Esercizio 1)

```
let split l x =
    let f y (l1, l2, b) =
        if b then (y :: l1, l2, b)
            else if y = x then (l1, y :: l2, true)
                else (l1, y :: l2, b)
    in let (a, b, c) = foldr f ([], [], false) l
    in (a, b);;
```

Esercizio 2)

Possibile soluzione 1

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

let rec precist l x = match l with
    [] -> []
  | y::ys -> if x = y then []
             else y :: precist ys x ;;

let prec l x =
    let l1 = precist l x
    in ( l1, contepos l1) ;;
```

{ sappiamo che x compare in l }

Esercizio 2)

Possibile soluzione 2

```
let rec l x =
  let rec preca l x (l1,n) = match l with
    y::ys -> if y=x then (l1,n)
             else if y>0 then preca ys (l1@[y],n+1)
                  else preca ys (l1@[y],n)
  in preca l x ([],0);;
```

{usiamo solamente questo pattern perché sappiamo che x occorre in l}

Esercizio 2

Possibile soluzione 3

```
let rec prec l x = match l with
```

Usiamo solamente questo pattern perché sappiamo che x occorre in l

```
y::ys -> if x = y then ([], 0)
         else let (l1, n) = prec ys x
              in if y > 0 then (y::l1, n+1)
                 else (y::l1, n);;
```

Esercizio 3)

```
let rec subst bt x n = match bt with
                Void -> Void
  | Node (y, lbt, rbt) ->
        if n=1 then Node (x, lbt, rbt)
          else Node (y, subst lbt x (n-1),
                        subst rbt x (n-1)) ;;
```

{ sappiamo che $n > 0$ }