```
type ('k, 'v) tree =
  | Empty
  | Node of ('k, 'v) tree * 'k * 'v * ('k, 'v) tree

type ('k,'v) res = Done of ('k,'v) tree
  | ReqNF of 'k * ('v,('k,'v) res) subcont

let rec update4 : ('k,'v) res prompt ->
  'k -> ('v->'v) -> ('k,'v) tree -> ('k,'v) tree =
  fun pnf k f ->
    let rec loop = function
    | Empty -> Node(Empty,k,
                take_subcont pnf (fun c () -> ReqNF (k,c)),Empty)
    ...
    | Node (l,k1,v1,r) ->
        begin
          match compare k k1 with
          | 0 -> Node(l,k1,f v1,r)
          | n when n < 0 -> Node(loop l,k1,v1,r)
          | _            -> Node(l,k1,v1,loop r)
        end
    in loop
```