X,[a,b,c]).

?- \+ \+ member(X,[a,b,c]).

?- \+ \+ \+ member(X,[a,b,c]).

?- \+ \+ \+ \+ member(X,[a,b,c]).

?- \+ \+ \+ \+ \+ member(X,[a,b,c]).

?- \+ \+ \+ \+ \+ \+ member(X,[a,b,c]).
...
```

**Exercise 6**

Write a program `bs(+List1,?List2)` that succeeds if and only if `List2` is a permutation of `List1` and the elements in `List2` are ordered by the `=<` relation.

Use the Bubblesort algorithm, that is, swap adjacent elements that are not in the correct order until the list is sorted.

Try to write a recursive program that consists on just one clause, with one if-then-else construction, one use of `<` and one use of `=`, and with two calls to `append`.