Homework 2
Types for programs and proofs
due 26 September 2013, 13.15

All exercises should be implemented in Agda.

1. Write the Ackermann function in Agda! You can find its definition in wikipedia. There are actually several variants, but it suffices if you implement one of them. Compute the result of the Ackermann function for some arguments! Which is the largest number you can output?

2. Define equality of natural numbers using the primitive recursion combinator `natrec`!

3. (a) Prove the law of the excluded middle for booleans:

$$(a : \mathrm{Bool}) \to a \vee \neg a$$

   (b) Prove one of de Morgan's laws for booleans :

$$(a\ b : \mathrm{Bool}) \to \neg(a \& b) \Leftrightarrow \neg a \vee \neg b$$

4. Define the append function which concatenates two lists! Prove that it is associative.

5. Write an Agda program which terminates for all inputs but which is not accepted by Agda's termination checker!

6. SASL (an early lazy functional programming language) had a general tautology function. It is a higher order function which takes a boolean function (a predicate) of arbitrary arity as argument and checks whether it is a tautology, that is, whether it is true everywhere. With dependent types we can give it the following type

   ```
   taut : (n : N) -> Pred n -> Bool
   ```

   where `Pred` n is the type of n-place predicates:

   ```
   Pred zero     = Bool
   Pred (succ n) = Bool -> Pred n
   ```

   Your task is to program `taut` so that `taut` n f = `true` iff f is a tautology.

7. Implement proofs of the following propositions in Agda

   (a) $\neg\neg\neg P \rightarrow \neg P$

   (b) $\neg(P \vee Q) \leftrightarrow \neg P \& \neg Q$.

   (c) $\neg(\exists x \in D)P(x) \leftrightarrow (\forall x \in D)\neg P(x)$

   You should of course use the propositions as types interpretation of Curry and Howard! See for example chapter 4 in "Dependent types at work!!