Chapter 1

Library projet

```
Include Coq.Init.Nat.

Variable A : Type.

Fixpoint repeat (x:A) n := match n with
  | 0 ⇒ nil
  | S k ⇒ cons x (repeat x k)
  end.

Theorem repeat_length : ∀ x n,
  length (repeat x n) = n.

Inductive mem : A → list A → Prop :=
| mem_cons : ∀ x l, mem x (cons x l)
| mem_tail : ∀ x y l, mem x l → mem x (cons y l).

Theorem repeat_spec : ∀ n x y,
  mem y (repeat x n) → y=x.

Fixpoint alternate (x:A) (y:A) n := match n with
  | 0 ⇒ nil
  | S k ⇒ cons x (cons y (alternate x y k))
  end.

Lemma plus_n_Sm : ∀ n m,
  n + S m = S (n + m).

Theorem alternate_length : ∀ x y n,
  length (alternate x y n) = 2*n.

Theorem alternate_spec : ∀ n x y z,
  mem z (alternate x y n) → z=x ∨ z=y.

(* **************************************************

                         Exercice 2
```

```
   *****************************************************)
Variable E : Set.
Variable R : E → E → Prop.
Hypothesis R_refl : ∀ (x : E), R x x.
Hypothesis R_antisym : ∀ (x y : E), R x y → R y x → x=y.
Hypothesis R_trans : ∀ (x y z : E), R x y → R y z → R x z.
Definition smallest (x0 : E) := ∀ x : E, R x0 x.
Definition minimal (x0 : E) := ∀ x : E, R x x0 → x=x0.
Theorem q1 : ∀ (x y :E),
  (smallest x) ∧ (smallest y) → x=y .
Theorem q2 : ∀ (x : E),
  smallest x → minimal x.
Theorem q3 : ∀ (x y : E),
  smallest x ∧ minimal y → x=y.
(* ***************************************************

                    Exercice 3

   *****************************************************)
Variables T U : Type.
Hypothesis T_eq_dec : ∀ (x y : T), {x=y} + {~x=y}.
(* *** Partie 1 ****)
(* *** Question 1 ****)
Definition empty1 := nil : list (T×U).
Fixpoint find1 (l : list (T×U)) (k : T) := match l with
  | nil ⇒ None
  | cons (u,v) l2 ⇒ match T_eq_dec u k with
    | (left _) ⇒ Some v
    | (right _) ⇒ find1 l2 k
  end
end.
Definition add1 (l : list (T×U)) (k : T) (i:U) := cons (k,i) l.
Theorem prop1_assoc : ∀ x, find1 empty1 x = None.
Theorem prop2_assoc : ∀ t x v, find1 (add1 t x v) x = Some v.
Theorem prop3_assoc : ∀ t x y v, x ≠ y → find1 (add1 t x v) y = find1 t y.
(* *** Question 2 ****)
Definition empty2 := nil : list (T×U).
```

```
Fixpoint find2 (l : list (T×U)) (k : T) := match l with
  | nil ⇒ None
  | cons (u,v) l2 ⇒ match T_eq_dec u k with
    | (left _) ⇒ Some v
    | (right _) ⇒ find2 l2 k
  end
end.

Fixpoint change_value (l : list (T×U)) (k : T) (i:U) := match l with
  | nil ⇒ nil
  | cons (u,v) l2 ⇒ match T_eq_dec u k with
    | (left _) ⇒ (u,i)::l2
    | (right _) ⇒ (u,v)::(change_value l2 k i)
  end
end.

Definition add2 (l : list (T×U)) (k : T) (i:U) := match find2 l k with
  | None ⇒ (k,i)::l
  | _ ⇒ change_value l k i
end.

Inductive wf : list (T×U) → Prop :=
  | wf_head : wf empty2
  | wf_tail : ∀ (l : list (T×U)) (k : T) (i:U),
                wf l → wf (add2 l k i).

Lemma empty2_is_wf : wf empty2.

Lemma add2_is_wf : ∀ (l : list (T×U)) (k : T) (i:U),
          wf l → wf (add2 l k i).

Theorem prop1_assoc_q2 : ∀ x, find2 empty2 x = None.

Theorem prop2_assoc_q2 : ∀ t x v, find2 (add2 t x v) x = Some v.
Theorem prop3_assoc_q2 :
  ∀ t x y v, x ≠ y → find1 (add1 t x v) y = find1 t y.
Theorem prop1_assoc :
  ∀ x, find1 empty1 x = None.
Theorem prop2_assoc :
  ∀ t x v, find1 (add1 t x v) x = Some v.
Theorem prop3_assoc :
  ∀ t x y v, x ≠ y → find1 (add1 t x v) y = find1 t y.
```