Homework #1

Deadline: Tuesday, 9 September 2014, 12:00 noon

Question 1 (1 point)

Implement a Prolog predicate called `answer/1` that will correctly answer the following three questions about your personal TA: What is their first name? What is their favourite pancake topping? Where were they when Holland lost against Argentina during the World Cup 2014? Here is an example (for a fictional TA):

```
?- answer(name).    ?- answer(pancake).    ?- answer(world_cup).
Joost               Appelstroop            Apeldoorn
Yes                 Yes                    Yes
```

Question 2 (1 point)

Solve Exercise 2.1 from the Lecture Notes (analysing lists).

Question 3 (2 points)

This exercise is about numbers. You are used to representing, say, the number *twelve* using the decimal system, in which it is written as '`12`'. But you could also use the *unary* system, in which it can be written as '`111111111111`'. Next week we will learn how to work with numbers in the usual decimal system, but even today you actually already know everything you need to know to work with numbers in the unary system. Your task will be to implement some basic arithmetical operations for working with unary numbers. We will represent unary numbers as lists of `x`'s of the appropriate length. Thus, *five* would be `[x,x,x,x,x]`, *twelve* would be `[x,x,x,x,x,x,x,x,x,x,x,x]`, and *zero* would be `[]`. In the sequel, all numbers are understood to be such non-negative integers given in unary notation.

(a) The *successor* of a number is the number we obtain if we add *one* to it. Thus, for example, the successor of *five* is *six*. Write a predicate called `successor/2` that will return, in the second argument position, the successor of the number provided in the first argument position. Examples:

```
?- successor([x, x, x], Result).    ?- successor([], Result).
Result = [x, x, x, x]               Result = [x]
Yes                                 Yes
```

(b) Implement a predicate `plus/3` to compute the sum of two given numbers. Example:

```
?- plus([x, x], [x, x, x, x], Result).
Result = [x, x, x, x, x, x]
Yes
```

(c) Implement a predicate `times/3` to multiply two given numbers. Examples:

```
?- times([x, x], [x, x, x, x], Result).
Result = [x, x, x, x, x, x, x, x]
Yes

?- times([x, x, x], [x, x, x, x, x], Result), write(Result).
[x, x, x, x, x, x, x, x, x, x, x, x, x, x, x]
Result = [x, x, x, x, x, x, x, x, x|...]
Yes
```

Note that in the last example, the result (a list of fifteen `x`'s) is too long for Prolog to print, so we force printing using the `write`-command at the end of our query. Make sure your predicate works correctly also when one of the numbers is *zero*.

Hint: You don't need to use any "normal" numbers in your program and you should not use any arithmetic operations provided by Prolog (which we will only cover next week).

Question 4 (2 points)

Write a predicate `bulldoze/2` that will take a list that may contain several levels of sub-lists as its first argument and that will return a "bulldozed" version of that list, removing all the square brackets, in the second argument position. Examples:

```
?- bulldoze([[peter,mary],[paul,colette,[luke]],[bob]], Result).
Result = [peter, mary, paul, colette, luke, bob]
Yes

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

It is sufficient for your predicate to work for input lists that are ground terms.

Question 5 (1.5 points)

Solve Exercise 2.3 from the Lecture Notes (removing duplicates).

Question 6 (1.5 points)

Solve Exercise 2.4 from the Lecture Notes (reversing lists).

Question 7 (1 point)

Recall the big animal program from the first lecture, consisting of four facts and two rules. Change the order of the two subgoals of the second rule. What happens when you execute the following query, asking for all possible alternative solutions using the semicolon key? Briefly explain *why* this happens.

```
?- is_bigger(A, donkey).
```

Advice: Try to predict what will happen before trying it on the computer.