ESERCITAZIONE PER LA VERIFICA INTERMEDIA

→ FUNZIONI RICORSIVE
→ FUNZIONI SENZA RICORSIONE ESPLICITA

} CAML

PROVA SCRITTA DEL 15/7/2015

ESERCIZIO 3

twice : 'a → 'a list → bool

Tale che Twice x xs = True se x occorre esattamente 2 volte in xs, false altrimenti

CON RICORSIONE ESPLICITA

```
let Twice x xs =
      let rec conta y ys =
            match ys with
               [] -> 0
             | z::zs -> if z=y Then 1+ (conta y zs)
                                    else conta y zs
      in
            conta x xs = 2 ;;
```

VERSIONE PIÙ EFFICIENTE CHE SI FERMA NON APPENA TROVA 3 COPIE DELL'ELEMENTO

```
let Twice x xs =
    let contam n y ys =
        match (n, ys) with
           (0, []) -> True
          | (m, []) when m>0 -> false
          | (0, z::zs) -> if z=y then false
                              else contam 0 y zs

          | (m, z::zs) when m>0 ->
                          if z=y then
                              contam m-1 y zs
                          else
                              contan m y zs

  in
     contan 2 x xs ;;
```

giovedì 14 dicembre 2017 16:38

SENZA RICORSIONE ESPLICITA

[3; 5; 7; 4; 2; 4; 8]   4

```
let Twice z l =
    let f x y = if x=z then 1+y
                       else y
    in
        (foldr f 0 l) = 2 ;;
```

giovedì 14 dicembre 2017 16:44

VERIFICA SCRITTA DEL 16/12/2015

ESERCIZIO 2

sumflat : int list list -> int list

data una lista di liste di interi fa la somma di tutti gli elementi di ogni lista

[[8;2];[5;4;1];[];[7;3]]

⇓

[10;10;0;10]

SOLUZIONE RICORSIVA

```
let rec sumflat lis =
        let rec sum l =
            match l with
              [] -> 0
            | x::xs -> x+(sum xs)
        in
          match lis with
            [] -> []
```

```
| e::es -> (sum e) :: (sumflex es) ;;
```

SOLUZIONE SENZA RICORSIONE ESPLICITA

[[8;2];[4;1;3];[];[7;1]]

[10; 8 ; 0 ; 8]

NON HO BISOGNO DI PORTARMI DIETRO DELLE INFORMAZIONI DA UNA LISTA DI INTERI ALL'ALTRA .... MI BASTA LA MAP

```
let sumfelt lis =
      let sum l =
            let f x y = x+y
            in foldn f 0 l
      in
         map sum lis ;;
```

II VERIFICA INTERMEDIA DEL 2016

ESERCIZIO 3

canc : 'a list → 'a → 'a list

tale che (canc lis n) cancella l'ultima occorrenza di n in lis

CON RICORSIONE ESPLICITA

$$[3;\overbrace{4;5;\underline{4};2;1}] \qquad 4$$

```
let rec canc l n =
    let rec member x lis =
        match lis with
          [] -> false
        | z::zs -> if (x=z) then true
                            else member x zs
  in
    match l with
    [] -> []
  | x::xs -> if x=n then
| x::xs -> if x=n then
              if member n xs then
                   x :: (canc xs n)
              else xs
           else
              x :: (canc xs n) ;;
```

```
if (x = n && not member n xs)
   then xs
   else x :: (canc xs n)
```

SOLUZIONE SENZA RICORSIONE ESPLICITA

[8;4;7;4;5]   4

[] → [], false
        y

5::[] → 5::[], false
x

4::[5] → 5::[], True

7::[4;5] → 7::[5], True

4::[7;4;5] → 4::[7;5], True

⋮

```
let canc l n =
    let f x (y1,y2) =
        if x = n && not y2 then
                (y1 , True)
            else
                (x::y1, y2)
  in
     let (lis, b) = foldr f ([],false) l
        in
           lis ;;
```

APPELLO 19/01/2017

ESERCIZIO 4

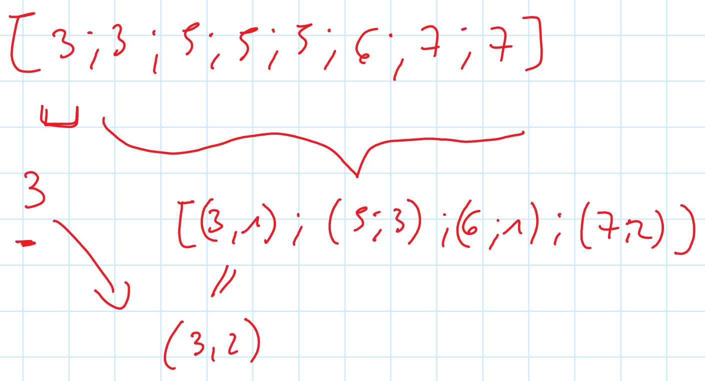
multiset : 'a list → ('a * int) list

[3;3;5;5;5;6;7;7]

⇓

[(3,2);(5;3);(6;1);(7,2)]

SOLUZIONE RICORSIVA



```
let rec multiset lis =
    match lis with
        [] -> []
       | x::xs -> let l= multiset xs
                  in match l with
                     [] -> (x,1)::[]
```

```
| (y,n)::ys -> if (x=y) Then
                    (y,n+1)::ys
                else
                    (x,1) :: (y,n)::ys;;
```

SENZA RICORSIONE ESPLICITA

[3;3;4;5;5;5;7;7]

[] → []

7::[] → (7,1)::[]

7::[7] → (7,2)::[]

5::[7;7] → (5,1)::[(7,2)]

```
let multiset lis =
    let f x y =
        match y with
         [] -> (x,1)::[]
         (z1,z2)::zs -> if z1=x then
                           (z1,z2+1)::zs
                        else
                           (x,1)::(z1,z2)::zs
  in
    foldr f [] lis ;;
```