Esercizio 1)

```
let lenseq l =
    let f x (l, s, max) =
        match l with
        [] -> ([x], 1, 1)
      | y::ys -> if x=y then
                   if s+1 > max
                   then ([x], s+1, s+1)
                   else ([x], s+1, max)
                 else ([x], 1, max)
    in let (l, s, max) = foldr f ([], 0, 0) l
    in max ;;
```

Esercizio 2)

```
let rec member el l =
        match l with
          [] -> false
        | x::xs -> if el = x then true
                   else member el xs;;

let split l =
    let rec split_aux l =
        match l with
          [] -> ([], [])
        | x::xs -> if x = ø then ([], x::xs)
                   else let (l1, l2) = split_aux xs
                   in (x::l1, l2)

    in
      if not member ø l
        then ([], l)
        else split_aux l;;
```

Esercizio 3)

```
let rec conta el l =
    match l with
      [] -> 0
    | x :: xs -> if x = el
                 then 1 + conta el xs
                 else conta el xs ;;

let two l =
    let rec two_ans l1 l2 =
        match l1 with
          [] -> []
        | x :: xs -> if conta x l2 = 2
                     then x :: two_ans xs l2
                     else two_ans xs l2
    in two_ans l l ;;
```