PROGRAMMAZIONE I (A,B) - a.a. 2017-18
II Verifica Intermedia – 20 Dicembre 2017

Esercizio 1

Si definisca in CAML una funzione ricorsiva in modo esplicito

`cancella : int -> 'a list -> 'a list`

che, dato un intero `n` e una lista, elimina gli ultimi `n` elementi della lista.

SOLUZIONE Prima soluzione, utilizzando la funzione ausiliaria `length` per calcolare la lunghezza della lista e la funzione ausiliaria `cancella_aux` che viene usata per scorrere i primi `length lis - n` elementi eliminando i successivi `n`.

```
let cancella n lis =
  let rec length lis2 =
    if lis2=[] then 0 else 1 + length (tl lis2)
  in
    let rec cancella_aux m lis3 =
      match m,lis3 with
          m,xs when m<=0 -> []
        | m,[] when m>0 -> []
        | m,x::xs when m>0 -> x::(cancella_aux (m-1) xs)
    in
      cancella_aux (length lis - n) lis;;
```

Soluzione alternativa che esegue un'unica scansione della lista. Questa volta la funzione ausiliaria cancella gli elementi uno alla volta partendo dal fondo. La funzione restituisce anche il contatore degli elementi cancellati. Quando il contatore raggiunge `n` la funzione smette di eliminare gli elementi e li inserisce uno alla volta nel risultato.

```
let cancella n lis =
  let rec cancella_aux n lis2 =
    match lis2 with
     | [] -> [],0
     | x::xs -> let (lis3,k) = cancella_aux n xs
                in
                  if k<n then (lis3,k+1) else (x::lis3,k)
  in
    let (ris,q) = cancella_aux n lis
    in ris;;
```

Esercizio 2

Si definisca in CAML la funzione `cancella` descritta nell'Esercizio 1, senza usare la ricorsione esplicita.

SOLUZIONE La funzione `f` passata alla `foldr` si porta dietro il contatore degli elementi eliminati (non inseriti nella soluzione)

```
let cancella n lis =
  let f x (lis2,c) =
    if (c<n) then (lis2,c+1)
             else (x::lis2,c)
  in
    let (ris,c2) = foldr f ([],0) lis
    in ris;;
```

Esercizio 3

Si definisca in CAML una funzione ricorsiva in modo esplicito

`inserisci : 'a list -> 'a -> 'a -> 'a list`

tale che `inserisci lis x y` inserisce nella lista `lis` una nuova occorrenza dell'elemento `x` immediatamente dopo l'ultima occorrenza dell'elemento `y`. Se `y` non è presente nella lista, `x` non viene inserito.

SOLUZIONE Si usa la funzione ausiliaria `member` per verificare, ogni volta che si incontra `y`, se si tratta dell'ultima occorrenza nella lista.

```
let rec inserisci lis x y =
  let rec member z lis2 =
    match lis2 with
        [] -> false
      | w::ws -> w=z || member z ws
  in
    match lis with
        [] -> []
      | w::ws -> if (w=y && not member y ws) then w::x::ws
                    else w::(inserisci ws x y);;
```

Esercizio 4

Si definisca in CAML, senza usare la ricorsione esplicita, una funzione

`primidiliste : 'a list list -> 'a list`

che, data una lista di liste `lis`, restituisce la lista contenente il primo elemento di ogni lista NON VUOTA contenuta in `lis`.

SOLUZIONE Prima soluzione: si usa la `filter` per scartare tutte le liste vuote e la `map` per estrarre il primo elemento da ognuna delle rimanenti (passandole la funzione `hd`).

```
let primidiliste ll =
  let nonvuota lis = lis<>[]
  in
    map hd (filter nonvuota ll);;
```

Seconda soluzione: tramite `foldr` si esamina una lista alla volta e si inserisce nel risultato il primo elemento delle liste non vuote incontrate.

```
let primidiliste ll =
  let f x y =
    if x=[] then y else (hd x)::y
  in
    foldr f [] ll;;
```