Second Exam
Logic Programming, LVA 703113

April 9, 2018

Name: **Student Number**:

The exam consists of 6 exercises with a total of 100 points. Please fill out your name and credentials *before* you start the exam.

1	2	3	4	5	6	Sum

0–49: **5**	50–59: **4**	60–74: **3**	75–89: **2**	90–102: **1**

1. Consider the following type definition of (untyped) lists:

```
is_list([]).
is_list([_|Xs]) :- is_list(Xs).
```

   – Prove that this definition is *complete* with respect to the set

$$M := \{[0|\cdot]^n([]) \mid n \geqslant 0\} = \{[], [0], [0,0], [0,0,0], \dots\} .$$

(10 pts)

   – Disprove that this definition is *correct* with respect to the set M. (10 pts)

   – Provide a definition of an intendend meaning M' of the type definition, such that the type definition is correct and complete for M' (10 pts)

2. Consider the predicate subsequence(Sub,Seq), which is true if *Sub* is a subsequence of *Seq*, that is, *Sub* is derived from *Seq* by deleting some elements without changing the order.

   a) Implement the predicate such that it works as intended on ground terms. (10 pts)

   b) Construct the Prolog search tree of the ground query subsequence([a,c],[a,b,c,d] (based on Prolog's selection function). (5 pts)

   c) Does the predicate work as intended (without side-effects) on arbitrary terms? Explain your answer. (5 pts)

3. Implement a predicate *duplicate*/3 that duplicates the elements of a list a given number of times. For example the query duplicate([a,b,c],2, Xs) should deliver the answer Xs = [a, a, b, b, c, c]. Use difference-lists in your implementation, where you can assume that \ separates difference lists. (10 pts)

4. Consider the following Prolog program.

```
foo(X,Y) :-
        foo([X|Xs]\Xs,Y,[X]).

foo([]\[],Y,Visited) :-
        !, fail.
foo([A|Xs]\Ys,A,Visited).
foo([A|Xs]\Ys,B,Visited) :-
        setof1(N,edge(A,N),Ns),
        foo2(Ns,Visited,Visited1,Xs\Ys,Xs1),
        foo(Xs1,B,Visited1).

foo2([N|Ns],Visited,Visited1,Xs,Xs1) :-
        member(N,Visited),
        foo2(Ns,Visited,Visited1,Xs,Xs1).
foo2([N|Ns],Visited,Visited1,Xs\[N|Ys],Xs1) :-
        \+ member(N,Visited),
        foo2(Ns,[N|Visited],Visited1,Xs\Ys,Xs1).
```

```
foo2([],V,V,Xs,Xs).

setof1(Template,Goal,Set) :-
          setof(Template,Goal,Set).
setof1(Template,Goal,Set) :-
          \+ setof(Template,Goal,Set), !, Set = [].
```

The meaning of the program changes if *setof1* /3 is replaced by the system predicate *setof* /3. Give an example of a goal that succeeds in the original program, but fails in the altered program. (10 pts)

5. Implement the n^{th} factorial as constraint logic program:

```
:- factorial(N,1).
N = 0;
N = 1;
false
```

(10 pts)

6. Determine whether the following statements are true or false. Every correct answer is worth 2 points, every wrong answer -1 points. (The worst that can happen is that you get zero points for this exercise.) (20 pts)

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*.	☐	☐