Assignment 10
Introduction to Computational Logic, SS 2011

Prof. Dr. Gert Smolka, Dr. Chad Brown
www.ps.uni-saarland.de/courses/cl-ss11/

Read in the lecture notes: Chapters 5-6

Note: It is very important to do all the examples in the lecture notes and the exercises below in the system Coq.

Exercise 10.1 Prove the following lemmas.

```
Lemma ex5811 n : even n -> even (4+n).
Lemma ex5812 n : even (S (S n)) -> even n.
```

Exercise 10.2 Prove the following lemmas.

```
Lemma ex58221 : ~ even 3.
Lemma ex5822 n : even (4+n) -> even n.
```

Exercise 10.3 Prove the following lemmas. Do not use induction on *nat*; only use induction on proofs of propositions of the form *even s*.

```
Lemma even_sum m n : even m -> even n -> even (m+n).
Lemma even_sum' m n : even (m+n) -> even m -> even n.
```

Exercise 10.4 Prove that the inductive and the boolean definitions of evenness are equivalent.

```
Lemma evenib n : even n <-> evenb n.
```

Exercise 10.5 Here is an impredicative definition of evenness.

```
Definition evenp (n : nat) : Prop :=
forall p : nat -> Prop,
p 0 -> (forall n, p n -> p (S (S n))) -> p n.
```

Prove that the inductive and the impredicative definitions of evenness are equivalent.

```
Lemma evenip n : even n <-> evenp n.
```

Exercise 10.6 Consider the following inductive definition of equality where the type constructor *eq2* takes two proper arguments.

```
Inductive eq2 (X : Type) : X -> X-> Prop :=
| eq2_I : forall x : X, eq2 X x x.
```

a) Give the typing rule for matches at *eq2*.

b) Prove the following property of *eq2*.

```
Lemma eq2_E (X : Type) (x y : X) :
eq2 X x y -> forall p : X -> Prop, p y -> p x.
```

Exercise 10.7 Consider the following inductive definition of an order predicate for the natural numbers.

```
Inductive lei : nat -> nat -> Prop :=
| leiO : forall x : nat, lei O x
| leiS : forall x y, lei x y -> lei (S x) (S y).
```

a) Given the typing rule for matches at *lei*.

b) Prove the following lemmas.

```
Lemma lei_refl x : lei x x.
Lemma lei_trans x y z : lei x y -> lei y z -> lei x z.
Lemma leib x y : leb x y <-> lei x y.
```

Exercise 10.8 Use proof scripts to give inhabitants of the following two types.

a) $\forall p : Prop. p \vee p \to p + p$

b) $\forall p : Prop. p \vee False \to p + False$

Exercise 10.9 Complete the following definition and prove the lemma.

```
Definition forget {X Y} {p : X -> Prop} {q : Y -> Prop} : sig p + sig q -> X + Y.
Lemma forget_div2c (n : nat) : forget (div2c n) = divmod2 n.
```

Exercise 10.10 Recall the definition of the type *Search* from the lecture.

```
Definition Search : Type :=
forall (p : nat -> bool) (n : nat),
{x | x <= n /\ p x} + (forall x, x <= n -> ~ p x).
```

Define a similar type *SearchMax* such that any function of type *SearchMax* will return the maximum $x \leq n$ such that $p\, x$ if such an x exists, or a proof that no such x exists. Construct a certifying function of this type.