1. *Solution.* A program P is called (i) *correct* with respect to the intended meaning M, if the meaning of P is a subset of M (ii) *complete* if the intended meaning M is a subset of the meaning of P. Furthermore the *meaning* of P is the set of ground facts deducible from P. In the concrete case of is_list/1 this is the set LL of all (untyped) lists. Clearly if holds that $M \subset L$, but not $L \subseteq M$; a counter-example for the latter would be the list $[a, b, b, a]$. Thus, it remains to correct the definition of the intended meaning as follows:

$$M' := \{[t_i|\cdot]^n([]) \mid n \geqslant 0, t_i \text{ an arbitrary ground term}\} .$$

□

2. *Solution.*

   c) No, for example subsequence([a],[X,Y]) yields that $X \mapsto a$.

□

3. *Solution.* See 2s.pl □

4. *Solution.*
   - foo(X,Y) holds if Y is reachable from X in a graph represented by the predicate *edge*/2. The graph is traversed breadth-first.
   - setof1(Template,Goal,Set) succeeds with the empty list, if no instance of *Template* can meet *Goal*. This is in contrast to the system predicate *setof*/3, which simply fails in this case. If *setof1*/3 is replaced by *setof*/3 in the considered program, then the breadth-first search fails. Let us call the new programm foo'. For example, if we define the following facts:

     ```
     edge(a,b).
     edge(a,c).
     ```

     we have that foo(a,c) holds (as it should), but foo'(a,c) fails.

□

5. *Solution.* See 2s.pl □

6. *Solution.*

statement	**yes**	**no**
In logic programming, terms are built from logical variables, constants and functions.	☐	☑
A computation of a goal G from a program P is the verification of an inference $P \vdash G$.	☑	☐
A type is an arbitrary, but finite set of terms.	☐	☑
We call a type complete, if it is closed under instantiation.	☑	☐
Difference lists are effective if independently different sections of a list are built, which are then concatenated.	☑	☐
Consider the standard implementation of *append*/3. Then any call to *append* terminates iff the second argument is a complete list.	☐	☑
A Prolog clause is called *tail recursive* iff it has one recursive call and zero or more calls to system predicates that appear before the recursive call.	☐	☑
A cut fixes all choices between (and including) the moment of matching the rule's head with parent goal and the cut. If backtracking should reaches the cut, then the cut succeeds and the execution is continued with the clause after the clause containing the cut.	☐	☑
Like almost any other programming language, answer set programming is Turing complete.	☐	☑
The predicate *bagof*($Template$,$Goal$,Bag) unifies Bag with the first alternative of $Goal$ that meets $Template$.	☐	☑

□