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



definition swap_opt :: "nat ⇒ nat ⇒ 'a list ⇒ 'a list option" where
  "swap_opt i j xs = do {
     let c1 = xs!j;
     let c2 = xs!i;
     let c3 = xs[i := c1];
     let c4 = c3[j := c2];
     Some c4
  }"

lemma swap_opt_termination: "swap_opt i j xs = Some (swap i j xs)"

synth_definition swap_impl is [hnr_rule_diff_arr]:
  "hnr (master_assn' (insert (xs, xsi) F) * id_assn i ii * id_assn j ji)(⌑:: ?'a Heap) ?Γ'
  unfolding swap_opt_def
  by hnr_diff_arr

definition partition_opt :: "nat × nat × ('a::linorder) list ⇒ (('a::linorder) list × na




lemma partition_opt_termination: "partition_opt (i, j, xs) = Some (partition i j xs)"




synth_definition partition_impl is [hnr_rule_diff_arr]:
  "hnr
    (master_assn' (insert (xs, xsi) F) * id_assn i ii * id_assn j ji)
    (⌑:: ?'a Heap)
    ?Γ'
    (partition_opt (i, j, xs))"
  unfolding partition_opt_def
  apply(hnr_recursion
           "(λF p pi.
                 master_assn' (insert (snd(snd p), snd (snd pi)) F) *
                 id_assn (fst p) (fst pi) *
                 id_assn (fst (snd p)) (fst (snd pi)))"
           "(λF p pi r ri.
                 master_assn' (insert (snd(snd p), snd (snd pi)) (insert (fst r, fst ri) F)
                 id_assn (snd r) (snd ri) *
                 id_assn (fst p) (fst pi) *
                 id_assn (fst (snd p)) (fst (snd pi)) *
                 true
                 )"
           hnr_diff_arr_match_atom
         )
  by hnr_diff_arr



end
```