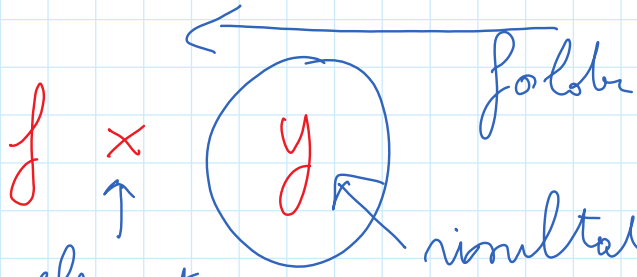
Cancellare, mediante foldr, gli elementi che seguono l'ultima occorrenza del valore $\emptyset$. Se non ci sono elementi $= \emptyset$ la lista va cancellata completamente.

$[3; \emptyset; 5; \emptyset; 7; 8] \rightarrow [3; \emptyset; 5; \emptyset]$



$$f\ x_1\ (f\ x_2\ \ldots\ (f\ x_n\ a)\ \ldots))$$

$y = (l, b)$

liste degli elementi risultanti

ho incontrato uno $\emptyset$ oppure no

```
let cancØ l =
  let f x (l,b) =
    if b then (x::l, b)
    else if x<>Ø then (l, b)
    else (x::l, true)
  in
  let (l,b) = polobr f ([], false) l
  in l ;;
```

false

(*)

Stesso esercizio, se la lista non contiene $\varnothing$ il risultato è la lista stessa

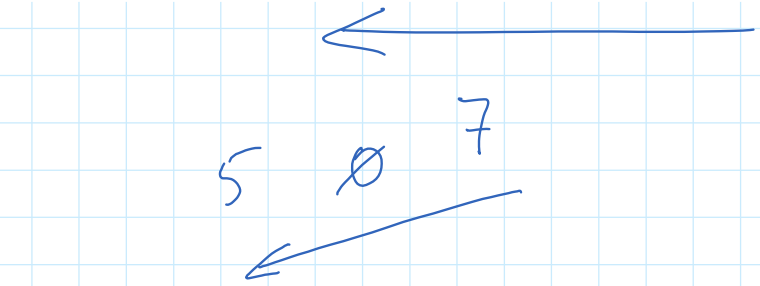
$[3;\varnothing;4;5;\varnothing;7] \longrightarrow [3;\varnothing;4;5;\varnothing]$

$[3;4;5] \longrightarrow [3;4;5]$

```
let canc∅ l =
    let f x (l, b) =
        if b then (x::l, b)
        else if x <> ∅ then (x::l, b)
             else ([x], true)
    in let (l1, b) = foldr (f, false) l
    in l1;;
```

true

$[3;4;5;\varnothing;7]$



$f \times g$

```
let conc∅ l =
    let f x (l, b) =
        if b then (x :: l, b)
        else if x <> ∅ then (l, b)
        else (x :: l, true)
    in
    if member ∅ l
    then let (ls, b) = fold_right ....
    else l ;;
```

Si definisce

inizio : int list → int → int list

che, date una lista di interi >= 0 e un intero n, restituisce il massimo prefisso (la più lunga porzione iniziale della lista) in cui la somma degli elementi è strettamente minore di n.

inizio [1;2;3;4;5] 7 = [1;2;3]

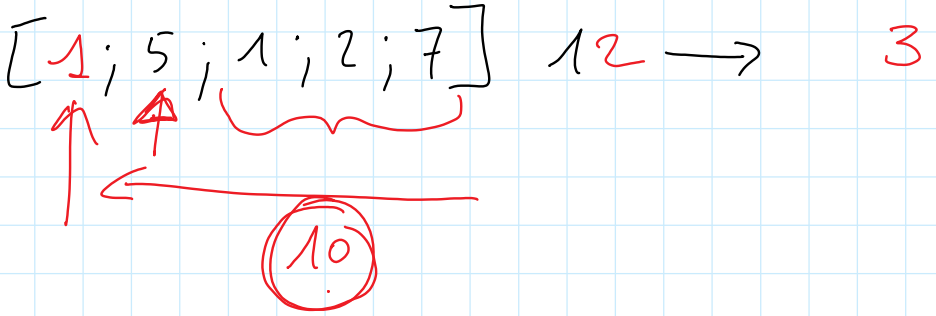
```
let inizio l n =
  let rec ina l n s = match l with
      [] -> []
    | x::xs when s+x >= n -> []
    | x::xs when s+x < n ->
        x:: ina xs n (s+x)
  in ina l n 0 ;;
```

```
let rec inizio l n = match l with
    [] -> []
  | x::xs when n-x <=0 -> []
  | x::xs when n-x >0 ->
        x:: inizio xs (n-x) ;;
```

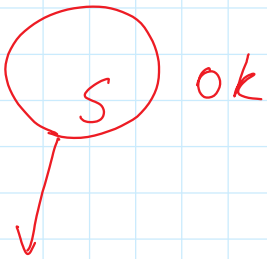
Si definisca, senza usare la ricorsione esplicita (usando foldr), una funzione

`num : int list → int → int`

che, date una lista di interi ≥ 0 e un intero n, calcola la lunghezza del massimo suffisso in cui la somma degli elementi è strettamente minore di n.



```
let num l n =
    let f x (l, s) =
        if s >= n then (l, s+x)
        else if s+x >= n then (l, s+x)
            else (x :: l, s+x)
    in let (l1, s1) = foldr f ([], 0) l
    in l1 ;;
```



```
let num l n =
    let f x (m, s) =
        if s >= n then (m, s+x)
        else if s+x >= n then (m, s+x)
             else (1+m, s+x)
    in let (n1, s1) = fold_left f (0, 0) l
    in n1 ;;
```

