logical verification 2008-2009
exercises 2

Exercise 1. This exercise is concerned with dependent types. We use the following definition in Coq:

```
Inductive natlist_dep : nat -> Set :=
  | nil_dep : natlist_dep 0
  | cons_dep : forall n : nat,
               nat -> natlist_dep n -> natlist_dep (S n).
```

a. What is the type of `natlist_dep`?
What is the type of `natlist_dep 2`?
Describe the elements of `natlist_dep 2`.

b. Suppose we want to define a function `nth` that takes as input a list and gives back the nth element of that list. How can dependent lists be used to avoid errors?

Exercise 2. This exercise is concerned with dependent types.

a. Give the type of `append_dep`, the function that appends two dependent lists.

b. Give the type of `reverse_dep`, the function that reverses a dependent list.

c. Consider the following two terms:

```
reverse_dep (plus n1 n2) (append_dep n1 n2 l1 l2)
append_dep n2 n1 (reverse_dep n2 l2) (reverse_dep n1 l1)
```

(Here `n1` and `n2` have type `nat`, the term `l1` has type `natlist_dep n1`, the term `l2` has type `natlist_dep n2`.)

What are the types of the above terms?
Are the types convertible?

Exercise 3. This exercise is concerned with λ-calculus with dependent types (λP).

a. A typing rule that is characteristic for λP is the following:

$$\frac{\Gamma \vdash A : * \qquad \Gamma, x : A \vdash B : \Box}{\Gamma \vdash \Pi x{:}A.\, B : \Box}$$

Explain how this rule is used to infer that the type of `natlist_dep` is ok.

b. Another typing rule that is characteristic for λP is the conversion rule:

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s}{\Gamma \vdash A : B'} \qquad \text{with } B =_\beta B'$$

Explain with an example (for instance `natlist_dep`) how the conversion rule can be used.

Exercise 4.

a. Give an inhabitant of $(\Pi x{:}\mathsf{Terms}.\, P\, x) \to (P\, M)$.

b. Give an inhabitant of $(\Pi x{:}\mathsf{Terms}.\, P\, x \to Q\, x) \to (\Pi x{:}\mathsf{Terms}.\, P\, x) \to (\Pi y{:}\mathsf{Terms}.\, Q\, x)$.

Exercise 5. This exercise is concerned with the Curry-Howard-De Bruijn isomorphism between first-order predicate logic and λP.

a. Give the encoding of algebraic terms (from predicate logic) in λP.

b. Give the encoding of formulas from predicate logic in λP.

c. How are the introduction rules (for $\to$ and $\forall$) from predicate logic represented in λP?

d. How are the elimination rules (for $\to$ and $\forall$) from predicate logic represented in λP?

Exercise 6. First-order propositional logic can be encoded in Coq using dependent types as follows:

```
(* prop representing the propositions is a Set *)
Variable prop:Set.
(* implication on prop is a binary operator *)
Variable imp: prop -> prop -> prop.
(* T expresses if a proposion in prop is valid
   if (T p) is inhabited then p is valid
   if (T p) is not inhabited then p is not valid *)
Variable T: prop -> Prop.
```

Give the types of the variables `imp_introduction` and `imp_elimination` modelling the introduction- and elimination rule of implication.

Exercise 7. This exercise is concerned with polymorphic lambda-calculus and second-order minimal propositional logic.

a. What is the type of the polymorphic identity?

b. Show how the polymorphic identity is used to get the identity on the type `nat` of natural numbers.

c. Give the polymorphic version of the following function:
$\lambda f{:}\mathsf{nat} \to \mathsf{bool} \to \mathsf{nat}.\, \lambda x{:}\mathsf{nat}.\, \lambda y{:}\mathsf{bool}.\, f\, x\, y.$

(In the polymorphic variant neither nat nor bool occurs.)

d. Explain why the following proof is not correct:

$$\dfrac{\exists a.\, a \to b \qquad \dfrac{[a \to b^x]}{(a \to b) \to (a \to b)}\, I[x]\to}{a \to b}\, E\exists$$

Exercise 8. This exercise is concerned with second-order propositional logic and polymorphic λ-calculus ($\lambda 2$).

a. Show that $\forall a.\, ((\forall b.\, b) \to a)$ is a tautology.

b. Give the $\lambda 2$-term corresponding to the formula $\forall a.\, ((\forall b.\, b) \to a)$.

c. Give a $\lambda 2$-term that is an inhabitant of the answer to *8b*.

Exercise 9. This exercise is concerned with second-order minimal propositionla logic and polymorphic λ-calculus.

a. Show that $(\forall c.\, ((a \to b \to c) \to c)) \to a$ is a tautology of second-order minimal propositional logic.

Exercise 10.

a. What is the impredicative definition of $\bot$ in second-order propositional logic?

b. What is the corresponding term in $\lambda 2$?

Exercise 11. This exercise is concerned with the encoding of logic and datatypes in polymorphic λ-calculus ($\lambda 2$).

a. Define the type new_or

$$(\mathsf{new\_or}\, A\, B) = \Pi c{:}*\, .\, (A \to c) \to (B \to c) \to c$$

Assume $\Gamma \vdash a : A$. Give an inhabitant of $(\mathsf{new\_or}\, A\, B)$.

(NB: it is not asked to give the type derivation.)

b. Assume new_or as in a, and in addition $\Gamma \vdash f : A \to D$, and $\Gamma \vdash g : B \to D$, and $\Gamma \vdash M : (\mathsf{new\_or}\, A\, B)$. Give an inhabitant of D.

(NB: it is not asked to give the type derivation.)

c. We define the booleans B and *true* (T) and *false* (F) as follows:

$\mathsf{B} = \Pi a{:}*\, .\, a \to a \to a$
$\mathsf{T} = \lambda a{:}*\, .\, \lambda x{:}a.\, \lambda y{:}a.\, x$
$\mathsf{F} = \lambda a{:}*\, .\, \lambda x{:}a.\, \lambda y{:}a.\, y$

Give a definition of negation in $\lambda 2$.

Exercise 12. We assume $a : \star$. Give inhabitants in $\lambda 2$ of the following types:

a. $(\Pi b : \star.\, b) \to a$,

b. $a \to \Pi b : \star.\, (b \to a)$,

c. $a \to \Pi b : \star.\, ((a \to b) \to b)$.