1. – First observe that append(Us,Vs,Xs) is linear in $|Us|$. Now, let $|Xs| = n$. Thus we need n recursive calls to reverse in the proof-tree. For each of these calls at most n calls to append are necessary. Hence the proof-tree contains at most $\mathsf{O}(|Xs|^2)$ nodes.

   – The following variant of *reverse*/2 is linear in length of the first list.

```
reverse(Xs,Ys) :-
    reverse(Xs,[],Ys).

reverse([],Ys,Ys).
reverse([X|Xs],Ys,Zs) :-
    reverse(Xs,[X|Ys],Zs).
```

2.

```
duplicate(Xs,N,Ys) :-
     duplicate2(Xs,N,Ys\[]).

duplicate2([],_N,Ys\Ys).
duplicate2([X|Xs],N,Ys0\Ys2) :-
     generate(X,N,Ys0\Ys1),
     duplicate2(Xs,N,Ys1\Ys2).

generate(_X,0,Ys\Ys).
generate(X,N,Ys0\Ys1) :-
     N > 0,
     N1 is N - 1,
     generate(X,N1,Ys0\[X|Ys1]).
```

3. – 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. 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.

4. We give the complete solution of the problem.

```
complete_knights_tour(N, Knights) :-
    knights(N, Knights), Knights = [X/Y|_],
    jump(N,X/Y,1/1).

knights(N, Knights) :-
    M is N*N-1,
    knights(N,M,[1/1],Knights).

knights(_,0,Knights,Knights).
knights(N,M,Visited,Knights) :-
    Visited = [X/Y|_],
    jump(N,X/Y,U/V),
    \+ memberchk(U/V,Visited),
    M1 is M-1,
    knights(N,M1,[U/V|Visited],Knights).

jump(N,A/B,C/D) :-
    jump_dist(X,Y),
    C is A+X, C > 0, C =< N,
    D is B+Y, D > 0, D =< N.

jump_dist(1,2).
jump_dist(2,1).
jump_dist(2,-1).
jump_dist(1,-2).
jump_dist(-1,-2).
jump_dist(-2,-1).
jump_dist(-2,1).
jump_dist(-1,2).
```

5.

statement	**yes**	**no**
A rule is a universally quantified logical formula of the form $A \leftarrow B_1, B_2, \ldots, B_n$, where A is a goal and for all $i = 1, \ldots, n$: B_i is a goal.	✓	
An SLD-refutation is a finite SLD-derivation ending in the goal to be proven.		✓
Logic programming is a declarative programming paradigm, that is, the computation of a function is made a first-class citizen.		✓
The declarative semantics of a program P is the minimal model of P.	✓	
The order of goals is irrelevant in the computation model of logic programming, but not the order of rules.	✓	
The order of goals and the order of rules is irrelevant in the computation model of Prolog.		✓
Prolog is a language without types and the main technique to manipulate data is unification.	✓	
Difference lists are ineffective if the generation of different sections of a list depend on each other.	✓	
A meta-interpreter in Prolog interprets Prolog terms on the Warren abstract machine.		✓
The predicate *bagof*(*Template*,*Goal*,*Bag*) unifies *Bag* with the alternatives of *Goal* that meet *Template*.		✓