Si definisca una funzione ricorsiva
split : int list → int list * int list
che data una lista ls di interi restituisca la coppia (l1, l2) tale che:

- l1 @ l2 = ls
- l1 e l2 sono possibilmente vuote
- l1 è la sottolista INIZIALE più lunga possibile i cui elementi sono in ordine strettamente crescente

[3;4;5;2;7;8;5] → ([3;4;5], [2;7;8;5])

[3;5;7] → ([3;5;7], [])

[4;3;7] → ([4], [3;7])

---

intuitivamente:

```
[] -> ([], [])
| [x] -> ([x], [])
| x::y::ys when x<y ->
```

```
    let (l1, l2) = split (y::ys)
    in (x::l1, l2)

| x::y::ys when x >= y ->
        ([x], y::ys)
```

```
let rec split l = match l with
    [] -> ([], [])
  | [x] -> ([x], [])
  | x::y::ys when x >= y -> ([x], y::ys)
  | x::y::ys when x < y ->
        let (l1, l2) = split (y::ys)
        in (x::l1, l2) ;;
```

(x) No

```
x::y::ys when x < y ->
    let (l1, l2) = split ys
    in (x::y::l1, l2)
```

No!

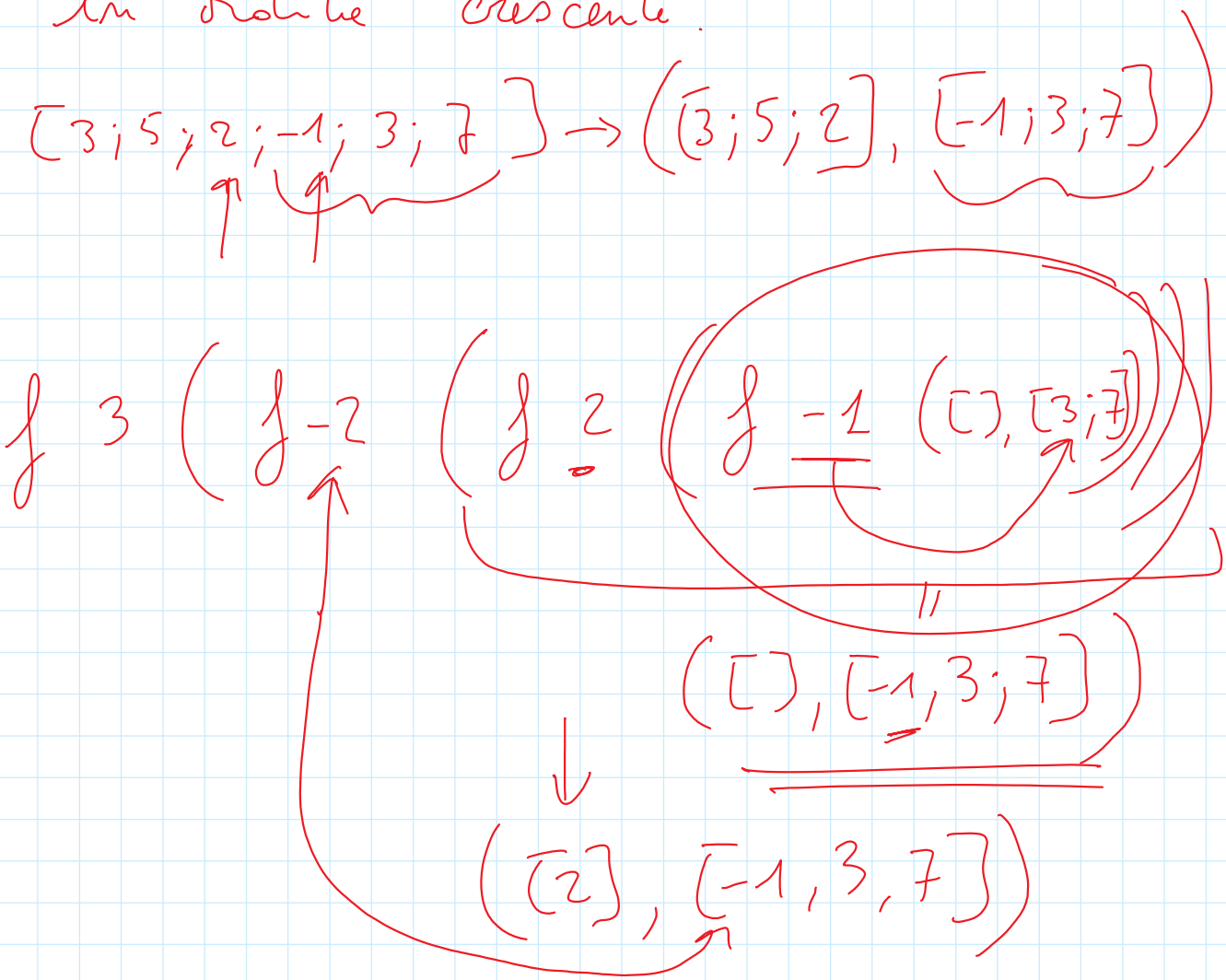
Si definisca, usando le folds,

split : int list → int list * int list

che data una lista l, restituisca la coppia (l1, l2) tale che:

- l1 @ l2 = l
- l2 è la sottolista FINALE più lunga possibile i cui elementi sono in ordine crescente.

[3;5;2;-1;3;7] → ([3;5;2], [-1;3;7])



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

l2=[] then (l1,[x],b) else if

```
let split l =
  let f x (l1, l2) =
    if l1 <> [] then (x::l1, l2)
    else if l2 = [] then (l1, [x])
    else if x < hd l2
      then (l1, x::l2)
      else ([x], l2)
  in foldr f ([], []) l;;
```