```
theory Example_Lomuto
  imports Hnr_Diff_Arr Hnr_Array  Definition_Utils "HOL-Library.Multiset" "HOL-Library.Rew
begin

definition swap :: "nat ⇒ nat ⇒ 'a list ⇒ 'a list" where
  "swap i j xs ≡ (xs[i := xs!j])[j := xs!i]"

fun partition :: "nat ⇒ nat ⇒ ('a::linorder) list ⇒ ('a list * nat)" where
  "partition i j xs = (if 1 < j then
      (if xs ! 0 < xs ! (j - 1)
       then partition (i - 1) (j - 1) (swap (i - 1) (j - 1) xs)
       else partition i (j - 1) xs)
    else (swap (i - 1) 0 xs, i - 1)
  )"

abbreviation partition' where
  "partition' xs ≡ partition (length xs) (length xs) xs"

definition inv :: "nat ⇒ nat ⇒ ('a::linorder) list ⇒ bool" where
  "inv i j xs ≡
    let p = xs ! 0 in
    0 < length xs ∧
    0 < i ∧
    i ≤ length xs ∧
    j ≤ length xs ∧
    j ≤ i ∧
   (∀h ∈ set (drop i xs). p < h) ∧
   (∀l ∈ set (take (i - j) (drop j xs)). l ≤ p)"

definition is_valid_partition where
  "is_valid_partition ys m ≡ ∀l ∈ set (take m ys). ∀h ∈ set (drop m ys). l ≤ h"

lemma partition_correct_elements: "partition i j xs = (ys, m) ⟹ set xs = set ys"

lemma partition: "inv i j xs ⟹ partition i j xs = (ys, m) ⟹ is_valid_partition ys m"

lemma partition':
  "partition' (p#xs) = (ys, m) ⟹ is_valid_partition ys m"

end
```