Short Reference — lesson 2

Basic commands

`Require Import Bool Arith List.`
Load libraries `Bool`, `Arith` and `List`

`Definition name (arg : type):= body.`
Defines a new constant called name.

`Lemma name : statement.`
State a lemma

`Proof.`
Start the proof of a lemma

`Qed.`
Terminate the proof of a lemma

`SearchAbout (_ + _ = _ + _).`
Search lemmas by pattern.

`About lem.`
Print the statement of lemma `lem`

Syntax for the statements

Mathematics	Coq
$\neg p$	`~ p`
$p \wedge q$	`p /\ q`
$p \vee q$	`p \/ q`
$a = b$	`a = b`
$a \neq b$	`a <> b`
$p \Rightarrow q$	`p -> q`
$\forall x \in A . p(x)$	`forall (x : A), p x`
$\exists x \in A . p(x)$	`exists (x : A), p x`

Example of a lemma

```
Lemma good_name :
 forall (n m : nat), n - m + m =
     n.
Proof.
(* your proof *)
Qed.
```

Basic proof commands

`reflexivity.`
Prove an equational goal by trivial means, or fail

```
=========
 2 + 1 = 3          →
```

`exact p.`
Prove a goal by using an assumption, or fail

```
 x : nat
 p : x = 0          →
=========
 x = 0
```

`intros x px.`
Introduce `x` and `P x` naming them `x` and `px`

```
                        x : T
=========               px : P x
 forall x,         →   =========
  P x -> Q x -> G       Q x -> G
```

`apply H.`
Apply `H` to the current goal

```
 H : A -> B             =========
=========          →    A
 B
```

`destruct ab as [a b].`
Eliminate the conjunction

```
                        a : A
 ab : A /\ B            b : B
=========          →   =========
 G                      G
```

`destruct ab as [a | b].`
Eliminate the disjunction

```
 ab : A \/ B      a : A          b : B
=========    →   =========      =========
 G                G              G
```

`destruct exg3 as [n ngt3].`
Eliminate the existential quantification

```
 exg3 : exists n, 3 <        n : nat
    n                   →    ngt3 : 3 < n
=========                   =========
 G                           G
```

`exists 7.`
Prove an existential statement

```
=========                         =========
 exists n, n = 3 + 4       →      7 = 3 + 4
```

`split.`
Prove a conjunction

```
=========      →   =========      =========
 A /\ B             A              B
```

`left.`
Prove a disjunction choosing the left part.
`right` chooses the right part

```
=========             →   =========
 A \/ B                    A
```

`rewrite -> Eab.`
Rewrite with `Eab` left to right

```
 Eab : a = b               Eab : a = b
=========            →    =========
 P a                       P b
```

`rewrite <- Eab.`
Rewrite with `Eab` right to left

```
 Eab : a = b               Eab : a = b
=========            →    =========
 P b                       P a
```

`unfold name.`
If name is a `Definition` then it replaces name by its body.

Good practices

- Choose meaningful names for intros, not H1, H2.
- Structure the proof using indentation.