Chapter 1

Library AVLTree

```
Require Import
  FSetInterface
  FMapInterface
  FMapList
  ZArith
  Int.

Import Z_as_Int.

Set Implicit Arguments.

Notation "s #1" := (fst s) (at level 9, format "s '#1'") : pair_scope.
Notation "s #2" := (snd s) (at level 9, format "s '#2'") : pair_scope.

Definition int := Z.

Definition key := int.

Inductive avltree :=
  | Leaf : avltree
  | Node : key → avltree → avltree → int → avltree.

Definition height (t:avltree) : int :=
  match t with
    | Leaf ⇒ 0%Z
    | Node _ _ _ h ⇒ h
  end.

Definition create k l r :=
  Node k l r (plus (max (height l) (height r)) 1).

Definition assert_false := create.

Fixpoint balance k l r :=
  let hl := height l in
  let hr := height r in
  if gt_le_dec hl (hr+2) then
```

```
      match l with
      | Leaf ⇒ assert_false k l r
      | Node lk ll lr _ ⇒
        if ge_lt_dec (height ll) (height lr) then
          create lk ll (create k lr r)
          else
            match lr with
            | Leaf ⇒ assert_false k l r
            | Node lrk lrl lrr _ ⇒
                create lrk (create lk ll lrl) (create k lrr r)
            end
      end
  else
    if gt_le_dec hr (hl+2) then
      match r with
        | Leaf ⇒ assert_false k l r
        | Node rk rl rr _ ⇒
          if ge_lt_dec (height rr) (height rl) then
            create rk (create k l rl) rr
          else
            match rl with
              | Leaf ⇒ assert_false k l r
              | Node rlk rll rlr _ ⇒
                  create rlk (create k l rll) (create rk rlr rr)
            end
      end
  else
    create k l r.

Fixpoint insert (k:key) (t:avltree) :=
  match t with
    | Leaf ⇒ Node k Leaf Leaf _1
    | Node k' l r h ⇒
        match (k ?= k')%Z with
          | Lt ⇒ balance k' (insert k l) r
          | Eq ⇒ Node k' l r h
          | Gt ⇒ balance k' l (insert k r)
        end
  end.

Definition x := insert _1 Leaf.
Definition y := insert _2 x.
```